Reducing Communication Costs in Robust Peer-to-peer Networks

Jared Saia* Maxwell Young*

Abstract

Several recent research results describe how to design Distributed Hash Tables (DHTs) that are robust to adversarial attack via Byzantine faults. Unfortunately, all of these results require a significant blowup in communication costs over standard DHTs. For example, to perform a lookup operation, all such robust DHTs of which we are aware require sending $O(\log^3 n)$ messages while standard DHTs require sending only $O(\log n)$, where n is the number of nodes in the network. In this paper, we describe protocols to reduce the communication costs of all such robust DHTs. In particular, we give a protocol to reduce the number of messages sent to perform a lookup operation from $O(\log^3 n)$ to $O(\log^2 n)$ in expectation. Moreover, we also give a protocol for sending a large (i.e. containing $\Omega(\log^4 n)$ bits) message securely through a robust DHT that requires, in expectation, only a *constant* blowup in the total number of bits sent compared with performing the same operation in a standard DHT. This is an improvement over the $O(\log^2 n)$ bit blowup that is required to perform such an operation in all current robust DHTs. Both of our protocols are robust against an *adaptive* adversary.

Keywords: algorithms; distributed computing; fault-tolerance; peer-to-peer; adaptive adversary.

1 Introduction and Related Work

A distributed hash table (DHT) is a structured peer-to-peer (p2p) network which provides for scalable and distributed storage and lookup of data items (see e.g. [11, 9, 12]). Because peer-to-peer networks have little to no admission control, there has been significant effort in designing DHT's which are robust to *Byzantine* faults. When a peer suffers a Byzantine fault it is assumed to be controlled by an adversary who uses that peer to try to disrupt the network. We refer to the peers that suffer Byzantine faults as *bad* and the remaining peers as *good*.

Several recent results describe DHTs that are robust to adversarial attack via Byzantine faults [3, 5, 7, 4, 1]. The current state of the art in this area is a recent, striking result due to Awerbuch and Scheideler that describes a DHT that remains robust even if a number of join and leave events by Byzantine peers is polynomial in the size of the network n [1]. All of these past results make use of *quorums*, which are sets of $\Theta(\log n)$ peers with the property that no more than a fixed constant fraction of the peers in a quorum have suffered Byzantine faults.

While each of these results creates and maintains quorums in a different manner, the results are similar in the way in which they use quorums to perform lookup operations. In particular, assume that a peer p initiates the lookup operation and that there is a sequence of l quorums, Q_1, Q_2, ..., Q_l, that the lookup request must pass through. The request is first forwarded from p to Q_1. Next, for all i from 1 to $l-1$, every peer in quorum Q_i forwards the request to every peer in Q_{i+1}. Peers in Q_{i+1} accept only those messages that are received from a majority of peers in Q_i. Finally, the requested data item is retrieved by quorum Q_l and the data item is sent back in the same manner from Q_l to Q_1 and then to p. Provided that a majority of peers in each quorum are

*Department of Computer Science, University of New Mexico, NM, USA; email: saia, young@cs.unm.edu. This research was partially supported by NSF grant CCR-0313160 and Sandia University Research Program grant No. 191445.

not controlled by the adversary, this simple procedure ensures robust lookup operations. However, this technique is not bandwidth efficient. In particular, if $l = O(\log n)$, as is common, then the procedure requires $O(\log^3 n)$ messages. Moreover, if a data item containing b bits is sent using this procedure, then the total number of bits sent is $O(bl \cdot \log^2 n)$.

Our Contributions: In this paper, we give two algorithms for efficient communication through quorums in a robust DHT. First, we give an algorithm to reduce the communication for lookups from $O(\log^3 n)$ messages to $O(\log^2 n)$ in expectation. Second, we give an algorithm for sending a large (i.e. containing $\Omega(\log^4 n)$ bits) message securely that requires, in expectation, only a *constant* blowup in the total number of bits sent compared with performing the same operation in a standard DHT. This is an improvement over the $O(\log^2 n)$ bit blowup that is required to perform such an operation in all previous quorum-based robust DHTs. Many first generation peer-to-peer systems sent large data items directly from the peer holding the data item to the peer requesting the item. However, increasingly in modern p2p systems such as BitTorrent, very large data items are being sent through the network to provide both 1) better download times for the receiver of the data item (due to increased bandwidth utilization); and 2) improved anonymity to the holder of the data item. To the best of our knowledge, our algorithm is the first with guaranteed robustness against a Byzantine adversary that will reduce bit blowup for modern p2p systems such as BitTorrent.

Each peer is assumed to have a unique ID (e.g. the IP address of the peer) and that peers in neighboring quorums know each others IDs. For simplicity, we will frequently use the same variable to refer both to the peer and to the ID of the peer. We will call peers controlled by the adversary *bad* and will call the remaining peers *good*. For both results, we assume that strictly less than a $1/4$ fraction of the peers in every quorum are bad. We also assume that each quorum is of size $C_1 \ln n$ where C_1 is a positive constant[1]. A primary focus of all of the robust DHT results mentioned above is to describe protocols to maintain these types of invariants for the quorums. Thus, in this paper, we assume that these invariants are true and focus instead only on enabling efficient and secure communication.

Given these invariants, our protocols ensure correctness, with high probability[2], over a polynomial number of peer join and leave events. Our first protocol is robust against a computationally unbounded adversary, while our second protocol is robust only against a polynomial-time bounded adversary. Our protocols are fully integrable with the p2p networks described in [1, 3, 5, 7, 4]. In addition, our protocols can be used to create more efficient protocols, not just looking up data items, but also for other p2p operations such as allowing a peer to join the network or maintaining link structure.

Finally, we believe our protocols may be applicable to reducing communication costs in robust radio networks. A common fault model for such networks is to assume that no more than a small, constant fraction of the nodes in any neighborhood of a certain size suffer Byzantine faults [**?**, **?**, **?**, **?**]. We believe we can treat each neighborhood of the radio network as a quorum and thereby reduce the communication costs of current algorithms for communicating robustly in radio networks. Of course, these new reduced-cost protocols will succeed only with high probability instead of with probability 1.

A preliminary version of the results in this paper appeared in [4].

2 Expected $\Theta(\log^2 n)$ Messages

In this section, we show how to improve message passing between quorums so that only $\Theta(\log n)$ messages are sent in expectation from a quorum L to a quorum R. To avoid the problem of the bad peers in L flooding the good peers in R, there must be some systematic way to match up each good peer $r \in R$ to those peers in L that should legitimately be sending to r. A naive approach for doing this would be to order peers in L and R by distance from the start of the quorum and then match

[1] We note that it is easy to generalize our results to the case that not all quorums are of the same size, but rather that each quorum is of size between $C_1 \ln n$ and $C_2 \ln n$ for fixed and known constants C_1 and C_2.

[2] We use the phrase *with high probability* throughout this paper to mean with probability $1 - 1/n^k$ for any desired constant $k > 0$.

Algorithm 1: Sending from L to R

1: Each peer $\ell \in L$ sends a message to peer $r \in R$ if $h_1(\ell) = h_2(r)$

2: Each peer $r \in R$ accepts a message from peer $\ell \in L$ if $h_1(\ell) = h_2(r)$

3: Each peer $r \in R$, does majority filtering on all the messages accepted from peers in L to decide on which message, if any, to commit.

up senders and receivers according to this ordering. Unfortunately, this approach fails since each peer may have a slightly different view of the set of peers in L and R. In particular, if some peer x does not know about even a single peer in L, then x's ordering of the peers in L may be completely incorrect.

To remedy this problem, we make use of two functions to match up senders and receivers. *We assume that the adversary has full knowledge of these functions before the protocol starts.* Thus, the naive approach of matching up senders in L and receivers in R according to the output of a single function will fail. Instead, we match up senders and receivers via a more clever two-stage technique. In particular, we assume all peers have functions h_1 and h_2 that map peers to the positive integers. The function h_1 maps a peer uniformly to an integer in the range 1 to $\ln n$, while h_2 maps a peer to C_2 integers selected uniformly and independently in the range 1 to $\ln n$, for some constant C_2 to be specified later. For ease of presentation, we assume the existence of functions h_1 and h_2 for the following analysis; we will show how such functions can be constructed in Section 4.

We first say that each peer $\ell \in L$ is assigned to a *bin* between 1 and $\ln n$ whose number is given by $h_1(\ell)$. *The bad peers in L can choose the bin they are assigned to and thus the bad peers get to make their bin selections after seeing the bins chosen by all the good peers.* However, we can still show that with high probability, at least a 6/10 fraction of the bins will be *good* in that they will contain a majority of good peers. Next, each good peer $r \in R$ is assigned to C_2 bins using $h_2(r)$. Finally, each peer $\ell \in L$ sends to every peer $r \in R$ that is assigned to the same bin as ℓ. Then each peer $r \in R$ first does majority filtering over each bin it was assigned to in order to get a message value for that bin. Next, r does majority filtering over the message values from each of the C_2 bins in order to decide on a single message value.

We then are able to show that no matter what set of $(4/10)\ln n$ of the bins are bad, that for C_2 chosen sufficiently large, almost all of the good peers in R will be assigned to a majority of good bins and will thus decide on the correct message. To show this requires a somewhat subtle proof that makes use of the probabilistic method since there are many possibilities for which bins may be bad. Our algorithm for reducing message cost when sending from quorum L to quorum R, is given in Algorithm 1. The proof of correctness of our lookup procedure is given as Theorem 1.

Lemma 1. *The expected number of messages sent by good peers in Algorithm 1 is* $\Theta(\log n)$

Proof. There are $C_1 \ln n$ peers in R and each of them maps to C_2 bins. Thus a bin chosen uniformly at random will have C_1C_2 peers in R that map to it. Each good peer in L maps to a bin chosen uniformly at random so the expected number of messages sent by any good peer is C_1C_2. By linearity of expectation, the expected number of messages sent by all good peers is $C_1C_2 \log n = \Theta(\ln n)$. □

Recall that we say that a bin is good if among the peers in L in that bin, a majority are good and have the correct message. The next lemma lower-bounds the number of good bins.

Lemma 2. *If at least* 74/100 *fraction of the peers in L good and have the correct message then for* C_1 *chosen sufficiently large, with high probability, at least a* 6/10 *fraction of the bins will be good.*

Proof. We will say that a bin is *full* if it has at least $(7/10)C_1$ good peers in it (and any number of bad peers). We first show that for any $\epsilon > 0$, we can choose C_1 such that, with high probability, all but $\epsilon \log n$ bins will be full. To see this, let B' be some set of $\epsilon \log n$ bins and let $N(B')$ be the number of good peers assigned to B'. Further let $f = 74/100$ be the fraction of good peers that have the correct message. Note that the expected number of good peers assigned to B' is exactly

$fC_1\epsilon \ln n$. Further note that each good peer is assigned independently to B'. Thus, by Chernoff bounds, we can say that for any $0 \leq \delta \leq 1$:

$$Pr(N(B') \leq (1-\delta)f\epsilon C_1 \epsilon \ln n) \quad \leq \quad e^{-(1/2)\delta^2 f\epsilon C_1 \ln n}$$

We are particularly interested in the case where $\delta = 4/74$ which ensures that $(1-\delta)f = 7/10$. Let ξ be the event that *any* set of $\epsilon \log n$ bins have less than $(1-\delta)fC_1\epsilon \ln n$ good peers in them. Then we can say that:

$$\begin{aligned} Pr(\xi) &\leq \binom{\ln n}{\epsilon \ln n} e^{-(1/2)\delta^2 f\epsilon C_1 \ln n} \\ &\leq e^{(1-(1/2)\delta^2 f\epsilon C_1)\ln n} \end{aligned}$$

For any $k > 0$, any $\delta > 0$, any $0 < \epsilon < 1$ and C_1 chosen greater than $\frac{2(1+k)}{f\epsilon\delta^2}$ this last probability will be no more than n^{-k}. Choosing, $\epsilon = 1/100$ and $\delta = 4/74$, we can say that w.h.p., no more than a $1/100$ fraction of the bins have less than $(7/10)C_1$ good peers in them.

We still must show that the adversary cannot place the bad peers in such a way as to take over too many bins. To make a full bin bad, the adversary must place at least $(7/10)C_1$ bad peers in the bin. The adversary has $(1/4)C_1 \ln n$ bad peers to place so it can take over no more than $\frac{(1/4)C_1 \ln n}{(7/10)C_1} = (5/14)C_1 \ln n$ of the full bins. Even assuming that the adversary also takes over all of the bins that are not full, the number of bad bins will be no more than $(5/14 + 1/100)C_1 \ln n < (4/10)C_1 \ln n$. Thus, the fraction of good bins will be at least 6/10. □

Lemma 3. *If at least a* $6/10$ *fraction of the bins are good then for* C_2 *chosen sufficiently large, with high probability, at least* $(74/100)C_1 \ln n$ *peers in* R *will be good and will commit to the correct message.*

Proof. We must show that the number of good peers in R that do not map to a majority of good bins is no more than $(1/100)C_1 \ln n$. Consider some set R' of good peers in R where $|R'| = (1/100)C_1 \ln n$. Let $N(R')$ be the number of good bins that peers in R' map to. Each peer in R' maps to C_2 bins so $E(N(R')) \geq C_2(6/10)(1/100)C_1 \ln n$. By Chernoff bounds, we can say that for any $0 \leq \delta \leq 1$:

$$Pr(N(R') \leq (1-\delta)E(N(R'))) \quad \leq \quad e^{-\delta^2 C_2(3/1000)C_1 \ln n}$$

We are particularly interested in the case where $\delta = 1/6$ which ensures that $(1-\delta)(6/10) = 1/2$. Let ξ be the event that *any* set of $(1/100)\log n$ good peers in R all do not map to a majority of good bins.Then we can say that:

$$\begin{aligned} Pr(\xi) &\leq \binom{C_1 \ln n}{(1/100)C_1 \ln n} e^{-\delta^2 C_2(3/1000)C_1 \ln n} \\ &\leq e^{(1-\delta^2 C_2(3/1000))C_1 \ln n} \end{aligned}$$

Choosing $C_2 > \frac{1000}{3\delta^2}$ ensures that the lemma holds with high probability. □

The following corollary is immediate from Lemmas 2 and 3.

Corollary 1. *For* C_1 *and* C_2 *chosen sufficiently large, with high probability, if at least a* $74/100$ *fraction of the peers in* L *are good and have the correct message then at the end of Algorithm 1, at least a* $74/100$ *fraction of the peers in* R *will be good and have the correct message.*

Let LOOKUP denote a lookup protocol consisting of communication through a sequence of $O(\log n)$ quorums.

Theorem 1. *For* C_1, C_2 *sufficiently large but depending only on* k*, the following is true with probability at least* $1 - 1/n^k$*:*

- *All calls to* LOOKUP *succeed.*

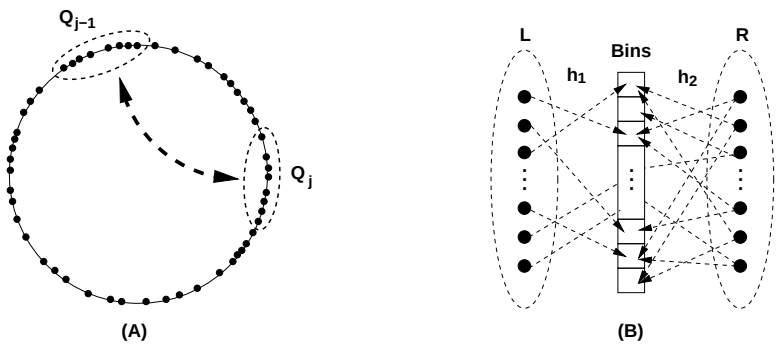

Figure 1: (A) Two quorums Q_{j-1} and Q_j on the unit circle in a DHT. The bidirectional dashed arrow implies that all peers in Q_{j-1} know the IDs of all peers in Q_j and vice-versa. (B) In the context of Algorithm 1, the quorums L and R are depicted. Each peer $\ell \in L$ is mapped to a single bin via h_1 and each peer $r \in R$ is mapped to C_2 bins via h_2 where in this simple depiction $C_2 = 2$.

- *All calls to* LOOKUP *send* $\Theta(\log^2 n)$ *messages in expectation.*

Proof. Consider some arbitrary call to the lookup protocol described by the sequence: $p, Q_1, Q_2, Q_3, \ldots, Q_l$ where p belongs to Q_1 and m is the request sent. Here p sends the message m to quorum Q_1, Q_1 sends the request to Q_2 via Algorithm 1, Q_2 sends the request to Q_3 via Algorithm 1 and so on. For a quorum Q, let $G(Q)$ be the set of good peers in Q that commit to m after majority filtering.

By a Union bound and repeated applications of Corollary 1, we can say that w.h.p., for any Q_i, $G(Q_i) \geq 74/100 C_1 \ln n$. Thus $(74/100)|Q_l|$ good peers in Q_ℓ receive the request m and fetch the appropriate data item, d. A similar argument from Q_ℓ, $Q_{\ell-1}$ and all the way back to Q_1 shows that p will receive the correct data item d after majority filtering on messages received from Q_1. Finally, note that since $l = O(\log n)$, by Lemma 2, the expected number of messages sent is $\Theta(\log^2 n)$.

□

3 Expected $\Theta(1)$ Bit Blowup

In this section we assume that peer p is trying to transmit a message m along the path $p, Q_1, Q_2, Q_3, \ldots, Q_l$ where Q_i, $i = 1, \ldots, l$ are quorums and p belongs to Q_1. We assume that the adversary is polynomially bounded. Let $|m|$ be the number of bits in the message m. Peer p first encodes m into $\log n$ pieces $e_1, e_1, \ldots, e_{\ln}$ with the following properties: 1) each piece has $O(|m|/\log n)$ bits and 2) m can be reconstructed from any 1/16-th fraction of the pieces. Any standard erasure code, such as tornado codes [6], can be used to create pieces with these two properties.

Peer p next creates fingerprints $f_1, f_2, \ldots, f_{\ln n}$ of all these pieces using a 1-way hash function, h_f known by all the peers. For all $i = 1, \ldots, \ln n$, $f_i = h_f(e_i)$. Each of fingerprint has $\log^2 n$ bits. This ensures that the probability that a random string maps to a fixed fingerprint is $1/n^{\ln n}$; therefore, the adversary requires superpolynomial time to find a string which maps to one of the fingerprints. We also make use of the function h_1 from the previous section that maps peers to integers in the range 1 to $\ln n$.

Peer p sends all of the fingerprints to all of the peers in Q_1. Then for all $j = 2, \ldots, l$, all peers in Q_{j-1} send all of the fingerprints to all the peers in Q_j and peers in Q_j accept a fingerprint if and only if it was received from a majority of peers in Q_{j-1}. This guarantees that all peers in the quorums $Q_1, Q_2, \ldots, Q_l$ know all of the fingerprints. We will thus assume, in the protocol, described in this section, that a peer accepts a string s as some piece e_j if and only if $h_f(s) = f_j$.

The first step of our protocol is simple. Peer p sends piece e_i to peer $x \in Q_1$ iff $h_1(x) = i$. The general protocol where Q_{j-1} sends to Q_j for all $j = 2, \ldots, l$ is given as Algorithm 2. This protocol

Algorithm 2: Sending from Q_{j-1} to Q_j

1: Each peer in Q_{j-1} sends the fingerprints of the message m, $f_1, f_2, \ldots, f_{\ln n}$, to each peer in Q_j.
2: The peers accept only those fingerprints that they receive from a majority of the peers in Q_{j-1}. In the remainder of the algorithm, a peer in Q_j only accepts a string s as some piece e_j if $h_f(s) = f_j$.
3: **while** TRUE **do**
4: Peers in Q_{j-1} come to consensus on a random integer r in $\{1, \ldots \ln n\}$.
5: Let $P = \{x \in Q_j | h_1(x) = r\}$. All peers in Q_{j-1} send all of the pieces they currently hold to all peers in the set P.
6: All peers in P reconstruct the message m from the pieces received. From the message m, they then recompute the pieces $e_1, e_1, \ldots, e_{\ln n}$.
7: For each $i = 1, \ldots, \ln n$, all peers in P send piece e_i to all peers $x \in Q_j$ such that $h_1(x) = i$.
8: The peers in Q_j come to consensus about whether they want a resend as follows:

1. Each peer $x \in Q_j$ does the following. If $h_1(x) = i$ and x received piece e_i, x writes all peers in Q_j that it received its piece.

2. Every peer $x \in Q_j$ does the following. If x received messages indicating that at least $(3/4)|Q_j|$ peers received their pieces, it tentatively sets an individual "resend" bit to 0 otherwise it sets this bit to 1.

3. All peers in Q_j do Byzantine agreement to come to consensus on the "resend" bit values set in the previous step.

9: The peers in Q_j send to the peers in Q_{j-1} the results of this consensus i.e. either that they want a resend or that they do not want a resend.
10: If the peers in Q_{j-1} receive responses from more than $1/4$ of the peers in Q_j that they do not want a resend, the algorithm terminates.

makes use of a Byzantine agreement protocol and the protocol for coming to consensus on a random number. For the latter, we employ the scheme proposed in [1]. The basic idea of the remaining steps of the protocol i.e. sending from Q_{j-1} to Q_j is as follows. The peers in Q_{j-1} agree on a randomly selected leader in Q_j to whom they will send all their pieces. This leader, if good, will reconstruct the message m; reconstruct the pieces of m; and then send out the appropriate piece to each peer in Q_j. Each peer in Q_j checks the piece they received against the fingerprint for that piece. The peers in Q_j then run an agreement procedure to determine if enough of them received the correct piece. If so, the step ends. If not, this information is sent back to Q_{j-1} and a new random leader is selected. For any quorum Q, we will now let $G(Q) = \{x \in Q |\ h_1(x) = i$ and x is good and has piece $e_i\}$.

Lemma 4. *For any fixed k, for C sufficiently large but depending only on k, the following is true with probability at least $1 - 1/n^k$. For all quorums Q, if $|G(Q)| \geq |Q|/2$ and all peers in Q send their pieces to some peer x, then x will be able to reconstruct the message m.*

Proof. We first fix a quorum Q and calculate the probability that the statement of the lemma is not true. For any set of good peers, X, let $U(X) = \{i | h_1(x) = i, \text{for some peer } x \in X\}$. Let Q' be some fixed subset of good peers in Q such that $|Q'| \geq |Q|/2$ and all peers in Q' have their correct pieces. Let P' be some fixed subset of the set of $\ln n$ pieces, such that $|P'| > (15/16) \ln n$. Let $\xi(Q', P')$ be the probability that no peer in Q' has a piece in P'. This is equivalent to a balls and bins problem where there are $|Q'|$ balls and $\ln n$ bins and we are asking the probability that none of the $|Q'|$ balls fall in a fixed set of $|P'|$ of the bins. Thus:

$$Pr(\xi(Q', P')) \ \leq \ (1/16)^{|Q'|} \ \leq \ (1/2)^{2|Q|}$$

Now let $\xi(Q)$ be the event that for any subsets Q' and P', the event $\xi(Q', P')$ occurs. Then we have:

$$\begin{aligned} Pr(\xi(Q)) &= Pr(\bigcup_{Q',P'} \xi(Q', P')) \leq \sum_{Q',P'} Pr(\xi(Q', P')) = \binom{|Q|}{|Q|/2}\binom{\ln n}{(15/16)\ln n}(1/2)^{2|Q|} \\ &\leq 2^{|Q|}2^{\ln n}((1/2)^{2|Q|}) \leq (1/2)^{|Q|-\ln n} \leq 1/n^{C-1}. \end{aligned}$$

Where the last inequality follows since $|Q| \geq C_1 \ln n$. A simple union bound over all n of the swarms gives that the probability that the statement in the lemma fails for *any* quorum is no more than $1/n^{C-2}$. Choosing C sufficiently large makes this probability no more than $1/n^k$ for any k. □

We will refer to one iteration of the loop in Algorithm 2 as a round.

Lemma 5. *For any fixed k, for C sufficiently large but depending only on k, the following is true with probability at least $1 - 1/n^k$ for all pairs of quorums, Q_{j-1} and Q_j. If $|G(Q_{j-1})| \geq (1/2)|Q_{j-1}|$ before Algorithm 2 starts, then $|G(Q_{j-1})| \geq (1/2)|Q_j|$ after termination. Further, if all peers in Q_{j-1} know all of the fingerprints before Algorithm 2 starts, then all peers in Q_j will know all the fingerprints after termination. Algorithm 2 will:*

- *Terminate in $O(1)$ rounds in expectation;*
- *Require good peers to send $O(\log^3 n)$ messages in expectation;*
- *Require good peers to send $O(|m| + \log^4 n)$ bits to be sent in expectation.*

Proof. Since quorum Q_{j-1} is good and the peers in Q_j do majority filtering in Step 2, we know that if all peers in Q_{j-1} know all of the fingerprints before Algorithm 2 starts, then all peers in Q_j will know all the fingerprints after termination.

If $|G(Q_{j-1})| \geq (1/2)|Q_{j-1}|$, then by Lemma 4, all peers in the set P which are sent the message pieces in Step 5 will be able to reconstruct the message m. Since 3/4 of the peers in Q_j are good, the probability that no peer in P is good is no more than $(1 - 1/\ln n)^{(3/4)C_1 \ln n} \leq e^{-(3/4)C_1}$. Thus with constant probability, some peer in the set P is good. If this is the case, then *all* peers in Q_j will receive their correct piece of the message. This implies that the algorithm will terminate in that round with $|G(Q_j)| \geq 1/2|Q_j|$. This implies that Algorithm 2 will terminate in an expected constant number of rounds.

We next establish correctness. Consider the situation where no peer in P is good. There are then two possible cases. First is the case less than $(1/2)|Q_j|$ peers in Q_j are sent their pieces in Step 5. In this case, no peer in Step 8.2 will receive at least $(3/4)|Q_j|$ messages saying that pieces were received. Thus all good peers will set their resend bits to 1 in this step and so the consensus will be to request a resend. This implies that the algorithm will continue for another round. The second case is that faulty peers in P send pieces to at least $(1/2)|Q_j|$ peers in Q_j. If this is the case, then it's safe for the algorithm to terminate.

We now compute the resource costs. Previous to the first round, the fingerprints are sent which requires $O(\log^2 n)$ messages, and $O(\log^4 n)$ bits. The expected size of the set P is $O(1)$. We employ techniques from secure multiparty computation in order to have quorums select a random value in $[0, 1)$. There are several results showing how to achieve secure multiparty computation in an asynchronous network provided that the fraction of Byzantine players is strictly less than 1/4 (see [2, 10, 8]). Srinathan and Rangan [10] give the most resource efficient secure multiparty computation protocol of which we are aware for this problem. In the case where there are $\Theta(\log n)$ peers, strictly less than 1/4 of which are faulty, we can compute a random number using $\Theta(\log^3 n)$ messages, with $\Theta(\log n)$ latency using their protocol. This protocol can be used to complete Step 3 in Algorithm 2. Therefore, in each round of the algorithm, the total number of messages sent is $O(\log^3 n)$ and the expected total number of bits sent is $O(|m| + \log^4 n)$. The expected resource costs of the entire algorithm then follow directly from the fact that there are $O(1)$ rounds in expectation. □

Lemma 6. *For any fixed k, for C sufficiently large but depending only on k, the following is true with probability at least $1 - 1/n^k$ if we run Algorithm 2:*

- *All calls to* LOOKUP *succeed.*
- *All calls to* LOOKUP *have latency* $O(\log n)$ *in expectation.*
- *All calls to* LOOKUP *require* $O(\log^4 n)$ *messages to be sent in expectation.*
- *All calls to* LOOKUP *require* $O(|m| \log n + \log^5 n)$ *bits to be sent in expectation.*

Proof. Consider a message m which is sent along a path described by the sequence: $p, Q_1, Q_2, Q_3, \ldots, Q_l, q$, where p belongs to Q_1 and q belongs to Q_l. Here peer p sends all pieces of m and fingerprints of these pieces to $Q_1 = Q(p)$, Q_2 sends the pieces and fingerprints to Q_2 via Algorithm 2, Q_2 sends the pieces and fingerprints to Q_3 via Algorithm 2 and so on until finally all peers in quorum Q_l sends all their pieces and fingerprints to peer q. Since, $|G(Q_1)| \geq 1/2|Q_1|$, Lemma 5 and induction give that $|G(Q_l)| \geq 1/2|Q_j|$ and that all peers in Q_l have all the fingerprints of these pieces. This implies that when peers in Q_l send their pieces and fingerprints to q, q will have enough information to reconstruct the message m. The latency, message complexity, and bit complexity follow immediately from Lemma 5 and the fact that $l \in O(\log n)$. □

4 Function Construction and Estimates of $\ln n$

In this section, we address the construction of functions h_1 and h_2 that are necessary for Algorithms 1 and 2. We then briefly address correctness for both Algorithm 1 and Algorithm 2 when $\ln n$ is not known precisely and estimates must be used.

4.1 Constructing Functions h_1 and h_2

In order for Algorithm 1 to be correct, the following must be true. First, the good peers must be mapped by h_1 and h_2 to bins uniformly at random in the appropriate ranges. Second, the assignment of all peers to bins must be known unambiguously to all good peers in L and R. Consider the function h_1. At first glance, a commonly known hash function that maps peer IDs to the non-negative integers modulo $\ln n$ would appear to suffice. Unfortunately, such a setup, while satisfying the second criteria, may fail to satisfy the first. That the adversary knows exactly how h_1 behaves is not problematic; we always assume the adversary can place the bad peers into whichever bin(s) it pleases. Rather, the problem is that the adversary can assign its peers IDs that all map to a specific set of bins B'. Since IDs are of fixed length, there are a finite number of IDs that map to bins in B' and, consequently, fewer IDs that map to B' will be available to good peers; this bias ruins the assumption that good peers are mapped to bins uniformly at random.

Fortunately, there is a simple solution. Upon joining the network at quorum L, a peer p generates a bin number b_p uniformly at random in $\{1, ..., \ln n\}$ using an internal random number generator. Bad peers may select their own bin number. Peer p tells all peers in L the value b_p. In order to prevent any ambiguity that might arise from a bad peer telling different peers in L different values for b_p, all peers in L come to consensus on the value received using a standard Byzantine agreement protocol. The peers in R are subsequently informed via all-to-all communication between R and L. In this way, peers in L and R know a peer's bin number unambiguously whenever it joins the network. The solution for constructing h_2 is essentially identical.

4.2 Using Estimates of $\ln n$

Since peers in a DHT are not likely to know the exact value of $\ln n$, we briefly discuss how to modify our algorithms to handle different estimates of $\ln n$. It is shown in [4] that for any two peers p and p', (lower) estimates l_p, $l_{p'}$ and (upper) estimates $u_p, u_{p'}$ can be obtained such that:

$$\ln n \leq l_p \leq u_{p'} \leq \ln n + C_0 \text{ and } \ln n \leq l_{p'} \leq u_p \leq \ln n + C_0$$

for fixed constant C_0. We can then view h_1 and h_2 as mapping the identifier space to the non-negative integers and, therefore, we can then change the first two bullets of Algorithm 1 as follows:

- Each peer $\ell \in L$ sends a message to peer $r \in R$ if $h_1(\ell) = h_2(r) \mod l_\ell$
- Each peer $r \in R$ accepts a message from peer $\ell \in L$ if $h_1(\ell) = h_2(r) \mod u_r$

Showing the modified algorithm is correct requires only minor modifications of the proof of Lemma 2. Let B again be the set of bins from 1 to $\ln n$. It is straightforward to show that, with high probability, an arbitrarily small fraction of the bins in B are "bad". Then, with high probability, an arbitrarily large fraction of the peers in Q_j fall in bins in B which are good.

The modifications to the Algorithm 2 are as follows. The peer p which starts sending the message m encodes m into l_p pieces and creates fingerprints for all of these pieces. When it sends the fingerprints to the quorum Q_{j-1}, it also sends the number l_p to all peers in Q_j. The peers in Q_{j-1} then use this number l_p as their estimate of $\ln n$. When the peers in Q_{j-1} send the fingerprints to peers in Q_j, all peers in Q_{j-1} also send the number l_p to all peers in Q_j. In this way, we maintain the invariant that all peers in the quorums that m is sent to know and use the estimate, l_p, calculated by p. We finally note that both algorithms are robust to minor inconsistencies in the views the peers have as to which peers are in the quorums Q_{j-1} and Q_j.

5 Conclusion

We have presented new algorithms for reducing communication costs in peer-to-peer networks that are robust against a dynamic adversary. Our algorithms are compatible with any type of p2p network that maintains quorums with strictly less than a 1/4 fraction of Byzantine nodes in each quorum. Further, our algorithms are fairly simple and therefore may be of use in practical p2p networks.

Many problems remain open including the following. First, we believe we can use the techniques in this paper to reduce communication costs in radio networks. However, several tricky problems still must be solved in order to apply our techniques to this new domain. Second, we would like to prove lower-bounds on communication costs and the adversarial power that we can tolerate. Are $O(log^2 n)$ messages necessary to perform a robust lookup? Can we tolerate more than a 1/4 fraction of bad peers in each quorum? Can we reduce the bit blowup to $O(1)$ even against a computationally unbounded adversary? Finally, we would like to demonstrate that these algorithms are practical through simulation and possible use in a deployed system.

Acknowledgements: We gratefully thank Amos Fiat for his help with this paper.

References

[1] Baruch Awerbuch and Christian Scheideler. Towards a scalable and robust DHT. In *18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2006.

[2] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth ACM Symposium on the Theory of Computing (STOC)*, 1993.

[3] Amos Fiat and Jared Saia. Censorship resistant peer-to-peer content addressable networks. In *Proceedings of the Thirteenth ACM Symposium on Discrete Algorithms (SODA)*, 2002.

[4] Amos Fiat, Jared Saia, and Maxwell Young. Making chord robust to byzantine attacks. In *Proceedings of the* 13^{th} *Annual European Symposium on Algorithms (ESA)*, pages 803–814, 2005.

[5] Kristen Hildrum and John Kubiatowicz. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proceedings of the 17th International Symposium on Distributed Computing*, 2004.

[6] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. Practical loss-resilient codes. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 150 – 159, 1997.

[7] Moni Naor and Udi Wieder. A simple fault tolerant distributed hash table. In *Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[8] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Asynchronous unconditionally secure computation: An efficiency improvement. In *INDOCRYPT 2002, Lecture Notes in Computer Science*, volume 2551, pages 93–107. Springer-Verlag, 2002.

[9] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[10] K. Srinathan and C. Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. In *INDOCRYPT 2000, Lecture Notes in Computer Science*, volume 1977, pages 117–129. Springer-Verlag, 2000.

[11] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[12] Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, University of California at Berkeley Technical Report, April 2001.