

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNICAL REPORT

AN ANALYSIS OF A SIMPLE P2P PROTOCOL FOR FLASH CROWD
DOCUMENT RETRIEVAL

Dan Rubenstein and Sambit Sahu

CU/EE/TR 2002-01-311

Columbia University
Department of Electrical Engineering
500 W. 120th Street, MC 4712
New York, NY 10027

<http://www.ee.columbia.edu>

Copyright © 2002 – The Trustees of Columbia University in the City of New York
All Rights Reserved

The technical reports in this series are considered to be semi-formal.
The ideas expressed are solely those of the authors, and questions about the content
should be directed to them.

An Analysis of a Simple P2P Protocol for Flash Crowd Document Retrieval

Dan Rubenstein
Dept. of Electrical Engineering
Columbia University
New York, NY
danr@ee.columbia.edu

Sambit Sahu
IBM T.J. Watson Research Center
Hawthorne, NY
sambits@us.ibm.com

January 31, 2002

Abstract

Today's Internet periodically suffers from hot spots, a.k.a., flash crowds. A hot spot is typically triggered by some unanticipated news event that causes an unanticipated surge of users to 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 mathematically model and simulate a simple, distributed, randomized peer-to-peer (P2P) protocol. This protocol runs atop a network overlay during a hot spot to quickly deliver the "hot" object from a small set of users who were able to obtain the object to a much larger set of users that desire it. Our results based on this evaluation show that this type of protocol can scale effectively to millions of users, demonstrating that peer-to-peer approaches can be an effective means to alleviate problems commonly associated with network hot spots.

1 Introduction

The Internet has become a primary source for retrieval of timely information such as stock quotes and news. Barring exceptional circumstances, the time between a user's request and the arrival of the response to that request is several seconds. This is in part due to content distribution services like Akamai [15] and Digital Island [9] that handle the transmission of significant portions of data that a content provider would otherwise need to transmit itself. However, even with these content distributors in place to handle a large bulk of data transmission, several architectural and economic factors require that users first visit the content provider's site. From there, a small *container page* is returned that informs the user's browser of more efficient locations to retrieve the remaining requested data.

While often robust, the current content distribution architecture has its flaws. September 11 is a recent striking example of a situation in which the current infrastructure failed to handle massive increases in requests to popular news sites. At the most critical times, only a small number of users could retrieve the heavily desired web objects whereas the overwhelming majority of users could not. This is in spite of ample evidence that network connectivity remained intact. For example, individuals could continue to communicate via e-mail, instant messaging and IP telephony [16]. From the perspective of a single user, the massive overloads of requests to a content provider/creator for their content, commonly dubbed *hot spots* or *flash crowds*, are rather infrequent. However, their occurrence provides an incredibly frustrating experience for the large set of users that have quickly developed a desire to retrieve a particular data object.

This paper proposes the use of a simple, client-side, peer-to-peer (P2P) protocol to provide an alternative means of delivering the otherwise inaccessible content and provides a preliminary analysis of this protocol's performance. The protocol allows users to retrieve duplicates of recently generated objects (e.g., timely news web pages) that originate at a server that is currently exhibiting hot spot symptoms via end-to-end communications among peers. The protocol's ability to deliver the content during the hot spot leverages off the fact that these pairwise end-to-end communications remain operational during hot spot periods. The protocol operates by having users organize themselves (prior to a hot spot event) 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. The challenge is to develop a protocol that, using queries upon the overlay, locates the object in a short period of time without saturating the network with its search requests.

The protocol that we consider for this purpose uses a series of distributed, randomized, scoped searches within the overlay to locate the object. Our goal is to produce a protocol that is not only sufficiently scalable but is also easy to implement. In this regard, we avoid distinguishing the roles of the participants, i.e., there are no hierarchies, and we do not concern ourselves with geographical position of nodes. The only state that a node must maintain is its set of neighbors within the overlay and a copy of the object (once retrieved) so that it can assist in subsequent searches.

This paper formally demonstrates that distributed, randomized searches for content similar to those used within Gnutella [10] used specifically to provide delivery of a small number of hot spot objects efficiently scale to large overlays. Here, we provide an analysis of our proposed protocol via a combination of mathematically derived exact and upper bounds and simulation results that quantify the time taken and number of transmissions performed by the protocol. We demonstrate that in theory, use of this type of protocol to retrieve "hot" content will scale within P2P overlays that connect up to and perhaps beyond one million participating users, regardless of the fraction of these users that seek the popular content. We demonstrate 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 no more than 400 transmissions. 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 results that we report in this paper are of a more theoretical nature to allow us to focus on the protocol's scalability. We relegate the task to dealing with more practical issues such as failing nodes and malicious users for future work.

The rest of this paper proceeds as follows. Section 2 discusses related work. In Section 3, we describe the network model and the protocol within the context of that model. In Section 4, we describe a process for constructing "good" topologies for our protocol within a practical setting. Section 5 analytically evaluates temporal and bandwidth performance of the protocol when executing a single node's search on a "good" overlay topology. Section 6 extends the analysis when multiple nodes search for the object. Section 7 evaluates the performance using these analytical results and simulations. Section 8 provides discussion and future work and Section 9 concludes the paper.

2 Related Work

The recent attention focused upon peer-to-peer (P2P) and overlay networking paradigms in the networking and performance communities has concentrated mainly on measurement studies of existing and deployed P2P protocols [18, 20], as well as on experimental experiences with application layer approaches that extend networking functionality in areas such as alternate path routing [21, 1], multicast [11, 6, 5, 4], and anycast [23]. More recent work has introduced clever object searching schemes within P2P overlays that efficiently locate content that is static (unchanging) and easily labeled [17, 22]. These schemes were designed to efficiently point large sets of users looking for different “cold” objects to different parts of the overlay. However, they point sets of users looking for the same “hot” objects to similar parts of the network. While it is possible to have multiple destinations for a single object, the number of destinations must be planned in advance and must be approximately the same for all network objects. Thus, it is difficult to adapt the number of destinations dynamically to those objects that are producing hot spot conditions in the network. In addition, the requirement that objects have a simple, static labeling system complicates the use of such schemes for handling temporally-based queries such as “locate a copy of object X generated within the last 2 minutes”.

Network paradigms such as multicast [7] and anycast [23, 13] can also potentially be used to retrieve “hot” objects at a relatively low cost. However, multicast is problematic in that it is difficult to appropriately scope queries to prevent flooding within the network of the numerous and simultaneous user requests. Anycast is a paradigm that can be used to direct users toward massively replicated content. However, anycast itself does not solve the problem of automating the process of dynamically replicating the content when a hot spot arises.

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 [2] 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 [14, 8, 12]. Most closely related to this work is the work of [12]. There, the authors consider a combination push-pull architecture where a node either pushes “hot gossip” from its neighbors, or a neighbor can pull “hot gossip” to its neighbors. 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 [19]. 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.

3 Model

In this section, we introduce a simple graphical model of the overlay network and, in the context of this model, the P2P protocol that we subsequently analyze. We emphasize here that the purpose of this paper is not to provide a P2P protocol that is ready for deployment, but to demonstrate the intrinsic scalability of P2P, ring-scoped distributed search protocols for retrieving popular information from web servers under hot spot conditions. To focus directly on these issues, we make several simplifying assumptions about the

network and the operation of the protocol. As a result, there are several practical considerations that this paper does not address that may indirectly affect scalability, such as handling node joins, departures, and failures, dealing with uncooperative users who seek objects but do not assist in others’ searches, and making the system robust to against being used as an attack tool. These practical considerations require practical solutions that are beyond the scope and outside of the theme of this paper.

3.1 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. A node, n_1 has the functional ability to communicate directly with any node in the overlay graph, just as any host can communicate with any other host within the Internet by forwarding a packet to the appropriate IP address. However, n_1 may not know the identities of all other nodes in the overlay. We say that a node, n_2 , is a **neighbor** of n_1 and draw a directed edge $\langle n_1, n_2 \rangle$ from n_1 to n_2 in \mathcal{G} whenever n_1 has knowledge of n_2 ’s identity (i.e., its IP address). The path taken by a communication between a pair of users (nodes) in the actual Internet will likely traverse a series of communication devices such as LANs, routers, and/or wireless devices that are shared with other transmissions that utilize the underlying network. The precise time that it takes for information to propagate from one node to another within the overlay is unpredictable in real-life settings. However, it is reasonable to assume that a threshold, t_r , bounds the time within which, to a very high probability, two nodes can exchange communications. Any round-trip communications that exceed this bound can be considered lost. Note that this gives a node an opportunity to assess the status of a neighbor, e.g., to determine if perhaps the neighbor has left the overlay.

Because overlays are implemented within the application layer of the network protocol stack, different overlays can be deployed simultaneously, allowing each application to utilize an overlay topology that specifically meets its needs. Recent work has explored the challenges of forming “good” overlay topologies for specific application needs such as alternate path routing [21, 1], multicasting [11, 6, 5, 4], and searches for obscure content [22, 17]. We therefore assume that the overlay topology upon which our P2P recovery protocol runs can be constructed specifically for this application.

Our mathematical analysis makes a simplifying assumption that the overlay graph is in fact fully connected, i.e., $\forall n_1, n_2 \in \mathcal{N}, \langle n_1, n_2 \rangle \in \mathcal{E}$. When N is small, this assumption is reasonable. However, one cannot expect all nodes to maintain the identities of all other nodes in the overlay when N grows large. We address this limitation of our analysis through simulation where we consider overlay topologies in which each node has a fixed bound, C , on the size of its neighbor set. In Section 4, we present a distributed algorithm technique can be used in practice using a process we call *k-shuffling*. In the appendix, we demonstrate that our protocol’s performance on a graph that has been *k*-shuffled is almost identical to that of its performance on a fully-connected graph. It follows that the performance of the protocol demonstrated by our theoretical results can indeed be realized in practice by constructing an overlay for hot spot alleviation using *k*-shuffling.

We examine the cost measured in time and bandwidth of retrieving 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 $0 < N_h \leq N_d \leq N$. We again emphasize that the P2P protocol will not function properly if $N_h = 0$. However, in practice, as N grows large, we expect the likelihood of no nodes in the overlay having a copy of the object to be negligible, i.e., somebody participating in the overlay was able to retrieve the page either before or during the overloaded conditions. We assume that the N_h nodes that initially have the object chosen from the set of nodes via uniform sampling without replacement.

3.2 The P2P Protocol

In this subsection, we formally describe the operation of the distributed, P2P, scoped search protocol in the context of our network model. The reader familiar with this area of work will find this protocol similar to existing P2P protocols such as that used within Gnutella. The novelty of our work here is its application to the problem of retrieving copies of massively requested objects.

We assign the name **LocateObject** to the protocol that runs upon an overlay to deliver an object to those users that desire it. We assume the existence of another protocol, termed **ConstructOverlay**, that constructs the overlay graph which determines the set of neighbors for each node, upon which searches for objects take place. **ConstructOverlay** is run continuously (i.e., as a daemon) prior to, during, and after any hot spot activity, adapting the underlying topology to users whose browsers choose to dynamically enter into or exit from the overlay. When users are unable to retrieve the desired object through conventional means, the user activates **LocateObject**.

The **LocateObject** protocol propagates *queries* throughout the overlay network. A query is a 3-tuple, $\{\mathcal{O}, addr, TTL\}$, where \mathcal{O} is a description of the object being requested, *addr* is the identity (IP address) of the user or node to which the object should be delivered once located, and *TTL* is an integer that limits the additional distance (in overlay hops) that a query can travel. Note that it is permissible for the object description, \mathcal{O} , to contain temporal restrictions, such as “return a copy of web page X that was generated within the past 2 minutes”. Queries are transmitted between nodes. Hence we use the terms *query* and *transmission* interchangeably.

When a node n seeks a given object \mathcal{O} and chooses to initiate a search, it constructs the query $\{\mathcal{O}, A, h\}$ where A is its identity, and $h \geq 0$ is an integer. Such a node is referred to as a *query-former*. The query-former then selects a subset of its neighbors and forwards the query to these neighbors. The time when the node chooses to initiate a search, the value to which h is set, and the set of neighbors selected are discussed below.

Our description and evaluation of the protocol discretizes time into *rounds*. Within a round, a node n_i 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 a given round. In other words, for a query to travel along the path from n_0 to n_1 , from n_1 to n_2 , \dots , from n_{k-1} to n_k in the overlay where $n_{i+1} \neq n_{i-1}$ requires k rounds. In practice, one can set the period of a round to a time longer than t_r (discussed above) to permit nodes to acknowledge queries they have received. We assume that a node that receives a query for an object that it has cached can return that object within the same round.

When a node n receives a query $\{\mathcal{O}, A, h\}$, it searches its local cache for a copy of the requested object. If found, n forwards \mathcal{O} to the node with identity A (i.e., the query-former) within the same round that n received the query. Otherwise, n modifies the query to $\{\mathcal{O}, A, h - 1\}$, decrementing the TTL value. If the modified TTL is positive, n then selects a subset of its neighbors, and forwards the query to this subset on the next round. Otherwise, if the TTL is 0 or smaller, it does nothing (and ignores the query).

The protocol **LocateObject** is itself most easily described as a tuple, $\mathcal{P} = \{\mathcal{I}, \langle \eta(1), \dots, \eta(\mathcal{I}) \rangle, f\}$. \mathcal{I} is the number of *iterations*. Iterations are performed in sequence, where the $i + 1$ st iteration is initiated only if the object is not returned upon completion of the i th iteration. When the i th iteration starts, the query-former transmits its query $\{\mathcal{O}, A, \eta(i)\}$ to f neighbors selected at random. The $i + 1$ st iteration is initiated only when no copies of \mathcal{O} are returned within $\eta(i)$ rounds after the initiation of the i th iteration. Note that by this time, all queries from the previous iteration would have terminated, since the TTL would have reached 0. If the object is not found after all \mathcal{I} queries complete, the protocol can be repeated indefinitely. f specifies the fanout to be used within the protocol. This is the number of neighbors to which a node forwards a query whenever it must forward a query onward.

We make several observations about **LocateObject**:

- The potential number of nodes contacted by the protocol within the i th iteration increases exponentially with $\eta(i)$. Hence, one needs to be careful when setting $\eta(i)$: if it is too large, then an excessive number of additional transmissions will occur. If too small, then it may take a long time (many rounds) to reach a node with the object, \mathcal{O} .
- `LocateObject` is stateless (though it uses state from `ConstructOverlay` to identify neighbors). Nodes do not remember information from queries received in previous rounds, the nodes from which these queries were transmitted, or the nodes to which queries were subsequently forwarded. In fact, our analysis assumes a protocol that is so stupid that neighbors are selected at random with replacement (i.e., the same query might be forwarded to the same neighbor within the same round!), and that a node might be its own neighbor and forward a query to itself.
- Once an iteration commences, this stateless nature of the protocol prevents the termination of searches within that iteration other than by reaching nodes containing the requested object, or by TTL expiry. In other words, once an iteration is initiated, it must be allowed to run its course.
- It is possible for different hops of the query to use different values for f . For simplicity of presentation, we assume all nodes utilize the same value for f . We have performed a preliminary exploration as to how varying f dynamically as queries progress through the network. We have not observed any striking differences and have not included this discussion in this paper due to lack of space.
- While we do not specifically address uncooperative users here, we wish to make one comment. If an uncooperative user is willing to admit that it will not forward packets further, then the protocol can operate without any change in performance by ensuring that the set of neighbors consists mainly of cooperative neighbors and that these neighbors are chosen during searches. Intuitively, an uncooperative neighbor that hides its lack of participation will likely increase the number of rounds to find content, but in fact reduce the number of transmissions since iterations are effectively being terminated prematurely.

3.3 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 receive until 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. Retrieval for multiple objects can be obtained by summing the bandwidths of the retrievals for the individual objects (i.e., we assume searches for different objects are independent).

Note that since there is a one-to-one correspondence between the set of all queries that are transmitted by nodes to the set of all queries that are received by nodes, the expected number of queries received by a node also equals the expected number of queries transmitted by a node. This is somewhat counter-intuitive since a node often sends f queries as a result of receiving a single query. However, a large fraction of the time, a node does not forward a query any further.

- **Time (in rounds):** we measure time simply by the number of rounds from the round in which a query-former initiates its first query until the round that the object is returned to the query-former.

¹Note 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 is likely to be spread out evenly over all nodes, even in the case where users’ searches are performed sequentially.

Our analytical results are performed upon two arrival patterns of initial search requests from users:

- **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 i th user has already received the object.
- **Simultaneous Attempts:** this is the case where all users actively seeking the object issue their queries simultaneously. We assume that the starting times of the users’ searches are independent, such that at any given time t , not all users’ searches are in the same round and iteration of the protocol. However, we do assume the period of a round is the same for all nodes, i.e., all protocols move to the next round at the same rate. Our computations determine upper bounds on the expected number of transmissions and rounds for this case.

4 Shuffling

In the next section, we will analyze the `LocateObject` protocol under the assumption that the overlay network is fully connected. Here, we justify that analysis by presenting a practical protocol that constructs an overlay (i.e., protocol `ConstructOverlay`) in which each node n_1 maintains a bounded set of neighbors where the probability that a given node n_2 is a neighbor is the same as it would be if neighbors were drawn from the set of all nodes with uniform distribution. We show through simulation that the probability of reaching a random set of nodes in this overlay using the `LocateObject` protocol is essentially the same as that in a fully connected graph.

The overlay is formed by a simple process we term *shuffling*, which consists of a series of *k-shuffle* operations performed by members of the overlay, where $1 \leq k < C$ is a fixed integer. Let $\sigma_i(t) = \{n_{i_1}(t), n_{i_2}(t), \dots, n_{i_C}(t)\}$ be the set of neighbors of node n_i at time t . n_i shuffles its neighbors to produce a new set of neighbors, $\sigma_i(t + \Delta t)$ as follows. A subset, $\gamma \subset \sigma_i(t)$ of size $k - 1$ is formed, where nodes are drawn uniformly at random without replacement. A final node n_j is drawn from the remaining set and $\gamma \cup \{n_j\}$ is forwarded to n_j .

Upon receiving the set γ , n_j selects a set of nodes γ' , first by including any nodes $n_\ell \in \sigma_j(t) \cap \gamma$, and subsequently selecting the remaining nodes at random with replacement. It forwards γ' back to n_i , and sets $\sigma_j(t + \Delta t) = (\sigma_j(t) \setminus \gamma') \cup \gamma$. n_i sets $\sigma_i(t + \Delta t) = (\sigma_i \setminus (\gamma \cup \{n_j\})) \cup \gamma'$. n_i and n_j have effectively swapped k neighbors, n_j now has n_i as a neighbor, and n_i has removed n_j as its neighbor.

The shuffling operation not only mixes the set of neighbors of a node, but also mixes the nodes with which the node exchanges its neighbors. There are several ways in which a node can decide the point in time to perform a shuffle. One possibility is to initiate a shuffle after a fixed period of time, or after a time that is exponentially distributed. Another possibility is to initiate a shuffle after being on the receiving end of a shuffle. The shuffling process itself requires bootstrapping: i.e., a node must first make contact with other nodes that are already part of the overlay, and mechanisms must be put in place to prevent partitioning of the overlay graph when nodes choose to exit the overlay. These issues are beyond the scope of this paper.

5 Performance Analysis of `LocateObject`

In this section, we develop the mathematical tools that will allow us to evaluate the bandwidth and time overheads of `LocateObject`. For simplicity of analysis, we assume the overlay graph is fully connected - an impractical assumption for a real overlay, but as discussed in the previous section, it is possible to generate overlays that will exhibit similar properties for the purpose of this analysis. 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 $1 - \nu$.

For a single query, up to f^j transmissions can occur within the j th round of an iteration. We call each such potential transmission a *transmission slot*. Those slots that represent actual transmissions that are performed during an instance of the running of the protocol are called *real transmissions*. Note that under the assumption that transmission destinations are selected uniformly at random from the set of all overlay nodes, the probability that a transmission slot that takes place in the j th round of the i th iteration is a real transmission, given that the i th iteration occurs, is $1 - \nu^{j-1}$.

Let $p_i(\nu)$ be the probability that a query would require more than $\eta(i)$ rounds to locate the object, given that a fraction ν of nodes hold the object. This equals the probability that the object is not found after $f + f^2 + \dots + f^{\eta(i)}$ transmissions. Hence,

$$p_i(\nu) = \nu^{\frac{f^{\eta(i)+1} - f}{f-1}}. \quad (1)$$

We first turn our attention to determining the expected number of rounds taken by protocol \mathcal{P} to either locate the object or terminate (without locating the object). Let $L_i^a(\nu)$ be a random variable that equals 0 when no transmission in the i th iteration arrives at a node with the object and 1 otherwise. Then $\Pr[L_i^a(\nu) = 0] = p_i(\nu)$. We define $L_i^b(\nu)$ to be a random variable that equals 0 when no transmission slot reaches a node with the object by the end of the i th iteration, and 1 otherwise. It follows that $\Pr[L_i^b(\nu) = 0] = \prod_{j=1}^i \Pr[L_j^a(\nu) = 0]$. By convention, we set $\Pr[L_0^b(\nu) = 0] = 1$.

Define $R_i(\nu)$ to be a random variable to equal the number of rounds in which a query, with TTL set to $\eta(i)$, is propagated prior to locating the object. $R_i(\nu) = k$ means that either the object was retrieved at the end of the k th round, or, for $k = \eta(i)$, it can also mean that the object was not retrieved.

$$E[R_i(\nu)] = \sum_{j=0}^{\eta(i)} j \Pr[R_i(\nu) = j] = \sum_{j=0}^{\eta(i)-1} \Pr[R_i(\nu) > j] = \sum_{j=0}^{\eta(i)-1} p_j(\nu) \quad (2)$$

Let $R_{\mathcal{P}}(\nu)$ 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

$$E[R_{\mathcal{P}}(\nu)] = \sum_{i=1}^{\mathcal{I}} \Pr[L_{i-1}^b(\nu) = 0] E[R_{\eta(i)}(\nu)] \quad (3)$$

We next turn our attention to determining the expected number of transmissions a single running of protocol \mathcal{P} makes (regardless of whether or not the object is located). We define $T_i(\nu)$ be the number of real transmissions that occur during the i th iteration of \mathcal{P} . Fixing ν , let T_j^s be the number of transmissions in the j th round such that $T_i(\nu) = \sum_{j=1}^{\eta(i)} (i) T_j^s$. Noting that there are f^j transmission slots that can occur in the j th round, we construct the random variable $X_{i,j,k}$ that equals 1 if the j th round's k th transmission slot within the i th iteration is a real transmission and 0 otherwise. Then $E[X_{i,j,k}] = \Pr[L_{i-1}^b(\nu) = 0] \nu^{j-1}$.

$$\begin{aligned} E[T_i(\nu)] &= \sum_{j=1}^{\eta(i)} E[T_j^s] = \sum_{j=1}^{\eta(i)} \sum_{k=1}^{f^j} E[X_{i,j,k}] = \Pr[L_{i-1}^b(\nu) = 0] \left(f + \nu f^2 + \dots + \nu^{\eta(i)-1} f^{\eta(i)} \right) \\ &= \Pr[L_{i-1}^b(\nu) = 0] f \frac{1 - (f\nu)^{\eta(i)}}{1 - f\nu} \end{aligned} \quad (4)$$

Let $T_{\mathcal{P}}(\nu)$ be a random variable equaling the number of transmissions performed by the protocol.

$$E[T_{\mathcal{P}}(\nu)] = \sum_{i=1}^{\mathcal{I}} E[T_i(\nu)] \quad (5)$$

Having computed the expected number of rounds and transmissions of a single running of the protocol, we can easily extend this to the expected number of rounds and transmissions when the protocol is repeated until the object is located. Let $T_{\mathcal{P}}^i(\nu)$ and $R_{\mathcal{P}}^i(\nu)$ respectively be the number of transmissions and rounds needed during the i th running of the protocol.

$$E[T_{\mathcal{P}}^i(\nu)] = \Pr[L_{\mathcal{I}}^b(\nu) = 0]^{i-1} E[T_{\mathcal{P}}(\nu)] \quad (6)$$

$$E[R_{\mathcal{P}}^i(\nu)] = \Pr[L_{\mathcal{I}}^b(\nu) = 0]^{i-1} E[R_{\mathcal{P}}(\nu)] \quad (7)$$

We define $T_{\mathcal{P}}^+(\nu)$ and $R_{\mathcal{P}}^+(\nu)$ 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.

$$E[T_{\mathcal{P}}^+(\nu)] = E\left[\sum_{i=1}^{\infty} T_{\mathcal{P}}^i(\nu)\right] = \sum_{i=1}^{\infty} E[T_{\mathcal{P}}^i(\nu)] = \frac{E[T_{\mathcal{P}}(\nu)]}{1 - \prod_{j=1}^{\mathcal{I}} \Pr[L_j^a(\nu) = 0]} \quad (8)$$

$$E[R_{\mathcal{P}}^+(\nu)] = E\left[\sum_{i=0}^{\infty} R_{\mathcal{P}}^i(\nu)\right] = \sum_{i=0}^{\infty} E[R_{\mathcal{P}}^i(\nu)] = \frac{E[R_{\mathcal{P}}(\nu)]}{1 - \prod_{j=1}^{\mathcal{I}} \Pr[L_j^a(\nu) = 0]} \quad (9)$$

6 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.

6.1 Isolated Attempts

We define θ_s and ρ_s 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:

$$E[\theta_s] = \sum_{i=N_h}^{N_d} E[T_{\mathcal{P}}^+(i/N)], \quad E[\rho_s] = \sum_{i=N_h}^{N_d} E[R_{\mathcal{P}}^+(i/N)] \quad (10)$$

6.2 Joined Attempts

We consider a system, S_0 , of users in which for any t , the time (round number) modulo t at which a user initiates its query is uniformly distributed between 0 and $t - 1$. 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 the upper bound on the expected number of transmissions sent/received by a node via the following steps:

1. We compute an expression for the “steady state” expected number of transmissions and rounds required by system \mathcal{S}_0 to increase the number of nodes with the object by one when a fraction, $1 - \nu$, of nodes initially contain the object. By “steady state”, we mean a system in which any node other than the initial $1 - \nu$ nodes that begin with the object, upon locating the object, does not save its copy, but instead restarts its protocol from the beginning, continually searching for the object. This creates a renewal process in which the renewal occurs each time a node receives a copy of the object. In this system, the fraction of nodes searching for the object is always ν . However, for decreasing ν , the likelihood that the renewal reaches later iterations decreases, since it becomes more likely that a copy of the object will be found in an early iteration, causing searcher to restart the protocol from the first iteration.
2. We conservatively count all transmission slots as real transmissions for any iteration that is executed.
3. We show that the “steady state” values 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 $\max\{N_h, m - \sum_{i=1}^T \eta(i)\}$ 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 $\max\{N_h, m - \sum_{i=1}^T \eta(i)\}/N$ of nodes has the object upper bounds 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)²

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 i th tick in system \mathcal{S}_0 and let $G_{i,\nu}$ be a random variable that equals 1 when the object is not located on the i th tick of system \mathcal{S}_0 where a fraction $1 - \nu$ nodes have the object.

$$E[\theta_j(\nu)] = \sum_{i=1}^{\infty} E[T_{i,\nu} \prod_{j=1}^{i-1} G_{j,\nu}] = \sum_{i=1}^{\infty} E[T_{i,\nu}] \prod_{j=1}^{i-1} E[G_{j,\nu}] = \frac{E[T_{1,\nu}]}{1 - E[G_{1,\nu}]} \quad (11)$$

²This assumption greatly simplifies modeling, and we suspect does not have any significant impact on the metrics of interest.

The proof of why the second equality holds, included in Appendix A, 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 onto 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 j th round within the i th 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:

$$\Pr[\sigma = (i, j)] = \frac{\nu^{A(i,j)} / (1 - L_{\mathcal{I}}^b(\nu))}{\sum_{k=1}^{\mathcal{I}} \sum_{\ell=1}^{\eta(k)} \nu^{A(k,\ell)} / (1 - L_{\mathcal{I}}^b(\nu))} = \frac{\nu^{A(i,j)}}{\sum_{k=1}^{\mathcal{I}} \sum_{\ell=1}^{\eta(k)} \nu^{A(k,\ell)}} \quad (12)$$

where $A(j, k) = k + \sum_{s=1}^{j-1} \eta(s)$, i.e., the number of rounds from the start of the protocol to the point at which the k th 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 i th iteration. We upper bound the number of queries transmitted in (i, j) by $a(j, k) = \sum_{s=k}^{\eta(j)} f^s$, which is the number of transmission slots from the k th round of iteration j to the final round $\eta(j)$ 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

$$E[T_{i,\nu}] \leq \sum_{j=1}^{\mathcal{I}} \sum_{k=1}^{\eta(j)} a(j, k) \Pr[\sigma(\nu) = (j, k)] \quad (13)$$

$$E[G_{i,\nu}] = \sum_{j=1}^{\mathcal{I}} \sum_{k=1}^{\eta(j)} \Pr[\sigma(\nu) = (j, k)] \nu^{A(j,k)} \quad (14)$$

Applying Equation (11) and $1 - E[G_{i,\nu}]$ via algebraic manipulation, we get

$$E[\theta_j(\nu)] \leq \frac{\sum_{j=1}^{\mathcal{I}} \sum_{k=1}^{\eta(j)} a(j, k) \nu^{A(j,k)}}{1 - \nu^{A(\mathcal{I}, \eta(\mathcal{I}))}}. \quad (15)$$

We now proceed onto step 3, which is completed via the following Lemma, whose proof is a simple sample-path argument that appears in Appendix A.

Lemma 1 $E[\theta_j(\nu)]$ is non-decreasing with increasing ν .

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

$$E[\theta_j(N_d, N, N_h)] \leq \sum_{i=N_h}^{N_d} E[\theta_j(1 - i/N)]. \quad (16)$$

6.2.1 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

$$E[\tau(a, b, c)] = \sum_{i=0}^{\infty} \Pr[\tau(a, b, c) > i] = \sum_{i=0}^{\infty} E[G_{i, 1-c/b}] / a = \frac{1}{a(1 - E[G_{1, 1-c/b}])} \quad (17)$$

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

The proof is in the Appendix.

7 Evaluation

In this section, we present performance results of the `LocateObject` Protocol via our mathematical analysis from the previous two sections and simulation framework which we now describe. Our simulations were performed on a home-grown, discrete-event simulator that allowed us to experiment with variations in the connectivity of the underlying network graph. In particular, we use simulation to evaluate the effects on our performance metrics when bounding the number of 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.

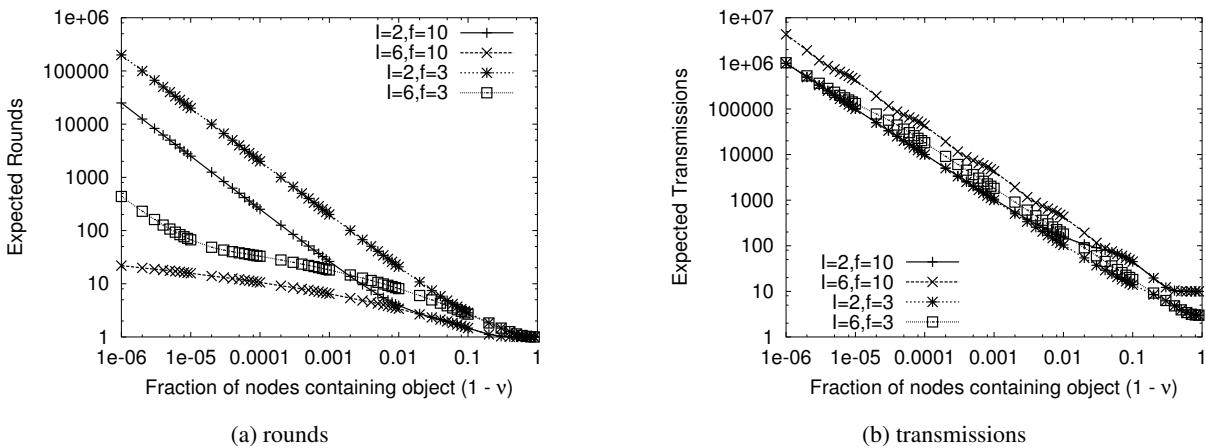


Figure 1: Total rounds taken and transmissions generated by a single node's query for a fixed ν .

Figure 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. Figure 1(a)

plots the expected number of rounds that the search will take to locate the object (Equation 9), and Figure 1(b) plots the expected number of transmissions that the user’s search for the object generates (aggregated over all nodes in the network, Equation (8)). 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 . Throughout this section, we fix $\eta(i) = i$.

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 $1 - \nu \rightarrow 0$. 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.

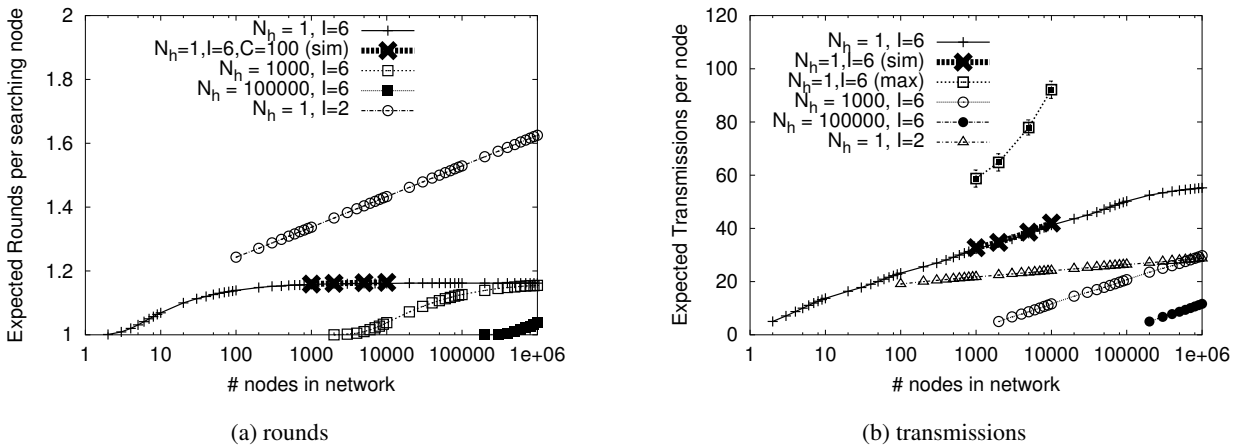


Figure 2: Total rounds taken and transmissions generated by isolated joiners.

Next, we turn our attention to the analysis of the system of in which multiple users wish to retrieve the hot object. We begin by noting that our equations of interest here (10), (16), (17) are increasing functions

of N_d . Thus, setting $N_d = N$ maximizes the expected number of rounds and transmissions, giving us a worst-case scenario for these expectations. We therefore limit our consideration to cases where $N_d = N$.

Figure 2(a) plots the expected number of rounds a user can expect to wait until it retrieves the object. Recall from Figure 1(a) that users whose searches occur earlier in the sequence of user searches are expected to take more rounds than those that occur later. Hence, this expected value is for a user whose position in this sequence is selected from a uniform distribution. Figure 2(b) plots the expected number of transmissions that a node *receives* over *all* queries transmitted in the network when various users' searches are 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.

In Figure 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 10000 (the curve's label contains "(sim)"). In Figure 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 10000 nodes. The parabolic shape of this curve suggests that for large overlays (of one million nodes), it may be the case that a disproportionate load of requests may overwhelm a small set of nodes in the network. Further investigation into this trend is required. 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 the expected number of transmissions received per node since this number is distributed across a large set of nodes³.

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 one million nodes searching for the object.

Last, we examine the case where all receivers' queries are run simultaneously. Figure 3(a) plots 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 Lemma 2. Figure 3(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 (equation (17)).

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 and

³A possible exception is if one is concerned that numerous transmissions between overlay members traverse a common set of links, causing these links to bottleneck.

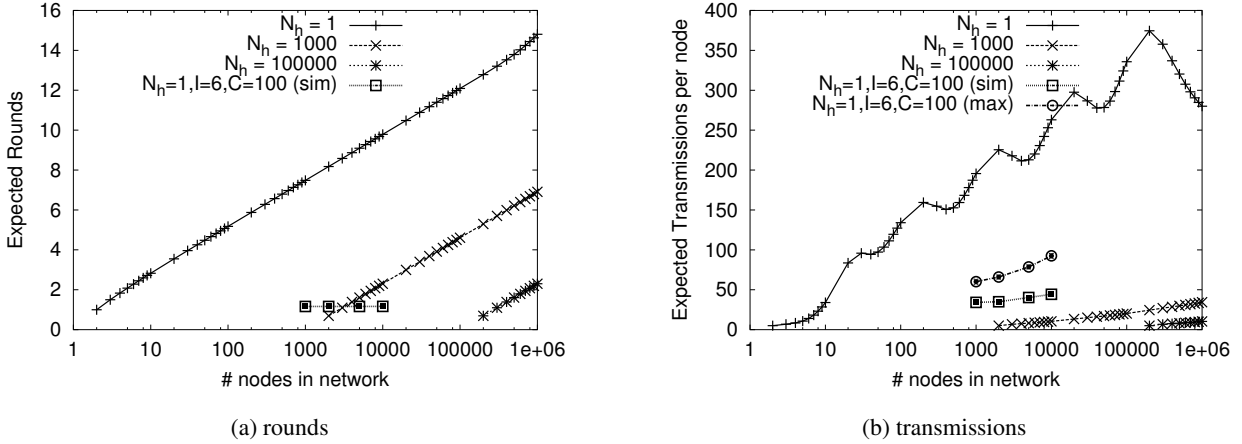


Figure 3: A comparison of rounds and transmissions taken by isolated and simultaneous searches.

2000, 5000, and 10,000 node overlay topologies with $N_h = 1, \mathcal{I} = 6$, and $C = 100$.⁴ For these simulations, we emulate a flash-crowd like environment in which a user that has not initiated its first query by the i th 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 searches tell their friends about 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 some sense, we expect that having all nodes searching simultaneously actually does more good than harm. The reason is that the simultaneous searches increase the likelihood of (multiple) objects being located quickly, rapidly increasing the number of nodes that have a copy, which in turn, rapidly reduces the expected length of subsequent searchers.

In Figure 3(b), we also include a curve that, as in Figure 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 one million users. In addition, the user should expect to receive the object in no more than 16 rounds.

7.1 Practical Implications

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

⁴99% 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.

be fairly easy to fit most queries into a 64-byte packet. Thus, 800 queries translates to 50KB of data to be transmitted in total. A 28.8 baud modem can handle this amount of data within 15 seconds. 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.

8 Discussion

In the previous sections, we have demonstrated that in theory, protocols that use randomized, distributed, scoped searches on a P2P overlay is an effective means for coping with server overloads due to flash crowds. Here, we briefly elaborate on some concerns that should be addressed within a practical implementation.

As described, in the case where the number of nodes that initially begin with the object, $N_h = 0$, copies of the object will not be propagated throughout the network and the number of transmissions sent by all the users in the protocol will flood the network. We note that this flooding can be controlled by each node pacing the set of queries it responds to. For example, a node could keep an exponentially weighted average of the number of queries it services a second. When this average reaches a threshold, the node could probabilistically drop any requests that would otherwise force the window above the threshold, and reduce the number of nodes to which it forwards accepted queries. The latter reaction would significantly limit the number of queries that propagate throughout the network. Such approaches could also be of assistance to protect the network from denial-of-service (DoS) attacks via queries for non-existent objects on the overlay.

Often, a distributed search will locate multiple copies of the object. If the system is to be used to download large objects other than container pages, the query-former could subsequently select from what appears to be the "best" source of the object. Alternatively, the use of sophisticated coding techniques could be used to parallelize the access of the document from these multiple servers [3].

We have not explicitly addressed the problem of malicious or non-cooperative users, or the effects of users that might choose to leave or join the overlay, perhaps as a hot spot is in progress. We leave addressing these challenges as future work. However, the problem of verifying the validity of a page can be easily accomplished if the server that becomes hot is one commonly visited by the user. In this case, the user can obtain a certificate of authenticity from the server during a time when the server is not overloaded. Any web page that was originally generated at the server, whether downloaded directly from the server, from a content distribution service, or by another node in the overlay can then be verified by verifying the certificate within the page.

Last, some costs discussed above could perhaps be alleviated by having users that have received the content subsequently run a gossip-style protocol to more quickly spread the "hot" object throughout the overlay.

9 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 one million users, showing promise in a real network setting.

Acknowledgments

We would like to thank Ed Coffman, Predrag Jelenkovic, Jason Nieh and Henning Schulzrinne at Columbia University for discussions that motivated our pursuit of research on hot spots. We also thank Anees Shaikh and Arup Acharya at IBM Research for discussions on peer-to-peer communication protocols.

References

- [1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, May 2001.
- [2] J. Bernabeu-Auban, M. Ammad, and A. Ammar. Resource Finding in Store and Forward Networks. *Acta Informatica*, 28, 1991.
- [3] J. Byers, M. Luby, and M. Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.
- [4] Y. Chawathe, S. McCanne, and E. Brewer. RMX: Reliable Multicast for Heterogeneous Networks. In *Proceedings of IEEE INFOCOM'00*, Tel-Aviv, Israel, March 2000.
- [5] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.
- [6] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM SIGMETRICS'00*, Santa Clara, CA, May 2000.
- [7] S. Deering and D. Cheriton. Multicasting routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.
- [8] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, 1987.
- [9] Digital Island Coporation. <http://www.digitalisland.net/>.
- [10] The Gnutella Protocol Specification v0.4, revision 1.2. Available from <http://gnutella.wego.com>.
- [11] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of USENIX*, San Diego, CA, October 2000.
- [12] R. Karp, S. Shenker, C. Schindelhauer, and B. Vocking. Randomized rumor spreading. In *41st Symposium on Foundation on Computer Science (FOCS'00)*, Redondo Beach, CA, November 2000.
- [13] D. Katabi and J. Wroclawski. A framework for scalable global ip-anycast (gia). In *Proceedings of ACM SIGCOMM'00*, Stockholm, Sweden, September 2000.
- [14] D. Kempe, J. Kleinberg, and A. Demers. Spatial Gossip and Resource Location Protocols. In *Proceedings of the Thirty Third Annual ACM Symposium on Theory of Computing (STOC)*, Crete, Greece, July 2001.
- [15] T. Leighton. The Challenges of Delivering Content Over the Interent, June 2001. Keynote Address given at ACM SIGMETRICS 2001 Conference.

- [16] Matrix.Net. Internet withstands attack on america. Available at http://www.matrix.net/company/news/20010911_internet_withstands_attack_on_america.html.
- [17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.
- [18] M. Ripeanu and I. Foster. Peer-to-Peer Architecture Case Study: Gnutella Network. Technical report, University of Chicago, TR-2001-26, July 2001.
- [19] J. Ritter. Why Gnutella Can't Scale. No, Really, February 2001. Available from <http://www.monkey.org/~dugsong/mirror/gnutella.html>.
- [20] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. Technical report, University of Washington, UW-CSE-01-06-02, July 2001.
- [21] S. Savage, Andy Collins, Eric Hoffman, John Snell, and Tom Anderson. The End-to-End Effects of Internet Path Selection. In *Proceedings of SIGCOMM'99*, Cambridge, MA, September 1999.
- [22] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.
- [23] E. Zegura, M. Ammar, Z. Fei, and S. Bhattacharjee. Application-Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service. *Transactions on Networking*, 8(4), August 2000.

Appendix

A Proofs of Lemmas in Upper Bound Computation

Here, we prove the result that justifies Equation (11)

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 \mathcal{S}_j^k , $0 \leq j < kt, 0 \leq k \leq N_d$ 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,

$$E[\theta_j(\nu)] = \sum_{k=0}^{N_d} \sum_{\ell=0}^{t-1} \sum_{i=1}^{\infty} \left(E[S_\ell^k T_{i,\nu}^{\ell,k} \left(\prod_{j=1}^{i-1} \prod_{m=1}^{N_d} G_{j,\nu}^{\ell,m} \right) \prod_{m=1}^{k-1} G_{i,\nu}^{\ell,m} \right] \quad (18)$$

$$= \sum_{k=0}^{N_d} \sum_{\ell=0}^{t-1} \sum_{i=1}^{\infty} \left(E[S_\ell^k] E[T_{i,\nu}^{\ell,k}] \left(\prod_{j=1}^{i-1} \prod_{m=1}^{N_d} E[G_{j,\nu}^{\ell,m}] \right) \prod_{m=1}^{k-1} E[G_{i,\nu}^{\ell,m}] \right) \quad (19)$$

$$(20)$$

By substituting $E[S_\ell^k] = 1/(N_d t)$, that $\sum_{k=0}^{N_d} \sum_{\ell=0}^{t-1} E[T_{i,\nu}^{\ell,k}] = t E[T_{i,\nu}]$, that $E[G_{j,\nu}^{\ell,m}] = E[G_{j,\nu}]$, 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 Equation (11). ■

Next, we prove the result that $E[\theta_j(\nu)]$ is non-decreasing with increasing ν .

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 produce a 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 ($\langle n_{1,1}, n_{1,2} \rangle, \langle n_{2,1}, n_{2,2} \rangle, \dots, \langle n_{\ell,1}, n_{\ell,2} \rangle$) 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 ($\langle \phi(n_{1,1}), \phi(n_{1,2}) \rangle, \langle \phi(n_{2,1}), \phi(n_{2,2}) \rangle, \dots, \langle \phi(n_{\ell,1}), \phi(n_{\ell,2}) \rangle$) 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 . ■

Proof of Lemma 2: 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 $S = \{t_0, \dots, t_A\}$ 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 $R = \{r_1, \dots, r_n\}$ 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 , $1/(A - j') > 1/(A - j)$. ■

B How Valid are Analyses on a Fully Connected Graph?

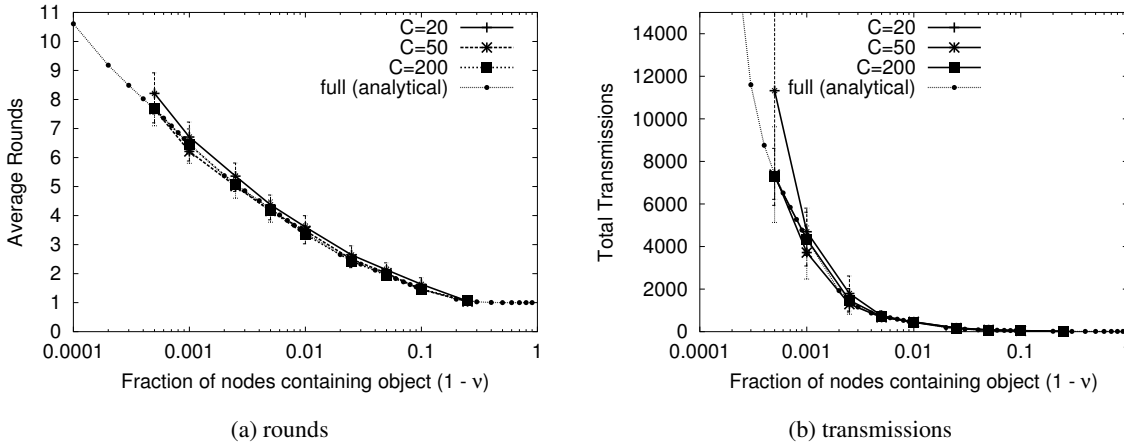


Figure 4: Fully connected overlay versus values for C

Here, we consider the effects of limiting C , the number of neighbors of a node while using the shuffling algorithm described in Section 4. Figure 4 presents the expected number of rounds (4(a)) experienced by each node and expected number of transmissions (4(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, $1 - v$ 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.