Spitty Bifs are Spiffy Bits: Interest-Based Context Dissemination Using Spatiotemporal Bloom Filters*

Evan Grim and Christine Julien

Mobile and Pervasive Computing Lab
The University of Texas at Austin
{evangrim,c.julien}@mail.utexas.edu
http://mpc.ece.utexas.edu/

Abstract. Acquiring accurate context information is crucial to mobile and pervasive computing, and *sharing* context among nodes enables unique applications. As context information and the applications that consume it become increasingly diverse, they will need an efficient means to indicate tailored interest in this context information. This paper proposes a new probabilistic data structure, spatiotemporal Bloom filters (SpTBF) or "spitty bifs," which allow nodes to efficiently store and share their context interests. SpTBF provide both spatiotemporal locality and a fine-grained ability to control how context interests are disseminated. SpTBF are evaluated by modifying the Grapevine context sharing framework to inform its context dissemination capabilities, and the benefits are characterized in a variety of network scenarios.

Keywords: context awareness, publish/subscribe, mobile computing, bloom filters.

1 Introduction

Mobile and pervasive computing applications are strongly influenced by their environments, and this has driven research to seek effective and efficient means for sensing, characterizing, and acting upon this valuable context information. Many approaches focus on purely egocentric notions of context, which limit the information a node can directly collect or otherwise infer about its surroundings. Mechanisms that allow nodes to efficiently *share* context information with nearby nodes enhance local notions of context, enabling applications to better react to their environments and allowing applications to leverage the shared context of groups to stretch beyond solely egocentric approaches. For example, a group of nodes that can individually only sense the direction in which an object of interest lies can triangulate that information into a shared notion of *where* the object is.

* This work was funded in part by the National Science Foundation (NSF), Grant #CNS-0844850 and Grant #OCI-0753360. The views and conclusions herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

K. Zheng, M. Li, and H. Jiang (Eds.): MOBIQUITOUS 2012, LNICST 120, pp. 164–175, 2013.

Simple context dissemination approaches work well in networks where every node is interested in the same context information throughout the system's lifetime. Efficiently sharing context becomes challenging when interests are less static, and scenarios with more diverse and dynamic context needs abound. Many smart phones run applications that have widely varying interests in available context information. Even within a single application context needs may change over time. For example, a vehicular application providing localized traffic and safety information may find information from cars beside or near the interstate of no interest to vehicles speeding by. Furthermore, individual users may have differing interests. For instance, an application facilitating social interaction among park patrons need not enumerate all the capabilities of a multi-sport athlete if all nearby patrons are interested in quieter games like chess.

Efforts that ignore these dynamics and heterogeneities will either share too much context information, resulting in wasted network resources, or share too little information, missing opportunities for providing valuable context. These scenarios suffer when distribution relies on the *producer* of context information to know what information is desired by nearby nodes, when it is the *consumers* who are best suited to provide this information. What is needed is a lightweight means for nodes to communicate consumers' context *interests* to nearby producers.

This paper's novel contribution is a means for applications to efficiently maintain an awareness of what context information is of interest to other nearby nodes and a demonstration that this awareness improves the use of network resources significantly. We introduce a Bloom filter variant, spatiotemporal Bloom filters (SpTBF or "spitty bifs" for short), that allow a node to efficiently communicate, acquire, and maintain information about others' context interests within the node's spatiotemporal region of a mobile network. Our approach is rooted in the *publish/subscribe* paradigm, which has demonstrated efficiency improvement in the heterogeneous and dynamic network systems we target. The SpTBF approach can represent generic types of context and provides consumers the flexibility to control the scope in space and time within which their individual interest is distributed. In this paper, we introduce the SpTBF data structure, demonstrate it in practice by applying it to the Grapevine context dissemination framework [9], and evaluate its performance.

2 Related Work

Our task is to track context interests within a spatiotemporal region; our approach is informed by the domain of publish/subscribe (pubsub) mechanisms. We survey these predecessors, first examining more generic approaches that build an overlay to aid in the distribution and matching of publications and subscriptions. We then focus on approaches that leverage spatial and/or temporal properties for distribution, and discuss how they do not sufficiently address our challenges.

Many approaches provide pubsub capabilities by constructing overlay routing structures that hierarchically organize the network. The content based approach of [11] eschews address based routing entirely, opting instead to route only based

on content and leveraging Bloom filters to compress the routing information required to inform forwarding decisions. Other efforts maintain traditional addressing but cluster nodes in tree structures based on shared interests [2,10,12,18]. These approaches provide interest-based distribution but require a relatively high communication burden to maintain the overlay, especially in dynamic networks. We strive to reduce this burden by recognizing that many applications involve context information that is most useful to nodes that are *here* and interested *now* and thus are served best by pubsub mechanisms that use inherent spatial and temporal locality to quickly and efficiently identify interested consumers.

Other efforts have investigated pubsub informed by spatial and temporal properties. TACO-DTN [19], B-SUB [21], and ZigZag [22] subscribe to information using specified timing conditions, while [6] enriches subscriptions with location. These approaches provide improved efficiency through more granular subscription specifications that prevent unnecessary use of network resources. Other work leverages location to guide information to "bazaars" [17] or provide informed guesses as to where in the network information is most likely to be useful [5,6,14,15,16]. Many of these lessons are synthesized in the abstract context pubsub model in [8], which provides an expressive and generic system for spatially and temporally guided subscriptions. Our work diverges from these approaches by leveraging the fact that interest in context information will often be concentrated near where that information is generated. This allows us to simplify the distribution of interest information by enabling nodes to indicate interests that automatically decay over distance and time, i.e., a subscription is concentrated in space and time around its originator.

As in other approaches [10,11], SpTBF leverage the efficiencies made possible by probabilistic data structures. We introduce a variant of the Bloom filter [3], which has enjoyed recent popularity in network-centric applications due to its ability to encode large amounts of information within very small space requirements [4]. The original Bloom filter allows set membership to be encoded in an array of m bits using k hash functions to transform a potential set member into addresses within the bit array. To insert an element, each of the k addressed locations in the array is set to 1. Querying whether or not a value is in the set involves hashing the value using the same k functions and checking to see if all the bits addressed by the hash values are set. If any of the bits are 0, then it is certain that the value was never added to the set. If all the bits are 1, then the item is *probably* in the set. Applications use this functionality to solve varying problems (e.g., quick cached value checks [4], privacy-preserving algorithms [13]).

One downside to Bloom filters is that there is no way to reliably remove an element from the set. The Counting Bloom Filter (CBF) [7] replaces the array of bits with an array of counters. Instead of setting a bit to 1 upon insertion, a CBF increments the k associated counters; removal decrements the counters. The ability to remove comes at the cost of additional space requirements and introduces the possibility of a false negative (there is a small chance the associated counters may all have been decremented by unrelated remove operations [4]).

B-SUB proposes a further extension, the Temporal Counting Bloom Filter (TCBF) [21,22], which modifies the semantics of the counters in CBF so that entries are set to a time value representing the length of time the entry should remain in the structure. As time passes, maintenance operations decrement all counter values by the amount of time elapsed[1]. When any of the k counter values for an entry reaches 0, the entry is no longer in the set. This allows entries to automatically expire after a given amount of time and thus provides a subscription mechanism with a temporal component; however, the TCBF has no notion of spatial proximity.

3 SpittyBifs

This paper proposes a Bloom filter variant that provides the spatial component. We call this variant a *spatiotemporal bloom filter* (SpTBF), or "spitty bif" for short. It uses a construction similar to that found in TCBF involving k hash functions (h_i from $1 \leq i \leq k$) but replaces the array of timer values with an array a of m 2-tuples, each containing a hop limiter and a timer. Fig. 1 shows an example of a small SpTBF populated with values. The hop limiter is decremented each time the structure is passed from one node to another[2]. Timer values are all relative to a specific point in time, so each SpTBF also includes a timebase t_b that records this time value, allowing operations to decay values relative to the current time (as in TCBF). Entries are automatically removed from a SpTBF when any of the tuple value components reaches zero, indicating that *either* the structure has traveled a specified number of hops from its origin *or* a given amount of time has elapsed.

hops	time
4	6
0	0
1	21
0	0
1	21
0	0
4	6
4	21

timebase = 42

Fig. 1. SpTBF

Operations

A SpTBF supports the typical Bloom filter operations, allowing *insert*, *query*, and *merge* [3]. Two housekeeping operations are also required (*decrementHops* and *adjustTime*); these decay the tuple components in space and time. These operations are similar to those found in TCBF, however the use of a tuple requires a slightly different approach than that employed when the underlying array holds a single value.

Table 1. Operation signatures

$insert\,(label, hopLimit, timeLimit, currentTime)$
$query\,(label, currentTime)$
$merge\,(currentTime)$
$decrementHops\,()$
$adjustTime\,(currentTime)$

[1] This restores the perfect false negative properties lost in CBF, since counter values are all decremented simultaneously.

[2] For simplicity we use "hops" (number of node traversals) as a spatial metric. Future work could add additional tuple entries to encode more precise notions of location.

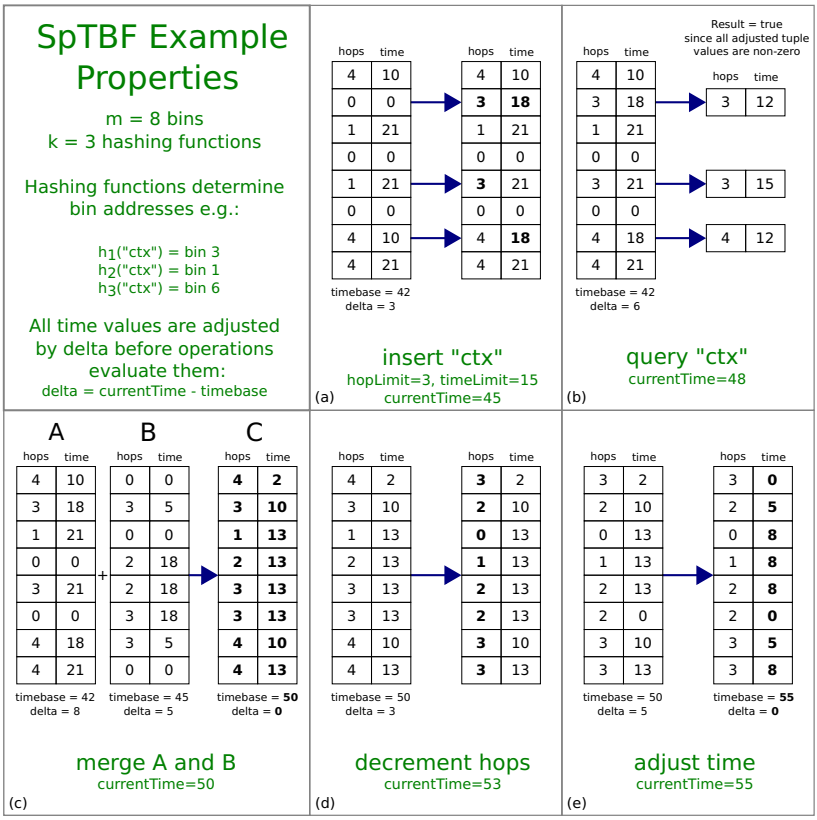

Fig. 2. SpTBF Operations

Insertion. To insert an item with label l, we use the k hash functions that each map l into one of the m addresses in the SpTBF's array a of tuples. The spatial and temporal components from each tuple are compared with l's specified limits, and the maximum of each forms the new tuple, which is returned to the same location in the array. This results in the following operation:

$$\forall i \in \{1, \ldots, k\} \left\{ \begin{array}{c} a\left[h_i\left(l\right)\right].hop = max\left(a\left[h_i\left(l\right)\right].hop, hopLimit\right) \\ a\left[h_i\left(l\right)\right].time = max\left(a\left[h_i\left(l\right)\right].time, timeLimit + \delta\right) \end{array} \right\}$$

The *timeLimit* is adjusted by δ, the difference between the current time and the timebase used for all the temporal components in the SpTBF. Fig. 2(a) shows an example, where the starting SpTBF is shown on the left (which already contains context interest elements). The label "ctx" is inserted with a *hopLimit* of 3, and a *timeLimit* of 5, resulting in the updated SpTBF on the right.

Query. Querying whether an item with label l exists in a SpTBF again uses the k hash functions to retrieve k tuples from the array a. Iterating through

these tuples, both the spatial and temporal components are checked. If either component from any of the tuples is zero, l is not in the set—either it was never inserted in the set or at least one of its components has decayed to zero. If all the tuples' components have non-zero values, then the item is likely to be in the set, and the query result is true. This operation can be expressed as:

$$\begin{cases} true & \text{if for } \forall i \in \{1, \ldots k\} \ a\left[h_i\left(l\right)\right].hop \neq 0 \wedge a\left[h_i\left(l\right)\right].time - \delta > 0 \\ false & \text{otherwise} \end{cases}$$

Again, the temporal component is adjusted by the difference between the current time and the structure's timebase. Fig. 2(b), shows that the newly inserted item labeled "ctx" will result in an affirmative query because all of the bins chosen by the hashing functions hold non-zero tuple elements.

Merge. Multiple SpTBF can be merged into a single structure by choosing the dominant tuple values from each to create a space efficient representation of the consolidated set memberships. Both structures must have the same underlying array a size of m; merging involves iterating over *all* the tuples in each, comparing their spatial and temporal components and storing the maximum of each in the new array at the same location. Merging A and B to create C is as follows:

$$\forall i \in \{1 \ldots m\} \left\{ \begin{array}{c} a_C\left[i\right].hop = max\left(a_A\left[i\right].hop, a_B m\left[i\right].hop\right) \\ a_C\left[i\right].time = max\left(a_A\left[i\right].time - \delta_A, a_B\left[i\right].time - \delta_B\right) \end{array} \right\}$$

$$t_{b_C} = current\, time$$

Each temporal component is adjusted by its timebase's difference from the current time. The temporal components are then relative to the current time, and as such the new SpTBF uses the current time as its own timebase. Fig. 2(c) shows an example. The new structure may end up using the spatial component from one structure and the temporal component from another. For example, imagine a SpTBF entry has existed locally for long enough that its temporal component(s) are on the cusp of expiring, but since the structure has stayed local, its hop count is still strong. If it is merged with a SpTBF with a more recent entry that shares an array slot, but one that has come from several hops away, then the resulting array entry would use the first spatial component and the second temporal component. In these scenarios an entry will persist beyond its originally intended spatiotemporal components if all of its array entries fall prey to this situation. The likelihood of these collisions is determined by the underlying array size m and the number of hashing functions used [4]; applications can control this likelihood by appropriately tuning these parameters.

Housekeeping. Two additional operations support the decay of spatial and temporal components, facilitating automatic removal of elements.

Spatial. Used whenever the SpTBF travels a network hop (i.e., is transmitted from one node to another), spatial decay is accomplished by a simple decrement

of all the non-zero spatial components within each of the m tuples in the array a, as seen in Fig. 2(d), and defined in the *decrementHops* operation:

$$\forall i \in \{1 \ldots m\}\ a[i].hop = max\{a[i].hop - 1, 0\}$$

Temporal. Decay in time involves adjusting the temporal tuple component values and the timebase to which they are relative to match the current time. Periodically performing this operation minimizes the amount of space required to represent the temporal tuple component and is also useful when sharing with external entities that may not use the same reference clock (e.g., preparing to transmit a SpTBF over a network connection). This adjustment is accomplished by calculating the difference between each temporal component's timebase-adjusted value and the current time, and then subtracting this value from each temporal entry, as seen in Fig. 2(e). The structure's timebase is set to the current time:

$$\forall i \in \{1 \ldots m\}\ a[i].time = max\{a[i].time - \delta, 0\}$$

$$t_b = current\,time$$

Application

SpTBF allow nodes to maintain an awareness of interest in context information unique to their location in the network; nodes can use this awareness to share only context information known to be of interest to nearby nodes. Tracking this interest requires each node to store a single SpTBF, which is initially empty (all tuple values initialized to zero) and is populated upon encountering other nodes. When such an encounter occurs, a node creates a copy of its SpTBF, *adjusts* its timebase to the current time, *decrements* the spatial components, and then *inserts* labels for each context item it is interested in receiving before sharing the newly constructed SpTBF with its neighbor. Upon receiving a SpTBF, a node sets the timebase for the received SpTBF to its current timeand *merges* the received SpTBF with its own. The node can then *query* its SpTBF to determine whether nearby nodes are interested in receiving context information it has available. Sharing the context of interest is outside the purview of the SpTBF structure; it can be handled by a variety of existing techniques [9].

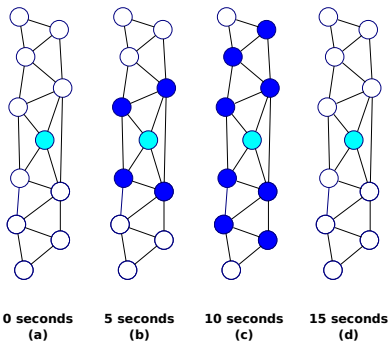

Fig. 3. Interest Lifecycle

Fig. 3 shows an example of a lifecycle of interest awareness for a single context interest. In (a), the center node determines that it will be interested in a given item of context from neighbors up to 2 hops away and for the next 15 seconds. It informs its neighbors of this interest by *inserting* the context item's label into outgoing SpTBF with a spatial component of 2 hops and a temporal component

of 15 seconds. Within the next 5 seconds, as seen in (b), the node has shared this interest with its direct neighbors. After another 5 seconds, each of these neighbors has shared their interest awareness with their own neighbors and the central node's interest has propagated to the limit of its spatial component. After another 5 seconds, the temporal component has decayed to zero, and the central node's interest in the context information will expire from each node's SpTBF. The potential *consumer* of context can specify the spatial and temporal lifetimes of each of their interests individually, allowing fine grained control over how wide a net to cast for desired context information on an item-by-item basis.

4 Evaluation

To evaluate the usefulness of SpTBF, we augmented the Grapevine context dissemination framework [9] to use SpTBF to track interest awareness and to improve context information sharing efficiency. Grapevine provides programmers a simple and efficient library for sharing context and collaboratively forming groups based on that shared context. Grapevine shares context by encoding it into space-efficient *context summaries* that are piggybacked onto outgoing traffic. In Grapevine, the onus of determining what context information to share rests on the producers of that information. With a few simple modifications to the existing framework, Grapevine can leverage SpTBF to send an *interest summary* in addition to context summaries. Nodes do not include context information in their context summaries until they know (via a received interest summary) that a nearby node indicated interest in that context label.

The SpTBF interest summaries pack the spatial and temporal tuple components into 1 byte. Each spatial component uses 2 bits, allowing up to 3 hops to be specified, and each temporal component uses 6 bits, allowing just over a minute (64 seconds) to elapse[3]. In total, an interest summary requires one byte per array entry and an additional four bytes for the baseline timestamp.

We evaluated Grapevine with SpTBF interest summaries using OMNet++ [20] and INET-MANET [1] to realistically simulate the communications of mobile *ad hoc* networks. Our goal is to characterize when an interest-enabled context dissemination mechanism outperforms an interest-agnostic approach and gives rise to two important questions: (1) how heterogeneous does context interest need to be to benefit from tracking interest; and (2) how much space should be allocated to the interest summary?

How heterogeneous does context interest need to be to benefit from tracking interest? Interest-based dissemination requires additional bytes to be transmitted; the goal is that this overhead can be reclaimed in savings from context information that does not need to be shared. If all the nodes are interested in all the available context, then the overhead of the interest communication is wasted.

To examine this question, we used 100 interest tracking Grapevine nodes using INET-MANET's IEEE 802.11b network layer and mass mobility profiles to

[3] These choices are specific to our particular evaluation and can easily be adapted for different spatial and temporal scopes.

simulate nodes traveling at a natural human walking speed and with realistic mobility. We evaluated their network resource use in a variety of scenarios, varying node density, interest allocation, and the number of hops over which interest and context information were shared to determine the savings that interest tracking provided over Grapevine's previous resource consumption. Each scenario was run for a simulated period of 20 minutes. We collected data using two different arena sizes, one which provided a 5 km by 5 km square area, simulating a sparse node density where nodes would encounter new neighbors relatively infrequently (approximating a neighborhood), and a smaller arena of 1 km by 1 km, simulating a more densely populated area (approximating a popular park).

We allocated interests node in two different ways. In the first, each node randomly selects a percentage of the available context labels for which they indicate interest. In the second, we divide the context labels among a set of applications and assign those applications to nodes in equal proportions. These allocations are evaluated using the sparse and dense configurations and with a handful of hop limits that determine how many hops both interest and context summaries are forwarded. For each scenario, we compare the total number of bytes sent by the nodes; the results are reported as percent savings of the SpTBF approach over the interest-agnostic approach in the same scenario.

Figs. 4 and 5 show that savings can be significant for sparse networks with diverse context interests. The savings are strongest in sparse scenarios with more heterogeneous interest, where each node is interested in only a small portion of available context ($< 30\%$ of the available contexts, as shown in Fig. 4) or when more applications are present (> 3 applications, as shown in Fig. 5). As the network density increases, so does the likelihood that a neighbor is interested in the context a node has to share; it becomes increasingly likely that all context information is shared from every node. As a result, savings are clearly less compelling in denser networks, but there is very little penalty in adding the SpTBF interest summaries (averaging at approximately five percent overhead) making interest dissemination potentially beneficial in networks with varying needs.

How much space should be allocated to the interest summary? The amount of space allocated to the interest summary can be chosen arbitrarily by using different values of m (the number of bins into which the tuples can be placed). However, smaller values of m will result in false positive rates that negate the benefits of interest tracking. To characterize the effects of this choice on false positive rates, we allocated a variety of sizes for the interest summaries and measured how much unnecessary context information was shared due to false positives indicating there was interest when if fact there was not. The results are reported in additional percentage of savings that could have been achieved had these false positives not occurred. Fig. 6 shows the savings missed due to false positives for various percent interest allocations (using the first scheme for interest allocation in which each node is assigned a random set of the available context types). We used a hop limit of three for all interest disseminations[4].

[4] Graphs for other configurations are similar and are omitted for brevity.

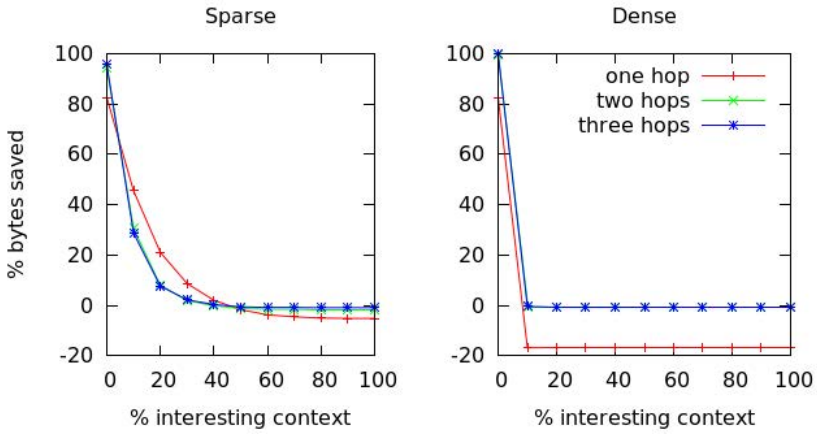

Fig. 4. Percent Allocation

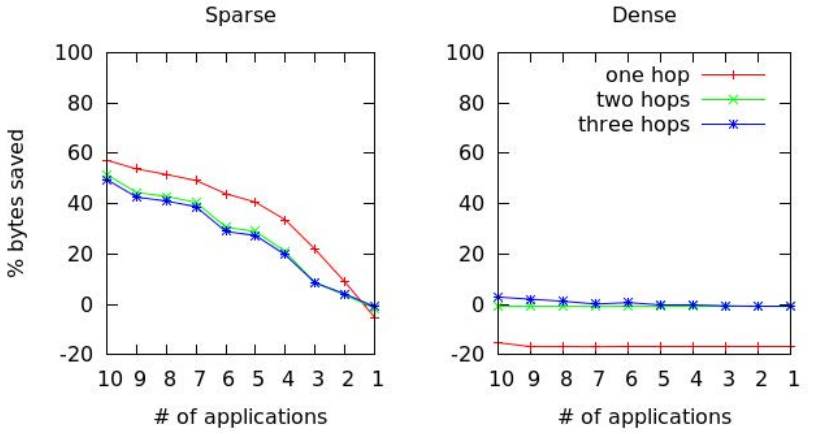

Fig. 5. Application Allocation

Additional space allocated for interest summaries are important in systems with heterogeneous interest, where the extra space provides significant benefit. However, allocating space for bins beyond the number of context items available (100 in this case) is unlikely to provide additional benefit. Furthermore, systems with largely homogeneous interest can safely reduce their allocations without significantly impacting the amount of context sent. This may allow the more heterogeneous parts of context sharing systems to benefit from interest-based context dissemination, while minimizing the negative impacts on more clustered or homogeneous parts of the network.

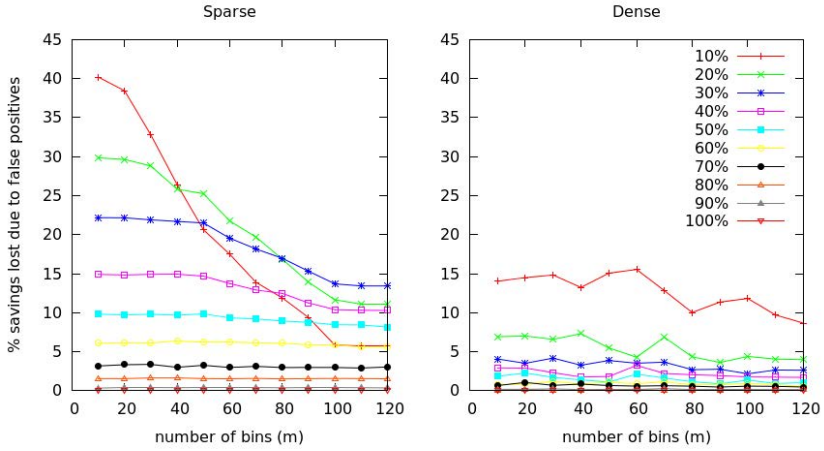

Fig. 6. Lost Savings Due to False Positives

5 Reflection

Sharing context information in mobile and pervasive computing environments is necessary to supporting expressive and flexible application behavior. While many approaches like the Grapevine context dissemination framework support sharing context information, they traditionally accomplish context dissemination while ignoring what interest nearby nodes have in context information. In this paper, we clearly demonstrated that the efficiency of Grapevine's context dissemination can be improved by enabling nodes to share their interests in available context information. Specifically, we explored using a novel probabilistic data structure (the spatiotemporal Bloom filter, SpTBF, or "spitty bif") to provide this interest awareness with a minimum impact on network resources. The SpTBF ensures that context information shared among network nodes is useful.

Reflecting on the results in this paper, we have opened several potential avenues for future work. First, in situations when nodes' overlap in context interest is high (e.g., $> 30\%$), the benefit of the SpTBF data structure is limited. Future work should investigate how and when the benefits of SpTBF can be achieved in these additional situations (e.g., in more diverse network scenarios). As described earlier, incorporating additional definitions of *space* in addition to network hops is also an item of future work.

Longer term, the traditional Bloom filter concept as we apply it in SpTBF is binary—a value is either (probably) in the structure or it is not. The SpTBF structure contains additional semantic information that could, for example, provide information about the *strength* of interest in a particular context label. Not only could this information be used to influence future context information dissemination, but applications might use it in other ways as well, for example influencing node movement (e.g., nodes may move towards where interests lie) or setting up interest *gradients* that can direct context information flows.

The presented SpTBF represents a promising new tool that aids in efficient and expressive sharing of context information that is relevant *locally* in space and time. This has potential to be useful in many application and network scenarios.

References

1. The INET-MANET Framework for OMNeT++
2. Balakrishnan, D., Nayak, A.: CSON-D: Context dissemination in autonomous systems using overlays. In: Proc. of IE (2008)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Comm. of the ACM 13(7), 422–426 (1970)
4. Broder, A., Mitzenmacher, M.: Network Applications of Bloom Filters: A Survey. Internet Mathematics 1(4), 485–509 (2004)
5. Carreras, I., De Pellegrini, F., Miorandi, D., Tacconi, D., Chlamtac, I.: Why neighbourhood matters: interests-driven opportunistic data diffusion schemes. In: Proc. of CHANTS, p. 81 (2008)
6. Eugster, P., Garbinato, B., Holzer, A.: Location-based Publish/Subscribe. In: Proc. of NCA, pp. 279–282 (2005)
7. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: a scalable wide-area Web cache sharing protocol. IEEE/ACM Trans. on Networking 8(3), 281–293 (2000)
8. Frey, D., Roman, G.-C.: Context-Aware Publish Subscribe in Mobile Ad Hoc Networks. In: Murphy, A.L., Vitek, J. (eds.) COORDINATION 2007. LNCS, vol. 4467, pp. 37–55. Springer, Heidelberg (2007)
9. Grim, E., Fok, C.-L., Julien, C.: Grapevine: Efficient situational awareness in pervasive computing environments. In: Proc. of Percom Workshops (2012)
10. Hebden, P., Pearce, A.R.: Data-Centric Routing using Bloom Filters in Wireless Sensor Networks. In: Proc. of ICISIP, pp. 72–77 (December 2006)
11. Jerzak, Z., Fetzer, C.: Bloom filter based routing for content-based publish/subscribe. In: Proc. of DEBS, vol. 71 (2008)
12. Khambatti, M., Ryu, K.D., Dasgupta, P.: Structuring peer-to-peer networks using interest-based communities. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) DBISP2P 2003. LNCS, vol. 2944, pp. 48–63. Springer, Heidelberg (2004)
13. Korkmaz, T., Sarac, K.: Single packet IP traceback in AS-level partial deployment scenario. In: Proc. of GLOBECOM, p. 5 (2005)
14. Leontiadis, I.: Publish/subscribe notification middleware for vehicular networks. In: Proc. of Middleware Doctoral Symposium, pp. 1–6 (2007)
15. Leontiadis, I., Costa, P., Mascolo, C.: Persistent content-based information dissemination in hybrid vehicular networks. In: Proc. of Percom, pp. 1–10 (2009)
16. Leontiadis, I., Mascolo, C.: Opportunistic spatio-temporal dissemination system for vehicular networks. In: Proc. of MobiOpp, p. 39 (2007)
17. Motani, M., Srinivasan, V., Nuggehalli, P.S.: PeopleNet: engineering a wireless social network. In: Proc. of MobiCom (2005)
18. Patel, J.A., Rivière, É., Gupta, I., Kermarrec, A.-M.: Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. Computer Networks 53(13), 2304–2320 (2009)
19. Sollazzo, G., Musolesi, M., Mascolo, C.: TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks. In: Proc. of MobiOpp (2007)
20. Vargas, A.: OMNeT++ Web Page
21. Zhao, Y., Wu, J.: B-SUB: A Practical Bloom-Filter-Based Publish-Subscribe System for Human Networks. In: Proc. of ICDCS, pp. 634–643 (2010)
22. Zhao, Y., Wu, J.: ZigZag: A Content-Based Publish/Subscribe Architecture for Human Networks. In: Proc. of ICCCN, pp. 1–6 (July 2011)