

# Approximating the number of differences between remote sets

*Sachin Agarwal*

Deutsche Telekom AG, Laboratories

Email: sachin.agarwal@telekom.de

*Ari Trachtenberg*

Boston University

Email: trachten@bu.edu

## Abstract

We consider the problem of approximating the number of differences between sets held on remote hosts using minimum communication. Efficient solutions to this problem are important for streamlining a variety of communication sensitive network applications, including data synchronization in mobile networks, gossip protocols and content delivery networks. Using tools from the field of interactive communication, we show that this problem requires about as much communication as the problem of exactly determining such differences. As a result, we propose a heuristic solution based on the counting Bloom filter. We provide analytic bounds on the expected performance of our protocol and also experimental evidence that they can outperform existing difference approximation techniques.

*A version of this work will appear at the IEEE Information Theory Workshop, Punta del Este, Uruguay, March 2006.*

## I. INTRODUCTION

Many distributed network systems maintain copies of the same information on different hosts of a network. In order to maintain even weak data consistency, hosts must periodically reconcile

This work was completed while Sachin Agarwal was at Boston University. This work is based, in part, upon work supported by the National Science Foundation under Grants CCR-0133521 and ANI-0240333.

their differences with other hosts as connections become available or according to prescribed scheduling. In a dense or constrained network the decision to reconcile should be based, in part, on the number of differences between the reconciling hosts' data sets. Although hosts with many differences between them should probably be fully reconciled, hosts that are fairly similar might wait for more differences to accumulate. Unfortunately, simple solutions, such as update timestamps, do not scale well to dynamic or large networks because of the need to maintain an update history with respect to every other host [1].

In this paper we introduce a new approach for approximating the number of differences between two remote data sets based on a variant of the counting Bloom filters. Formally, the problem is as follows: given two hosts  $A$  and  $B$  with data sets  $S_A$  and  $S_B$  respectively, we wish to approximate the size of mutual difference  $S_A \oplus S_B \hat{=} (S_A - S_B) \cup (S_B - S_A)$ . Our goal is to measure this size as accurately as possible and using as little communication as possible, measured both in terms of transmitted bytes and rounds of communication. As a secondary goal, we also seek to reduce the computational cost involved with such an approximation. We note that the accuracy of the approximations provided in this paper is statistical in nature (*i.e.*, an average accuracy over a number of approximations).

### A. Organization

In Section II we provide a baseline information-theoretic analysis of the difference approximation problem. Thereafter, we describe some existing protocols for difference approximation and very briefly review traditional Bloom filters. Our heuristic, a *wrapped filter* approximation based on the counting Bloom filter, is presented in Section III. The accuracy of this technique depends on the amount of probabilistic false positives incurred, which we analyze in Section IV. Finally, in Section V we experimentally compare our approach to existing approximation techniques. Conclusions and directions for future work are discussed in Section VI.

## II. PRELIMINARIES

We first provide an information-theoretic baseline of the amount of communication needed for difference approximation, and thereafter proceed to describe some existing solutions to the problem and some fundamental properties of the Bloom filter.

### A. Information-theoretic bounds

1) *Tools:* We shall rely on two tools in the analysis of the complexity of difference approximation.

The first is a well-known result of Yao [2], based on a deterministic two-way communication model in which two remote users with data  $X \in M$  and  $Y \in N$  respectively alternate, sending each other one bit at a time, with the goal of computing a given deterministic function  $f(X, Y)$ . Yao showed that at least  $\log_2(d(f)) - 2$  bits of communication are needed to correctly communicate  $f$ , with  $d(f)$  being the minimum number of monochromatic rectangles needed to partition  $f$  on  $M \times N$ .

The second tool comes from the work of Orlitsky and Roche [3], in which remote users have random variables  $X$  and  $Y$ . In this one-way communication model,  $X$  repeatedly sends (as one block) an instance of his variable to the second user, who then attempts to compute  $f(X, Y)$  with respect to instances of its own random variable, with vanishing block error probability. Orlitsky and Roche showed that the number of bits that must be transmitted per block for this model is given by the graph entropy [4]  $H_G(X|Y)$  of the characteristic graph [5] of function  $f$ <sup>1</sup>, namely  $H_G(X|Y) \hat{=} \min I(W; X|Y)$ , where  $I$  denotes mutual information, and  $W$  represents a random variable over the set of independent sets (*i.e.*, induced subgraphs with no edges) of  $G$ , with conditional probability  $\sum_{x \in w} p(w|x) = 1$  for instances  $w \in W$  and  $x \in X$ . The minimization is taken only over Markov chains  $W, X, Y$ .

2) *Results:* All the algorithms discussed in this paper compute the *approximate* number of differences between sets on remote hosts. Unfortunately, determining the *exact* number of set differences requires a large amount of communication, essentially at least as much information as contained between of the sets. This comes out from the fact that computing set equality is a special case of computing set differences [2, 3].

The following lemma shows that approximating differences within an *additive* constant also requires much communication.

**Lemma II.1** *For a fixed  $\mathcal{U}$ , no algorithm can deterministically and definitively compute an*

<sup>1</sup>This is a graph whose vertices are the support set of  $X$  and edges  $(x, x')$  are such that  $\exists y$  with  $p(x, y), p(x', y) > 0$  and  $f(x, y) \neq f(x', y)$ .

approximation  $\hat{\Delta}(S, S')$  of the number of differences  $\Delta(S, S')$  with

$$\Delta(S, S') - k \leq \hat{\Delta}(S, S') \leq \Delta(S, S') + k \quad \forall S, S' \subseteq \mathcal{U}.$$

using less than  $\Omega(|\mathcal{U}|)$  bits of communication at worst.

*Proof:* Consider the boolean function  $f(S, S')$  defined to be 1 exactly when  $\hat{\Delta}(S, S') \leq k$ . Clearly computing  $\hat{\Delta}$  requires at least as much communication as computing  $f$ . On the other hand, the number of ones in any row  $f(S, S') \forall S' \subseteq \mathcal{U}$  will, at most, consist of all sets that differ by  $\pm k$  elements from  $S$ ; there are  $O(|\mathcal{U}|^{2k})$  such sets. As such  $\frac{2^{|\mathcal{U}|}}{|\mathcal{U}|^{2k}}$  monochromatic rectangles are needed to partition the space of  $f$ , leading to the stated result under Yao's theorem [2]. In fact the result can be generalized to any approximation that results in a function  $f$  with asymptotically less than  $2^{|\mathcal{U}|}$  ones in any row. ■

We can deduce a stronger result under the model of Orlitsky and Roche, essentially showing that one cannot efficiently approximate set difference in many cases. This result is based on the following lemma generalizing a similar result in [6].

**Lemma II.2** *Let  $q_G(P)$  denote the probability that two vertices, chosen independently with distribution  $P$ , do not form an edge in  $G$ . Then*

$$H_G(X|Y) \geq \log \left( \frac{1}{q_G(P)} \right).$$

*Proof:* By definition,

$$-I(W; X|Y) = - \sum_{w,x,y} p(w, x, y) \log \frac{p(w, x|y)}{p(w|y)p(x|y)}.$$

Applying the log sum inequality,

$$\begin{aligned} -I(W; X|Y) &\leq \sum_y \left( p(y) \left[ \sum_{w,x} p(w, x|y) \right] \log \frac{\sum_{w,x} p(w, x|y)}{\sum_{w,x} p(w|y)p(x|y)} \right) \\ &= \sum_y \left( p(y) \log \sum_w \left( p(w|y) \sum_{x \in w} p(x|y) \right) \right) \end{aligned} \quad (1)$$

Adapting the technique in [6], we see that

$$\sum_{x \in w} p(w|y) = \sum_{x \in w} \sum_{z \in w} p(w, z|y) \quad (2)$$

$$\leq \sum_{(x, z) \text{ is not an edge}} p(w, z|y), \quad (3)$$

where the last line follows from the fact that  $w$  is an independent set. Inserting (3) into (1) concludes the lemma.  $\blacksquare$

Lemma II.2 is tight enough to show that difference approximation typically requires communication of the same size as the sets being compared.

**Theorem II.3** *Consider two sets  $X, Y \subseteq \mathcal{U}$  generated independently, with elements chosen with probability  $p$ . Then a one-way communication algorithm approximating differences within an additive error  $o(|\mathcal{U}|)$  requires at least  $\Omega(|\mathcal{U}|)$  bits of communication.*

*Proof:* Suppose an algorithm  $f$  approximates set differences within an additive error  $k$  that is  $o(n)$  where  $n \triangleq |\mathcal{U}|$ . Then the characteristic graph  $G$  of the function computed by this algorithm will have edges between all sets of distance  $> 2k$ , and the graph  $G'$  with only these edges lower-bounds the graph entropy of  $f$  because of the sub-additivity of  $H_G$  [7]. We can compute the probability  $q_{G'}(P)$  of two randomly chosen vertices not corresponding to an edge as follows:

$$q_{G'}(P) = \sum_{i=0}^{2k} \binom{n}{i} \alpha^i (1 - \alpha)^{n-i},$$

where two sets contain a given element with probability  $\alpha = p^2 + (1 - p)^2$ . Noting that  $\alpha(1 - \alpha) \leq \frac{1}{4}$  and that  $(1 - \alpha) \leq \frac{1}{2}$ , we get:

$$\begin{aligned} q_{G'}(P) &\leq \sum_{i=0}^{2k} \binom{n}{i} \frac{1}{4^i} \frac{1}{2^{n-2i}} \leq \frac{1}{2^n} \sum_{i=0}^{2k} \binom{n}{i}. \\ \log\left(\frac{1}{q_{G'}(P)}\right) &\geq n - \log\left(\sum_{i=0}^{2k} \binom{n}{i}\right) \\ &\geq n - k \log\left(\frac{n}{k}\right) \text{ which is } \Omega(n). \end{aligned}$$

In fact, Theorem II.3 can be trivially generalized.  $\blacksquare$

**Corollary II.4** Any algorithm on remote sets  $X, Y \subseteq \mathcal{U}$  returning an approximation  $f(X, Y)$  with

$$f_1(X \oplus Y) \leq f(X, Y) \leq f_2(X \oplus Y)$$

for some functions  $f_1, f_2$  such that

$$\exists c > 0 \quad f_1(x) > f_2(0) \quad \forall x > c|\mathcal{U}|$$

requires at least  $\Omega(|\mathcal{U}|)$  bits of one-way communication.

The difficulty of efficiently providing hard approximation guarantees leads us to the consideration of heuristic techniques.

### B. Existing solutions

Various existing techniques for approximating set difference size are surveyed nicely in [8]. A simple protocol for approximating differences involves random sampling, in which host  $A$  transmits  $k$  randomly chosen elements to host  $B$  for comparison. If  $B$  has  $r$  of the transmitted elements, then we approximate that  $\frac{r}{k}$  of the elements of  $B$  are common to  $A$ . The main problems with random sampling are a high error rate and low resolution, as we shall see in Section V.

The problem of determining similarity across documents [9] is also related to our work in this paper, though its solutions are generally more complicated due to the relative complexity of the similarity metric [10, 11]. Some of these approaches are based on clever sampling-based techniques called min-wise sketches [12]. Though better than random sampling, min-wise sketches suffer from poor data compressibility.

### C. Bloom filter basics

Bloom filters [13–15] are used to perform efficient membership queries on sets. The Bloom filter of a set is a bit array; each element of the set is hashed with several hashes into corresponding locations in the array, which are thereby set to 1 and otherwise 0. Testing whether a specific element  $x$  is in a set thus involves checking whether the appropriate bits are 1 in the Bloom filter of the set; if they are not, then  $x$  is certainly not in the set, but otherwise the Bloom filter reports that  $x$  is in the set. In the latter case, it is possible for the Bloom filter to incorrectly report that  $x$  is an element of the set (*i.e.*, a false positive) when, in fact, it is not.

The probability of a false positive of a Bloom filter for a set  $S$  is denoted  $P_f(S)$  and depends on the number of elements in the set  $|S|$ , the length of the Bloom filter  $m$ , and the number of (independent) hash functions  $k$  used to compute the Bloom filter. This false positive probability is given in [14] as

$$P_f(S) = \left( 1 - \left( 1 - \frac{1}{m} \right)^{k|S|} \right)^k. \quad (4)$$

### III. WRAPPED FILTER APPROXIMATION

Wrapped filters hold condensed set membership information with more precision than a Bloom filter. The additional precision comes at the expense of higher communication costs, but, surprisingly, this expense is outweighed by the benefits of improved performance. As we show later, wrapped filters often provide a more accurate approximation of set difference per communicated bit than traditional Bloom filters.

#### A. Wrapping

Wrapped filters are constructed in a fashion similar to counting Bloom filters [13, 14]. A wrapped filter  $W_S$  of a set  $S = \{s_1, s_2, s_3, \dots, s_n\}$  is first initialized with all zeroes, and then set elements are added to the filter by incrementing locations in  $W(S)$  corresponding to  $k$  independent hashes  $h_i(\cdot)$  of these elements. More precisely, we increment  $W_S[h_j(s_i)]$  for each set element  $s_i \in S$  and hash function  $h_j$  in order to construct the wrapped filter  $W_S$ .

The wrapped filter clearly generalizes the Bloom filter in that we may transform the former into the latter by treating all non-zero entries as ones. Host  $A$  can use this Bloom filter property of a wrapped function to determine  $|S_A - S_B|$  by inspecting  $B$ 's wrapped filter  $W_{S_B}$ ; in other words, all elements of  $S_A$  that do not fit the Bloom filter can be considered to be in  $S_A - S_B$ . Conversely, the unwrapping algorithm in Section III-B allows us to approximate  $|S_B - S_A|$  from the same wrapped filter, giving an overall approximation for the mutual difference  $|S_A \oplus S_B|$ .

Unlike Bloom filters, wrapped filters also have the feature of incrementally handling both insertions and deletions. Thus, whereas a Bloom filter for a set would have to be recomputed upon deletion of an element, one may simply decrement the corresponding hash locations for this element in the wrapped filter. The price for this feature is that each entry can now take any of  $kn$  values (where  $n = |S|$  is the size of the set being wrapped), requiring a worst-case

---

**Protocol 1** Unwrapping a wrapped filter  $W_{S_B}$  against a host set  $S_A$ .

---

**for each** set element  $s_i \in S_A$  **do**  
    copy  $W_{\text{temp}} = W_{S_B}$   
    **for each** hash function  $h_j$  **do**  
        **if**  $W_{\text{temp}}[(s_i)] > 0$  **then**  
             $W_{\text{temp}}[h_j(s_i)] = W_{\text{temp}}[h_j(s_i)] - 1$   
        **else** proceed to the next element  $s_i$   
    copy  $W_{S_B} = W_{\text{temp}}$   
**return** the approximate  $\delta_A = \frac{\sum_{i=1}^m W_{S_B}[i]}{k}$

---

of  $m \log(kn)$  bits of storage memory and communication for a filter of size  $m$ ; in contrast, Bloom filters require only  $m$  bits of communication. Fortunately, the expected case is for each entry to have only  $\frac{kn}{m}$  entries giving an expected multiplicative storage overhead of  $\log(\frac{kn}{m})$  over traditional Bloom filters.

### B. Unwrapping

Host  $A$  can unwrap a wrapped filter  $W_{S_B}$  to approximate  $|S_B - S_A|$ . This unwrapping procedure is presented formally in Protocol 1.

The strength of the wrapped filter rests in two features of the unwrapping algorithm. First, the *total weight* of the wrapped filter (*i.e.*,  $\sum_{s_i \in S} W_{S_B}(s_i)$ ) decreases as each set element is unwrapped. As a result, the false positive probability also generally decreases with each unwrapping, yielding a better overall approximate, as we shall see in Section IV.

The second feature of the wrapped filter is that it can sometimes compensate for false positives. Intuitively, when a false positive element is unwrapped from the filter, it prevents (at least) one other non false positive element from being unwrapped. Since we are only concerned with approximating the *number* of set differences, rather than actually determining the differences, this feature can mitigate the effect of the false positive.



## IV. ANALYSIS AND PERFORMANCE

### A. False positive statistics

Like traditional Bloom filters, wrapped filters admit a small probability of false positives during the unwrapping process. A false positive occurs when the hash values  $h_i(s)$  of an element  $s$  fit into the  $W_{S_B}$  despite the fact that  $s \notin S_B$ . Note that the expression for a false positive in a wrapped filter is identical to the Bloom filter false positive expression (4).

False positives can have two deleterious effects on our approximation procedure. First, they can reduce the overall set difference approximation by inappropriately reducing the weight of the wrapped filter. Secondly, a false positive can prevent a *valid* set element (*i.e.*, an element that is in the set intersection) from fitting in the resulting filter by reducing to zero (or causing to reduce to zero at some later time) one of the hash locations of the valid element. The inability to unwrap such valid elements adds error (in the form of unwanted residues) at the conclusion of the unwrapping.


The most accurate approximations are achieved when elements in  $S_A \cap S_B$  are unwrapped first, thereby reducing the false positive probability for other entries. The best and worst unwrapping paths are shown in Figure 1. The remaining cases can be summarized by a simple binomial-style upper bound. Specifically, the probability of  $\phi$  false positives occurring during a random unwrapping by host  $A$  of  $W_{S_B}$  is *at most*

$$\binom{|S_A|}{\phi} \left( P_f(|S_A|) \right)^\phi \left( 1 - P_f\left(\frac{\hat{\Delta}}{k}\right) \right)^{|S_A| - \phi}, \quad (5)$$

where  $\hat{\Delta}$  is the unwrapping algorithm's approximation.

### B. Effects of false positives

False positives introduce error in our approximation of the number of differences between two sets. Since the weight of our wrapped filter decreases with each unwrapping iteration, so does the false positive probability (as per (4)). In fact, the following theorem shows that, in expectation, the decrease in weight of the wrapped filter will be linear in the number of unwrapping iterations. This results in the superior performance of wrapped filters over Bloom filters in some cases, as we shall confirm experimentally.



trepezium.eps

Fig. 1. The wrapped filter weight never increases with unwrappings.

**Theorem IV.1** Consider a  $k$ -hash wrapped filter  $W$  with initial false positive probability  $p_f$  being (uniformly) randomly unwrapped. Then each unwrapping iteration decreases the expected weight of  $W$  by

$$\left(k \frac{|S_A| - d}{|S_A|}\right),$$

assuming that  $p_f \ll 1$  and given  $d = |S_A - S_B|$ .

*Proof:*

Let  $\Delta_i$  denote the decrease in the weight of the wrapped filter  $W_{S_B}^i$  after the  $i$ -th element of  $S_A$  is unwrapped. Then,

$$\begin{aligned} E(\Delta_i) &= k \cdot \left\{ \Pr(x_i \text{ fits } W_{S_B}^i, x \notin S_B) + \right. \\ &\quad \left. \Pr(x_i \text{ fits } W_{S_B}^i, x \in S_B) \right\} \\ &= k \cdot \left\{ P_f \left( \frac{w^{(i-1)}}{k} \right) \cdot \frac{d}{|S_A|} + \right. \\ &\quad \left. \Pr(x_i \text{ fits } W_{S_B}^i \mid x \in S_B) \cdot \frac{|S_A| - d}{|S_A|} \right\} \end{aligned}$$

Since the the false positive probability does not increase with unwrappings, and we assume  $P_f[w(0)] \ll 1$ , we may conclude that

$$E(\Delta_i) \longrightarrow \Pr(x_i \text{ fits } W_{S_B}^i \mid x \in S_B) \cdot \frac{|S_A| - d}{|S_A|}. \quad (6)$$

For the one remaining unknown term, we note that

$$\Pr(x_i \text{ fits } W_{S_B}^i \mid x \in S_B) = \left(1 - \frac{z(i)}{m}\right)^k, \quad (7)$$

where the right-hand size tends to 1 for  $p_f \ll 1$ . ■

The significance of Theorem IV.1 is that it identifies the expected wrapped filter behavior as corresponding to roughly the diagonal of the trapezium in Figure 1. This, in turn, determines (in expectation) the probability of error at any iteration of the decoding algorithm, and it is left as an open problem how to compute and correct for this bias.

We may also produce deterministic bounds on the effects of a false positive, as given by the following lemma.

**Lemma IV.2** *Each false positive will contribute an error of  $\varepsilon$  to our approximation, with*

$$-1 \leq \varepsilon \leq k - 1 \quad (8)$$

Note that for  $k = 1$ , the right hand size of (8) is 0, meaning that such wrapped filters will always produce a lower bound on the actual number of differences.

### C. Wrapped filter size and compression

On the one hand, it is important that the size  $m$  of the wrapped filter be as small as possible in order to minimize communication complexity. On the other hand, the accuracy of the wrapped filter approximation relies on  $m$  being as large as possible. It is possible to generalize the tradeoff analysis in [15] to lower bound the size of a compressed wrapped filter.

**Theorem IV.3** *The compressed size of a length  $m$  wrapped filter with  $k$  hash functions encoding  $n$  elements is (asymptotically) at most*

$$1.42 \left(1 - \frac{1}{m}\right) kn + 0.12m \text{ bits.}$$

*Proof:* Given an initial weight  $w = kn$ , the probability of a given wrapped filter location having weight  $i$  is given by a binomial distribution

$$p_i = \binom{w}{i} \left(1 - \frac{1}{m}\right)^{w-i} \left(\frac{1}{m}\right)^i.$$

Utilizing any scheme of entropy coding, we compress the average filter element to its entropy rate,  $H(p_i) = \sum_{i=0}^w p_i \log(p_i)$ . Using the normal distribution to upper bound this entropy gives

$$H(p_i) \leq \frac{\log(2\pi e)}{2} \left( \sum_{i=0}^{\infty} p_i i^2 - \left( \sum_{i=0}^{\infty} i p_i \right)^2 + \frac{1}{12} \right), \quad (9)$$

which can be manipulated to prove the theorem. ■

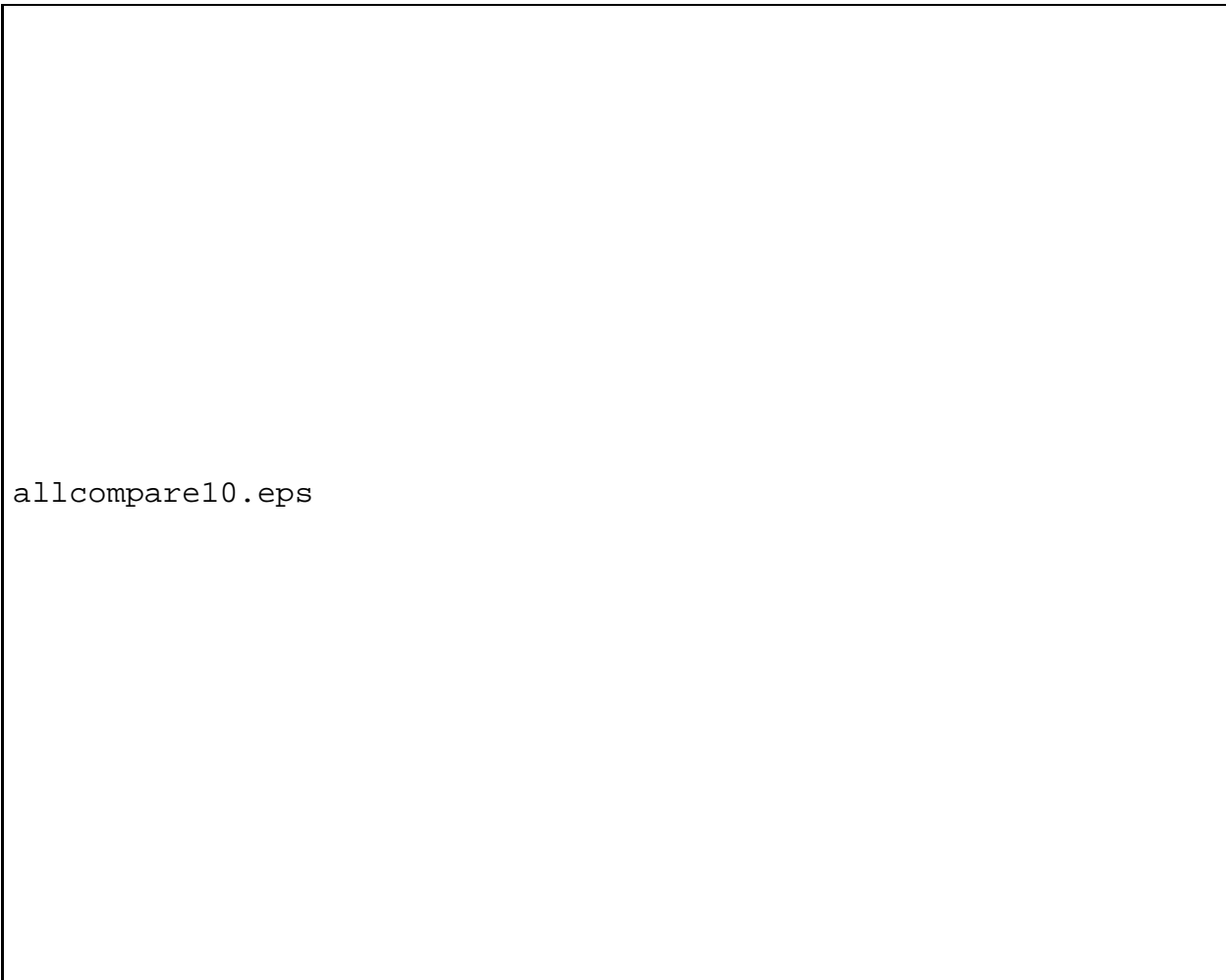


Fig. 2. Sample performance of the wrapped filter, Bloom filter, min-wise sketches, and random sampling: 100 differences, 500 trials.

## V. EXPERIMENTAL COMPARISONS

Figure 2 depicts a sample comparison of random sampling, min-wise sketches, standard Bloom filter, and wrapped filter approaches in approximating the number of differences between two sets; for sake of clarity, only three representative experiments are shown. A bar chart comparing the accuracies of various techniques is plotted as a function of their compressed communication complexity.

As may be expected, the standard Bloom filter and wrapped filter approaches are very similar for filter sizes that are large relative to host sets. For smaller filters, Bloom filters generally

provide a lower (and less accurate) approximation than wrapped filters, as predicted by our analysis. Random sampling, on the other hand, is quite inaccurate unless almost all the samples are transmitted; both min-wise sketches and random sampling have a high standard deviation compared to Bloom and wrapped filters.

## VI. CONCLUSIONS

We have shown that approximating set differences generally requires communication proportional to the amount of information between the sets. In the face of this limitation, we have analyzed and experimentally demonstrated several heuristic solutions. Our approach involves a Bloom filter known as a wrapped filter, whose data structure is also known as a counting Bloom filter in the literature. Our novel decoding algorithm for the wrapped filter allows it to be generally more accurate than the standard Bloom filter for the same communication complexity. In addition, the wrapped filter technique is non-interactive and can be easily maintained in an incremental fashion as data is inserted or deleted from a set. The encoding and decoding process can also be made computationally efficient. All these qualities make wrapped filters particularly suitable for the many network applications where there is a need to quickly and efficiently measure the consistency of distributed information.

## ACKNOWLEDGMENTS

The authors are grateful to John Byers, Saikat Ray, and David Starobinski.

## REFERENCES

- [1] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for PDAs and mobile devices," *IEEE Network*, vol. 16, no. 4, July 2002.
- [2] A. C. Yao, "Some complexity questions related to distributive computing," in *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, 1979, pp. 209–213.
- [3] A. Orlitsky and J. R. Roche, "Coding for computing," *IEEE Transactions on Information Theory*, vol. 47, pp. 903–917, March 2001.
- [4] J. Körner, "Coding of an information source having ambiguous alphabet and the entropy of graphs," *Proceedings of the 6th Prague Conference on Information theory*, pp. 411–425, 1973.
- [5] H.S. Witsenhausen, "The zero-error side information problem and chromatic numbers," *IEEE Trans. on Info. Theory*, vol. 22, no. 5, September 1976.
- [6] J. Körner and K. Marton, "Random access communication and graph entropy," in *IEEE Transactions on Information Theory*, vol. 34, pp. 312–314.

- [7] G. Simonyi, “Graph entropy: a survey,” in *Combinatorial Optimization, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1995, vol. 20, pp. 399–441.
- [8] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, “Informed content delivery across adaptive overlay networks,” *ACM SIGCOMM*, August 2002.
- [9] U. Manber, “Finding similar files in a large file system,” in *Proceedings of the USENIX Winter 1994 Technical Conference*, San Francisco, CA, USA, 1994, pp. 1–10.
- [10] D. S. Hirschberg, “Serial computations of Levenshtein distances,” in *Pattern matching algorithms*, A. Apostolico and Z. Galil, Eds., pp. 123–141. Oxford University Press, 1997.
- [11] R.A. Wagner and M.J. Fisher, “The String-to-String Correction Problem,” *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, January 1974.
- [12] Broder, “On the resemblance and containment of documents,” in *SEQS: Sequences '91*, 1998.
- [13] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.
- [14] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, “Summary cache: a scalable wide-area Web cache sharing protocol,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.
- [15] M. Mitzenmacher, “Compressed bloom filters,” in *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, Newport, Rhode Island, USA, August 2001, pp. 144–150, ACM Press.