# B-SUB: A Practical Bloom-Filter-Based Publish-Subscribe System for Human Networks

Yaxiong Zhao and Jie Wu
Department of Computer and Information Sciences
Temple University
{yaxiong.zhao, jiewu}@temple.edu

***Abstract*—The adoption of portable wireless devices is rapidly rising. The demand for efficient communication protocols amongst these devices is pressing. In this paper, we present a content-based publish-subscribe system, called B-SUB (Bloom-filter-based pub-SUB system), for the networks formed by human-carried wireless devices, which are called human networks (HUNETs). A novel data structure, called Temporal Counting Bloom Filter (TCBF), is proposed to perform content-based networking tasks. The TCBF's novelty is that it is able to handle temporal operations, which are not supported in the classic Bloom filter (BF) and are crucial to the success of forwarding messages in HUNETs. B-SUB uses TCBFs to encode users' interests and embed routing information. Using the TCBF, B-SUB can propagate interests by transmitting at most two TCBFs of dozens of bytes, which makes B-SUB space-efficient. B-SUB makes forwarding decisions through querying the TCBFs, which is simple and fast. These designs make B-SUB pretty suitable for resource-constrained HUNETs. However, the TCBF has false positives, which will potentially cause useless messages to be injected into the network. The issue that arises here is how to handle its false positives in queries, and at the same time maintain its spacial efficiency as well. So, we analyze several methods for controlling the TCBF's false positive rate. B-SUB's viability and usefulness are verified through extensive simulation studies using real-world human contact traces.**

***Index Terms*—Publish-subscribe, Bloom filter, delay tolerant networks, human networks, social network analysis.**

## I. Introduction

The fast paced development of portable wireless devices and wireless communication technologies has enabled ubiquitous networking capability for individuals. Currently, such devices are connected through wireless infrastructures. Recently, the research of delay tolerant networks (DTNs) has provided a new architecture for organizing intermittently-connected wireless devices without the aid of a pre-existent infrastructure. Therefore, we envision a new type of highly versatile and dynamic networks composed of human-carried wireless devices, which are called *human networks* (HUNETs). Fig. 1 shows a HUNET composed of 4 users.

The efficient communication protocols for HUNETs are of paramount importance to its success. However, there are several issues associated with the design of the communication protocols in HUNETs. First, the applications in HUNETs require content-based networking services. *Content-based networking* [5] is a novel style of communication that associates source and destination pairs based on actual content and interests, rather than letting source nodes specify the destination.

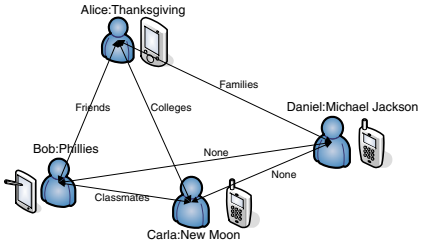



Fig. 1. An example of HUNETs. Each person is equipped with a mobile device. Their contact patterns are governed by their relationships. Each person has his/her own interests, which is represented as a *name:interest* pair. Messages are transported between users via (multi-hop) store-carry-forwarding.

Content-based networking allows ad hoc and autonomous access to network content.

The rise of social networking sites, like Twitter [18], beacons a new trend in network applications and services. Due to the direct correspondence between such applications and people's lives, these applications are able to infiltrate into every detail of people's lives. Due to its portability, wireless devices are becoming a desirable platform for such applications. For example, Twitter [18] is now widely used on wireless devices. We believe that, in the near future, social networking applications based on HUNETs will be prevalent. In such applications, people are interested in various topics, and the messages are forwarded according to their contents and users' interests. Therefore, it is required that the protocols of HUNETs should support content-based networking. An example is Bluejacking [2], where bluetooth users send messages when they are in the same place. The communication in this scenario can be made more efficient if content-based networking is used.

Second, the protocols used in HUNETs must be simple and efficient. The devices in HUNETs are powered by batteries, which limits their abilities to perform computational and communication tasks. So, they cannot afford complex protocols and high overhead. The memory capacity of the nodes in HUNETs is also limited, so the protocols have to be space-efficient.

Finally, HUNETs are different from general DTNs, thus require non-conventional protocol designs. Although HUNETs can be modeled as DTNs, they are different from general

1063-6927/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDCS.2010.89



IEEE computer society

Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.

DTNs in that their network dynamisms directly represent people's activity in a social group. As as result, the contact patterns of HUNETs are different from that of the general DTNs. Thus, we must consider the question of how to effectively exploit the benefits of the social contact patterns in designing protocols for HUNETs.

Existing DTN routing protocols are not well suited in HUNETs because: 1) they are based on the traditional IP-networking paradigm instead of content-based networking; 2) although many existing protocols [24] utilize social structures in DTNs, which is suitable in HUNETs, these protocols require complex off-line processing to obtain the required information for routing.

In this paper, we present a content-based publish-subscribe system called, B-SUB, which stands for "Bloom filter-based pub-SUB". In B-SUB, messages are identified by strings that summarize their contents, which are called *keys*. Users' interests are also represented as keys. *Publish-subscribe* (pub-sub for short) paradigm [14] is used in B-SUB. The pub-sub paradigm frees the users from tedious addressing and routing tasks. In a pub-sub system, message *producers* and *consumers* are agnostic of each other. Messages are forwarded solely by *brokers*, which perform content matching for the users.

B-SUB logically has two components: *broker allocation* and *pub-sub forwarding*. Our broker allocation scheme is based on our previous work [30] on socially-aware pub-sub routing. In B-SUB's broker allocation scheme, a swarm of socially-active nodes are selected to be brokers, based on the social contact patterns of the nodes. Normal users do not participate in interests propagation and message forwarding, which reduces the overall overhead in the system.

We use Bloom filters (BF) to encode users' interests. The BF is a space-efficient data structure for representing sets, which supports probabilistic membership querying. The BF maps a key, through multiple hash functions, into a bit-vector of a few bits being set. The locations of the set bits are determined by the hash functions. A query of a key to a BF checks if all the hashed bits of the key are set, which indicates if the key is contained in the BF. Using BFs to represent users' interests consumes far less memory than traditional methods [5]. Additonally, content matching through querying to BFs is also fast and efficient.

Specifically, we propose an extension to the BF, which is called *Temporal Counting Bloom Filter* (TCBF). The TCBF is able to handle temporally related operations using a concept called *decaying*, which is crucial to embed routing information in the TCBF. B-SUB uses TCBFs to encode users' interests and embed information needed for brokers to make forwarding decisions. So, in B-SUB, the interests propagation is merely the exchanges of nodes' TCBFs; and the content matching becomes a query to TCBFs. These designs significantly reduce B-SUB's computational complexity and communication overhead, which is particularly important for resource-constrained HUNETs.

An issue that appears in using the TCBF is how to control its *false positives*. False positives occur because a key's hashed bits are accidentally set by other keys that have already been put into the TCBF. Because of false positives, B-SUB may falsely inject useless messages into the network. We analyze, in theory, several parameters that are related to the false positive probability and their impacts on B-SUB's performance. The analysis is verified through extensive simulation studies.

Our contributions in this paper are as follows:

- We design B-SUB, a practical content-based pub-sub system for HUNETs featuring low complexity and overhead. B-SUB's broker allocation scheme can exploit the social structures in HUNETs. The pub-sub forwarding in B-SUB relies on a novel data structure called TCBF.
- We propose a method to perform content-based networking using the TCBF. The TCBF is an extension to the classic BF, which supports temporal operations that are not possible in the classic BF. Using TCBFs reduces the storage for representing content, and the complexity in content matching. Besides, the TCBF is able to embed routing information, which enables efficient pub-sub forwarding. We analyze the relationships between the B-SUB's parameters and the false positive rate of the TCBF.
- We conduct extensive simulation studies using real-world human contact traces. The results verify our analysis, and demonstrate the usefulness and applicability of the BF/TCBF for practical content-based pub-sub communications in HUNETs.

The rest of this paper is organized as follows: Section II introduces relevant previous work. A preliminary of the BF is given in Section III. The TCBF is defined in Section IV. The design and analysis of B-SUB are presented in Section V and VI. Simulation evaluations are presented in Section VII. The conclusion is given in Section VIII.

## II. Related work

### A. HUNET/DTN routing

General DTN routing has been an active topic in the past few years. DTNRG [7] framed the concept of the DTN, and inaugurated the research on DTN routing [19]. Recently, the routing of HUNETs has drawn much attention. In our previous work [24], we proposed an optimal forwarding rule based on the optimal stopping theory and the long-term relationships between users. Some recent studies [9], [12], [28] using real-world mobility/contact traces reveal that DTNs show certain social network properties. For example, two important metrics, familiarity and centrality, are measured based on nodes' direct or indirect observed contacts, and used to guide the forwarding in [12]. However, community/social-based schemes, such as [10], [17], [23], [29], need complex on-/off-line processing, and may not perform well in HUNETs because the networks are so dynamic, resulting in community structures becoming highly versatile and difficult to capture. Several routing protocols explicitly considered the social mobility and metrics to facilitate efficient forwarding [8], [16]. However, all of these protocols leave content-based networking untouched, which limits their use in HUNETs.



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.

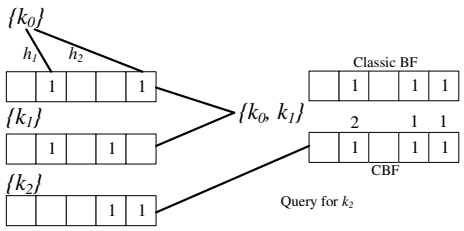



Fig. 2. Examples of the classic BF/CBF, their operations, and their false-positives. We use 5-bit vectors and 2 hash functions. A query for $k_2$ returns *true* since its bits are set by $\{k_0, k_1\}$, which is a false-positive.

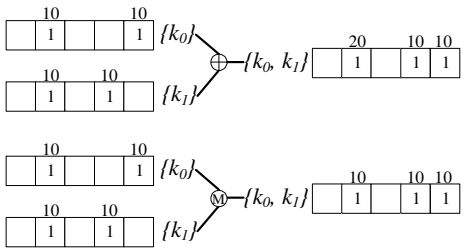



Fig. 3. Examples of the TCBF's A-&M-merge operations. The counters' initial value is 10. We use 5-bit vectors and 2 hash functions. Note that the counters' values of the resultant TCBFs in A-&M-merge are different.

### B. Content-based pub-sub system

Pub-sub is a powerful paradigm in solving communication problems [14]. The use of pub-sub systems in wireless networks has recently attracted interests [11], [25]. However, they do not consider unique properties of HUNETs.

Content-based networking has recently been used in opportunistic data diffusion [6]. Content-based addressing and routing can be implemented in many different ways [5]. In this paper, we use keys (raw strings) to represent content, which is popular in social networking applications. Our work also draws inspiration from several recent developments of pub-sub systems for DTNs [27].

### C. Network applications of the Bloom filter

The Bloom filter (BF) [3] enables a trade-off between space complexity and query accuracy, which is proven to be useful in many network applications [4]. The use of the bloom-filter in pub-sub systems can be found in various literatures [20], [21]. In [20], the BF is used to encode efficient interest predicates. In [21], the BF is used to encode addresses. On the contrary, we use the BF to represent interests in B-SUB. Our method is much simpler than that of [20]. To the best of our knowledge, this paper is the only work that uses the BF in the pub-sub systems in DTNs/HUNETs.

## III. Preliminary of the Bloom filter

The BF is a randomized data structure for representing sets, which supports probabilistic membership querying. A BF for a single key is a $m$-bit vector with $k$ bits being set. The locations of the set bits are determined by $k$ hash functions, each of which independently hash the key to an integer in $[0, m-1]$. A BF for a set of keys is obtained by sequentially inserting keys into the filter. To merge multiple BFs, we do a bit-wise OR on them.

The basic BF does not support deletions since we are unable to trace the associated keys of set bits. The Counting Bloom Filter (CBF) [22] is proposed to provide deletion. In a CBF, each bit is associated with a counter, which represent the number of keys that are associated with it. To delete a key from a CBF, we decrement the counters of the key's hashed bits. A bit will be reset once its counter reaches 0.

For a single key, the query to check if it is in a filter is performed by checking if all of its associated bits are set in the filter. A BF guarantees that a query for a key that is actually in the set, returns true. But, the BF has false positives, which means that queries for keys that are not in the set can also return true.

We first look at the expression of the false positive probability for a BF of $m$ bits, $k$ hash functions, and $n$ elements. It equals the probability for a message that is not contained in the filter, and whose $k$ hashed bits are set. The probability that a bit is not set by these $n$ keys in the filter is $(1-\frac{1}{m})^{nk}$. Since the locations of the $k$ bits of a key are independent, the probability that these $k$ bits are accidentally set by the $n$ keys actually contained in the filter is:

$$f = (1-(1-\frac{1}{m})^{nk})^k \approx (1-e^{\frac{-nk}{m}})^k \qquad (1)$$

The approximation is derived because $m$, the length of the bit-vector, is quite large. Since this probability is independent of the keys in queries, it is therefore termed the *false positive rate* (FPR). It is obvious that the FPR increases with the number of stored elements. Another concept, called *Fill ratio* (FR), is used to measure the number of elements in a BF. It is the ratio of the number of set bits to the length of the bit-vector. For the above settings, the number of set bits is:

$$\alpha = m\left(1-(1-\frac{1}{m})^{nk}\right) \approx m\left(1-e^{-\frac{kn}{m}}\right) \qquad (2)$$

The FR, and its approximation, are obtained as follows:

$$FR = \frac{\alpha}{m} = \frac{m\left(1-(1-\frac{1}{m})^{nk}\right)}{m} \approx 1-e^{-\frac{kn}{m}} \qquad (3)$$

## IV. Temporal counting bloom filter

### A. Definition and operations

The Temporal Counting Bloom Filter (TCBF) is an extension to the BF/CBF. Like a CBF, a TCBF associates each bit that has been set with a counter, as depicted in Fig. 3. But, these counters are not for representing the number of associated keys. Their uses are described below. For the convenice of discussion, we use the three definitions: BF, CBF, and/or TCBF, when appropriate.

The *insertion* and *merge* are different for the TCBF. Each time a key is inserted into a TCBF, the counters associated with the key's hashed bits will be set to an initial value $\mathbb{I}$. If



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.

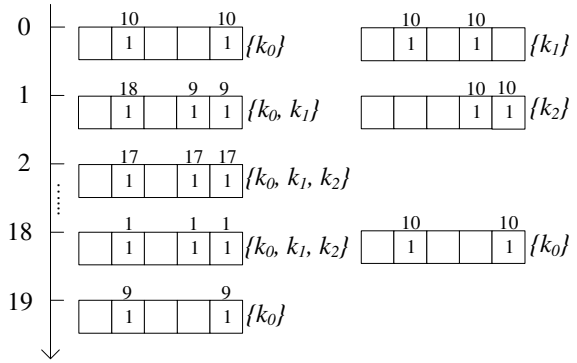


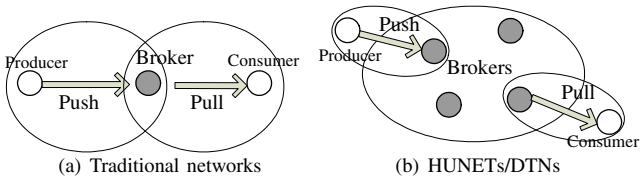

Fig. 4. The concept of the TCBF's decaying. We use 5-bit vectors and 2 hash functions. The DF is $1/unit\ time$. We can see that the counters get decremented with time. The only element left after time 19 is $k_0$.

the counter has already been set, we do not change its value. In other words, the results of insertions are always a TCBF with identical counters of a value of $\mathbb{I}$.

To merge two TCBFs, we create a new bit-vector by OR the bit-vectors of the two original filters. Two types of operations apply afterwards. First, the values of the new filter's counters are set as the maximum value of the counters of the two original filters. We call this *M-merge*, which is short for Maximum-merge. The other one is called *A-merge*, which stands for Additive-merge, where the counters' values are the sum of the values of the two original filters. These two types of operations are depicted in Fig. 3. The intention of these two different operations will be clear after we present the design of B-SUB in Section V. We can only insert a key into a filter that has never been merged before. In order to insert multiple keys into a merged filter, we first insert the keys into an empty TCBF, then merge the two TCBFs using A/M-merge.

The TCBF does not support direct deletion of elements. It only supports *temporal deletion*. That is, a filter constantly decrements the counters' values of all its set bits, which is called *decaying*. The rate that counters are decremented is called *decaying factor* (DF). If a key is not inserted into the filter frequently enough, it will be removed from the filter. The DF is a tunable parameter, directly influencing the performance of message forwarding. The relation between the DF and the content-based forwarding is discussed in Section VI. The concept of the TCBF's decaying is depicted in Fig. 4, where several keys are put into the filter at different times. The final structure of the filter represents the frequency of the key being inserted into the filter. We see that after 19 units of time, the only key left in the filter is $k_0$.

The query that checks if a key is in a TCBF is called *existential query*. The TCBF bears the same FPR as the classic BF, for existential queries. A difference is that the TCBF's FPR tends to decrease with the time because elements get removed after a certain amount of time. Another type of query for the TCBF is called *preferential query*. For a key $k$ and two filters $F_i$ and $F_j$, we get the counters of $k$'s associated bits in $F_i$ and $F_j$, which is two sets denoted $C_i$ and $C_j$. Then, we get the minimum value in $C_i$ and $C_j$, which is denoted $c_i$ and $c_j$. We define the *preference* of $F_j$ to $F_i$ against $k$, $PREF_{i,j}(k)$,




(a) Traditional networks (b) HUNETs/DTNs

Fig. 5. The concepts of the pub-sub system in traditional networks and HUNETs/DTNs. The "central broker" abstraction is not valid, or too expensive to maintain, in HUNETs/DTNs.

as the following equation:

$$PREF_{i,j}\{k\} = \begin{cases} \frac{c_j - c_i}{c_i} & \text{if } c_i \neq 0 \\ c_j & \text{if } c_i = 0 \end{cases}$$

Note that the preference is $c_j$ when $c_i$ equals 0.

### B. The advantages of the TCBF

TCBFs are used to represent interests in B-SUB. Basically, TCBFs map, through hashing $h_i$ $(1 \leq i \leq k)$, each interest into a bit-vector with $k$ bits being set using $k$ hash functions. First, using TCBF reduces storage for representing interests. The analysis in Section VII demonstrates that the TCBF uses half of the space used by the raw strings in representing interests. It also reduces bandwidth requirements in interests propagation. When two interests $I_0$ and $I_1$ of a user cannot be forwarded at the same time due to the bandwidth restriction in the contact, only one of them gets the opportunity to be served. If we can compress both $I_0$ and $I_1$ to fit the bandwidth, both interests might eventually be served. The content matching using TCBF is also more efficient than the string matching method [1].

Using the TCBF's decaying, we are able to more accurately measure the entering/exiting of a particular interest through addition/subtraction of counters. The challenge is to set appropriate parameters so that the gain (by compressing more interests) outweighs the loss of effort in handling false positives. In-depth study is needed on various trade-offs in selecting parameters, such as the number of interests that can be stored, and the length of the bit-vector.

## V. The Design of B-SUB

We present the design of B-SUB in this section. We show that B-SUB is simple and easy to implement due to its extensive use of the TCBF.

### A. Overview

We now describe some concepts used in B-SUB. The concept of pub-sub in a HUNET/DTN is depicted in Fig. 5. Fig. 5(a) shows the traditional pub-sub system, where a central broker exists, connecting message *producers* and *consumers*. Fig. 5(b) depicts the pub-sub systems in HUNETs/DTNs. We can logically treat a swarm of brokers as a central broker.

The content of a message is identified by a single *key*, which is a string that indicates the content of the message. It is desirable to use multiple keys to describe a message. However, we limit our scope in this paper for simplicity and



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.

ease of representation. Also, it is straightforward to extend the analysis to multi-key descriptions' cases. Users' interests are also represented by keys, and are stored in TCBFs. TCBFs are then used as probabilistic hints for the forwarding of messages.

Our system logically contains two components: *broker allocation* and *pub-sub forwarding*. The pub-sub forwarding component can be further separated into two parts: *interests propagation* and *message forwarding*. A two-tier structure is formed by B-SUB, where only brokers are responsible for collecting subscriptions and forwarding messages. A fact of this scheme is that it puts unbalanced burden on brokers. However, this is inevitable since it is beneficial to let more active nodes perform forwarding in the network, which has already been identified by past researches [15], [16], [17], [29]. In reality, it is applicable in practice too. When considering the social networking applications, it is natural that there are some altruistic users, meaning that they are willing to contribute their resources without any incentives. Besides, some users may be willing to contribute their resources in order to gain more popularity.

The sizes of the messages in B-SUB are small, which are in order of hundreds of bytes. This assumption is true in social networking applications. For example, Twitter [18], a popular micro-blogging application, requires a maximum size of 140 bytes for each post. If a message is wrongly injected into the network, the wasted bandwidth is acceptable. This is why the BF actually pays off, despite its false positives.

The storage of the TCBF is space-efficient, and its operations are of low complexity; as depicted in Section IV. B-SUB uses TCBFs for almost all of its functionalities, it therefore requires less memory and low computational power.

### B. Broker allocation

In our previous work [30], we have designed a decentralized broker allocation algorithm for HUNETs. The allocation method considers social characteristics, and features low complexity and overhead.

In B-SUB, we extend the previous scheme. B-SUB's broker allocation method is similar to an election. Basically, each user (producer or consumer) has three thresholds: a lower bound $low$, a upper bound $up$, and a time window $T_w$. For each user, if the number of brokers it meets in $T_w$ drops below $low$, it will designate the next nodes to be a broker. On the contrary, if the number exceeds $up$, it tries to designate the next broker to be a normal node. Brokers themselves do not perform these operations.

In order to select the nodes that have a higher probability to reach other nodes in the network, a node forces the nodes that are less "popular" to be normal users. That is, when a user wants to designate a broker to become a user because the number of brokers it meets in $T_w$ exceeds $up$, it compares the broker's *degree* to other brokers' degrees. A node's degree is the number of different nodes that it meets in $T_w$. The user designates the broker to be a user if the broker's degree is below the average value, otherwise it does nothing. In this way, less popular nodes are more likely to be removed from the brokers set. The resultant brokers are more efficient in forwarding.

### C. Interests propagation

We use TCBFs to compress users' interests. A message consumer stores its own interests in a TCBF, which is called the *genuine filter*. A broker stores a TCBF for propagating other users' interests, which is called the *relay filter*. TCBFs serve as "compressed" matching hints for content delivery, which are not precise due to the TCBF's false positives. This occurs when scoped forwarding identifies a marching, and the content needed to be delivered is sent back to the consumers in multiple hops. A TCBF containing multiple interests, is stored along the forwarding path, which is formed by brokers. The reverse of the path can be found with the guidance of the stored bloom filters in the brokers. Again, false positives may occur. Thus, a delivery tree, instead of a path, will be generated. Nonetheless, it is guaranteed that the original path to the subscriber is embedded in the tree.

A node, or a mobile device carried by a person, can simultaneously be a producer, consumer, and broker. When two nodes meet, they first exchange identity information. Then, different operations are performed depending on the identities of the two nodes. When a consumer meets a broker, it forwards its genuine filter to the broker. The broker then merges this filter into its relay filter using A-merge. Next time, when the consumer meets the same broker, it can increase the counters' values that associated with its interests. In this way, a consumer actually *reinforces* its interests upon brokers. The more frequently a broker meets a consumer, the higher its counter's value of the consumer's interests, and the higher the probability that the broker is selected as the forwarder for the consumer because the broker can travel for a longer time carrying the consumer's interests. In other words, the decaying and reinforcement identify closely related broker-consumer pairs.

When two brokers meet, say $N_0$ and $N_1$, they first exchange their relay filters. $N_0$ or $N_1$ then combines its own relay filter and that of the other broker using M-merge. The resultant filter will replace the old one.The intention for performing a M-merge, instead of A-merge, in interests propagation between brokers is to prevent *bogus counters*. Fig. 6 illustrates how bogus counters are generated using A-merge. Two brokers, $B$ and $C$, meet frequently, but $B$ only meets with $A$ once-in-a-while. If brokers use A-merge in combining filters from other brokers, the counters of the same key will get added in a loop. Our goal is to remove the interests from users a broker meets infrequently, whereas the results here are contrary to this goal. As a result, $B$ and $C$ will be selected as the forwarders of the messages that $A$ is interested in, by $D$ and $E$. However, $B$ and $C$ are actually not desirable candidates because they are not closely realted to $A$.

### D. Message forwarding

The decaying of the TCBF removes, from the brokers' relay filters, the interests from the consumers that they meet



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.

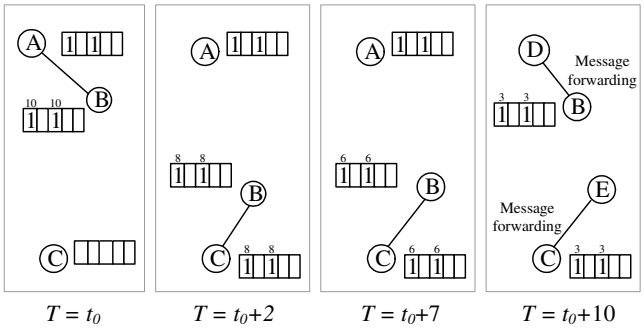



Fig. 6. Bogus counters are generated in A-merge due to the frequent contacts between brokers. Node $B$ only meets $A$ once. $B$ and $C$ obtain bogus counters from each other using A-merge because they meet frequently. Even after 10 units of time, $B$ and $C$ do not remove $A$'s interest. They are selected by $D$ and $E$ as the forwarders for the messages that A is interested in. however, $B$ or $C$ are unable to deliver those messages.

infrequently. The preferential query is used by brokers to select forwarders. All the data needed in forwarding messages is merely a TCBF. The operations performed are only hashing and table lookup. B-SUB is simple and efficient because of these designs.

When a producer meets a consumer, the consumer reports its interests in a BF (not TCBF) to the producer. The producer then queries all its messages against the filter, and forwards all the messages that match the filter, to the consumer.

Similar processes are performed where a broker meets a producer/consumer. When a broker meets a producer, it forwards a BF to the producer. The producer queries that filter and determines the events that need to be transmitted. This saves bandwidth. For the applications that we consider in this paper, this saving is not negligible. When a broker meets a consumer, the broker requests a BF containing the consumer's interests, then forwards the matched messages to the consumer. We reduce the communication overhead by ripping the counters from the TCBFs. This saving is not negligible in HUNETs. For a single message, at most $\mathbb{C}$ copies are replicated to $\mathbb{C}$ different brokers. The message is removed from the producer's memory after its copy number reaches the limit. The messages directly forwarded to consumers are not counted as copies. Messages' lifetime is controlled by their Time-To-Live (TTL) values, which are identical to their maximum tolerable delay. The TTL is counted since the message has been created.

When two brokers meet, they first exchange their relay filters. The two brokers, say $B_0$ and $B_1$, make message forwarding decisions before merging their relay filters. One of the two brokers $B_0$ and $B_1$, query the other's preference to itself against each message it carries. Those messages that have the largest positive preference are forwarded first. Messages are removed from brokers' memory after being forwarded. This is to prevent excessive copies in the network.

## VI. Analysis

The DF (Decaying Factor) is the key to adjusting B-SUB's behaviors. We consider the impact of the DF on interests propagation and the FPR. We also analyze the storage complexity of TCBFs and present a TCBF allocation method with optimal FPR.

### A. DF and interests propagation

If decaying is not used, that is, the counters of the set bits do not change after being set, then no interests will be removed. An obvious consequence is that a broker will end up with carrying the interests from the users that it meets infrequently. Bad forwarders may be selected as forwarders. Another consequence is that the FPR increases. This results in unnecessary traffic in the network, wasting devices' energy and bandwidth.

Another reason to use decaying is to enforce the messages timeliness. As stated before, we consider social networking applications in HUNETs. It is, therefore, reasonable to assume that a message is of no value if its delay is too long, since users can get the message through other connections. Suppose that each message has a delay limit of $\mathbb{T}$, we should set the DF in such a way that an interest will get removed after $\mathbb{T}$ since a consumer inserted the interest once. So, if a broker contains an interest, then that means that the broker has met a consumer that is interested in it, withing $\mathbb{T}$. If a message is forwarded by the broker, it is likely that the message will be delivered withing $\mathbb{T}$.

If the initial value of the counters is $\mathbb{I}$, according to the above analysis, the DF can be $\frac{\mathbb{I}}{\mathbb{T}}$. However, it is still possible that a key is not removed after $\mathbb{T}$ because its counters are accidentally incremented by some other keys. The counters can be incremented in two ways: *A-merge with filters from message producers* or *M-Merge with filters from brokers*. We only consider the first case due to the intractable complexity of the second one. We assume that each node has its own interests, the number of keys collected in $\mathbb{T}$ is obtained by accumulating all keys from the contacted nodes, denoted as $\mathbb{N}$. For one of the $k$ bits associated with a key and another key, the probability of the bit being accidentally incremented by the other key is $\frac{k}{m}$, that is, the other key has $k$ chances to be hashed to that bit location (we omit the probability that multiple hash functions return the same location). Overall, $\frac{k\mathbb{N}}{m}$ keys are hashed to the same bit.

The number of keys hashed to the location, $num$, is a binomial distribution with expectation $\frac{k\mathbb{N}}{m}$. Because a key is removed once and any one of its counters reaches 0, the key's lifetime is determined by the the minimum value of $\{num_0, ..., num_{k-1}\}$. Denote $F(x; \mathbb{N}, \frac{k}{m})$ the CDF of $num$, we get the expectation of $MIN\{num_0, ..., num_{k-1}\}$:

$$\widehat{MIN}\{num_0, ..., num_{n-1}\} = \prod_{x=1}^{\mathbb{N}} \{1 - F(x-1; \mathbb{N}, \frac{k}{m})^k - (1 - F(x; \mathbb{N}, \frac{k}{m})^k\} \quad (4)$$

Multiply the value in Eq. 4 with the counters' initial value $\mathbb{I}$, we get the expectation of the minimum incremental value of the key's counters: $\mathbb{I}\left(1 + \widehat{MIN}\{num_0, ..., num_{n-1}\}\right)$. Based on this result, the DF is set to Eq. 5, where a small



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.

constant $\triangle$ is added to account for the values in the second case.

$$DF = \frac{\mathbb{I}}{\mathbb{T}} + \frac{\widehat{MIN}\{num_0, ..., num_{n-1}\}\mathbb{I}}{\mathbb{T}} + \triangle \qquad (5)$$

### B. DF-FPR trade-offs

According to Eq. 5, $\mathbb{T}$ decreases when the $DF$ increases, which, in turn reduces $\mathbb{N}$ and $FPR$. However, the scope that interests can propagate is also reduced. It is, therefore, possible that interests are unable to reach the corresponding producers, which result in a lower delivery ratio. The point here is that we are able to control the FPR by adjusting the DF.

In practice, we can not get a close-form function of the DF and $\mathbb{T}$. However, we can tentatively adjust the DF, then re-adjust its value by observing the resultant FPR; until a desirable FPR is achieved.

If we know $\mathbb{T}$, since we have set the decaying factor so that the interests can get removed after $\mathbb{T}$, the number of the elements in the filter is determined by the number of keys, $\mathbb{N}$, that a broker collected within $\mathbb{T}$. Some interests may be duplicated. Suppose each producer has $\bar{k}$ keys, and there are $\mathbb{K}$ keys in total. The number of unique interests stored in a broker's filter is given by the following equation:

$$\widehat{\mathbb{N}} = \mathbb{N}\left(1 - \frac{1}{\mathbb{K}}\right)^{\mathbb{N}-\bar{k}} \qquad (6)$$

Then, we can calculate the FPR using Eq. 1. Alternatively, we can derive the number of elements using Eq. 2, Eq. 3, and the observed FR: $\widehat{\mathbb{N}} = \frac{m}{k} ln \frac{1}{1-FR}$. Combining Eq. 6 and Eq. 1, we obtain that the FPR is $(1 - e^{\frac{-k\widehat{\mathbb{N}}}{m}})^{\widehat{\mathbb{N}}}$, where $\widehat{\mathbb{N}}$ is given by Eq. 6, and $m$ is the length of the bit-vector.

Although these messages do waste bandwidth, not all of them get delivered to the consumers, because the matching is conducted at the final stage of the forwarding. The actual probability that a message, in which no consumers are interested, getting delivered to a consumer is $FPR^2$. These messages are considered completely wasted. On the other hand, even falsely injected messages can be delivered to users that are really interested in them, which is not considered completely wasted. Their ratio is $FPR \times (1 - FPR)$.

### C. Memory consumption

A TCBF has a bit-vector and a number of counters. For a bit-vector of length $m$, instead of using a raw bit-vector that has $m$ bits, we merely need to record the locations of the set bits. Each location needs $\lceil log_2 m \rceil$ bits. The overall size is $\alpha \times \lceil log_2 m \rceil$. This method is better only when $\alpha \times \lceil log_2 m \rceil < m$, that is, $\alpha < \frac{m}{\lceil log_2 m \rceil}$. $\alpha$ is given in Eq. 2. We use a 1-byte counter. Usually, 24 hours will be enough for delay. The best granularity is 5.6 minutes in this case. The memory consumption of the counters is therefore $\alpha$ bytes. Thus, the expected total size of a filter, in this case, is: $\alpha \times (1 + \lceil log_2 m \rceil)$.

The condition, $\alpha \times \lceil log_2 m \rceil < m$, in the above equation is usually met because the FR is low. A few optimizations are possible. If all the counters of a filter are identical, we merely save one value, which cuts the size to $\alpha + \lceil log_2 m \rceil$ bytes. When a broker requests messages from a source, it does not need to report the counters, which cuts the size to $\alpha$ bytes. If we use raw strings, instead of TCBFs, to represent interests and the memory consumption is obtained by summing up the memory space of all the interests, and the associated control information. Later, in Section VII, we discuss, in detail, the keys used in the simulation, and the comparison of the memory consumptions of two approaches.

### D. TCBF allocation for optimal FPR

We consider a dynamic TCBF allocation strategy, where a new TCBF is allocated when the fill ratio (FR) of the current filter(s) exceeds a threshold $\tau$.

As discussed in Section VI-B, we can derive the number of unique keys in a TCBF using Eq. 2 and Eq. 3. Then, we can get the FPR of the TCBF using Eq. 1.Before we step into the allocation strategy, let's consider the FPR of a collection of $h$ BFs $\{F_0, ..., F_{h-1}\}$ to represent a single set of $n$ elements. For a query against a key for $h$ filters, the joint FPR is the complementary probability that all $h$ filters return correct responses, which equals $\left(1 - \prod_{i=0}^{h-1}(1 - FPR_i)\right)$. $FPR_i$ is determined by Eq. 1, thus $FPR_{joint}$ is given by the following equation:

$$FPR_{joint} = 1 - \prod_{i=0}^{h-1}\left(1 - (1 - e^{\frac{-n_i k}{m}})^k\right) \qquad (7)$$

where:

$$\sum_{i=0}^{h-1} n_i = n$$

Each one's memory consumption in the $h$ filters is $\alpha \times (1 + \lceil log_2 m \rceil)$. The total memory of $h$ TCBFs is given by:

$$\begin{aligned}\mathbb{S} &= \sum_{i=0}^{h-1}\left\{\left[1 - \left(1 - \frac{1}{m}\right)^{kn_i}\right] m(1 + \lceil log_2 m \rceil)\right\} \\ &= m(1 + \lceil log_2 m \rceil)\sum_{i=0}^{h-1}\left(1 - e^{-\frac{kn_i}{m}}\right) \end{aligned} \qquad (8)$$

Eq. 8 is obtained because $m$ is quite large. Now, we return to the problem of how to choose the optimal parameters. Given an upper bound of storage $\mathbb{S}_{max}$, we are to determine what are the parameters for the minimum FPR:

$$\begin{aligned}\text{Minimize: } & FPR_{joint} \\ \text{Satisfy: } & \mathbb{S} < \mathbb{S}_{max}\end{aligned} \qquad (9)$$

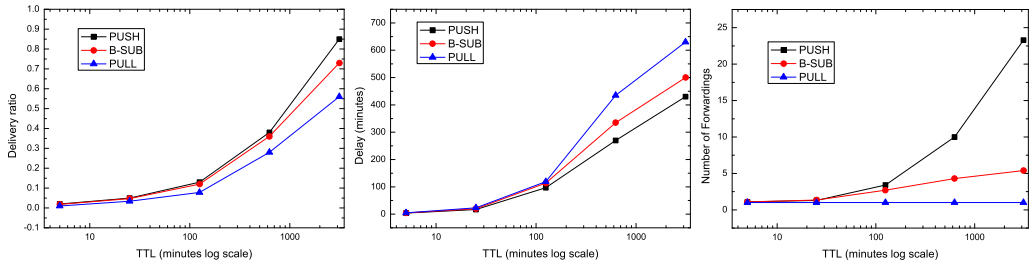
It is clear that $FPR_{joint}$ achieves the maximum value when $n_i = \frac{n}{h}$ for all $i \in \{0, ..., h-1\}$. We rewrite Eq. 9 as follows:

$$\begin{aligned}\text{Maximize: } & \left(1 - (1 - e^{-\frac{nk}{hm}})^k\right)^h \\ \text{Satisfy: } & h\left(1 - e^{-\frac{nk}{hm}}\right) < \frac{\mathbb{S}_{max}}{m(1 + \lceil log_2 m \rceil)}\end{aligned} \qquad (10)$$

Eq. 10 is too complicated to solve efficiently. We reduce its complexity by fixing $k$ and $m$. Here, $n$ is a constant that can



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.




(a) Delivery ratio (b) Delay (c) Number of forwardings

Fig. 7. The delivery ratio, delay, and number of forwardings per delivered message of PUSH, B-SUB, and PULL in Haggle(Infocom06) trace.

be measured in simulations. The problem now is to find the optimal $h$. Because $FPR_{joint}$ and $\mathbb{S}$ are monotone increasing functions of $h$, and $\mathbb{S}$ has a upper bound, the minimum $FPR_{joint}$ is achieved when $h$ reaches its maximum value. Since $h$ is an integer, its maximum value can be found by a binary search. The maximum number of keys in a signle TCBF is then obtained; and the corresponding FR calculated using Eq. 3 is set as the threshold $\tau$.

## VII. Simulation Evaluation

### A. Simulation settings

We consider three metrics in our simulation: *delivery ratio*, *delay*, and *overhead*. We compared B-SUB with two other techniques: *PUSH, PULL*. In the PUSH, a node replicates an event it stores to every node it encounters that has not received a copy. In the PULL, a node only collects messages that it is interested in from its directly encountered neighbors.

We use two real-world human contact traces in simulations: MIT reality [13] and Haggle (Infocom06) [26]. These two datasets are obtained by logging the bluetooth contact between portable devices. The characteristics of these two data sets are given in Table I. These two data sets have a moderate number of nodes, which are the environments we designed for. We use the entire Haggle trace obtained from the Infocom '06 conference, and the 3 day records from the MIT Reality trace, in the simulation.

Since the nodes in HUNETs are used by people, all nodes should have their own interests and generate messages. We assume that each node is interested in only one key. Each node has a fixed message generation rate $\mathbb{R}$. This rate is constant for each node, which is determined by its social standing. We use centrality to measure the social standing. The higher the centrality, the higher the message generation rate. Denote the minimum message rate $\hat{\mathbb{R}}$ for the smallest centrality $\hat{\mathbb{C}}$ and the message generation rate $\mathbb{R}$ for a node with a centrality $\mathbb{C}$ is $\mathbb{R} = \frac{\mathbb{C}\hat{\mathbb{R}}}{\hat{\mathbb{C}}}$. $\hat{\mathbb{R}}$ is set to $\frac{1}{30} message/minute$. $\mathbb{I}$ is set to 50.

| Data Set | Haggle(Infocom'06) | MIT reality |
|---|---|---|
| Device | iMote | phone |
| Communication method | Bluetooth | Bluetooth |
| Duration (days) | 3 | 246 |
| Number of nodes | 79 | 97 |
| Number of contacts | 67,360 | 54,667 |

TABLE I
Parameters of two data sets

| NewMoon | Twitter'sNew | funnybutnotcool | openwebawards |
|---|---|---|---|
| 0.132 | 0.103 | 0.0887 | 0.0739 |

TABLE II
The distribution of the top 4 keys in the simulation (spaces are removed).

We prepared 38 keys from the Twitter Trend search engine [18] in one week (from 16th to 22nd Nov. 2009). The probability of each key being selected as an interest for each node is determined by the key's weight in the original Twitter Trend, which is obtained by querying the Twitter API [18]. Messages have a maximum size of 140 bytes, which is the same as the Twitter posts. We assume that messages' length is uniformly distributed in $[1, 140]$. We list in Table II the probability of the top 4 keys in the simulation. The average length of the keys is 11.5 bytes. We use a bit-vector of 256 bits and 4 hash functions, thus, at most, 5 bytes are used to encode a single key. If the additional control information needed in using raw strings is considered, the space complexity of using TCBFs is quite desirable. The worst case FPR of the filter storing 38 keys, in theory, in this setting, is 0.04. The bandwidth of the wireless channel is 1Mbps, which is the peak value of Bluetooth devices. Wireless transmission errors are not considered. It is well-known that a wireless channel offers far less bandwidth than its claimed peak value. We assume that the average transmission rate is 250Kbps. The durations of all the contacts are already recorded in the trace. The maximum number of copies that can be forwarded by producers $\mathbb{C}$ is 3. The broker allocation threshold is 3 and 5, which maintains about 30% of the nodes being brokers in two traces. The time window is 5 hours.

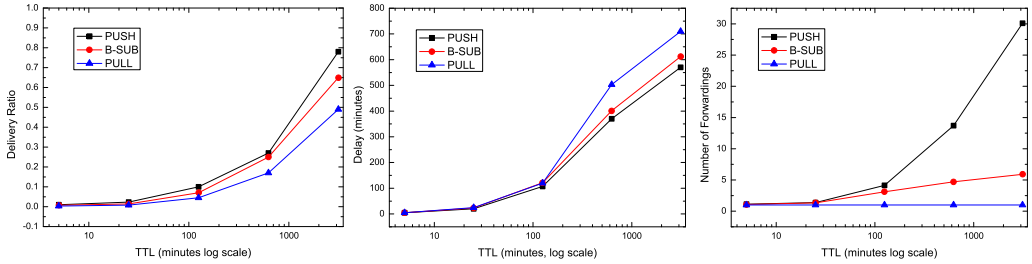
### B. Delivery ratio

We provide the results of the delivery ratio under different Time-To-Live (TTL) values. In order to compare with PUSH and PULL in the same setting, we set $\mathbb{T}$ the same as the TTL, and calculate DFs using Eq. 5. A small constant is added to the resultant DFs to account for the missed cases in Eq. 5.

Fig. 7(a) and Fig. 8(a) present the delivery ratio of three techniques in two traces. PUSH essentially floods messages in the network, so its delivery ratio indicates the best results we can achieve. B-SUB is only slightly lower than PUSH.



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.




(a) Delivery ratio (b) Delay (c) Number of forwardings

Fig. 8. The delivery ratio, delay, and number of forwardings per delivered message of PUSH, B-SUB, and PULL in MIT Reality trace.

The DF of B-SUB is calculated using Eq. 5. The number of encountered nodes in $\mathbb{T}$ is obtained by analyzing the traces. However, it is straightforward to set an appropriate DF online by counting the number of nodes a broker meets in the time window. On the contrary, PULL is the most conservative. It only performs one-hop forwarding. Due to the limited diameters in the network, the delivery ratio of PULL is not far from PUSH and B-SUB. The results in MIT Reality trace have the similar trend. Overall, the MIT Reality trace forms a sparser network, and its contact frequencies are lower than that in Haggle trace. So, the delivery ratio in the MIT Reality trace is lower.

We present the changing of the delivery ratio with the DF in Fig. 9(a). The TTL is set to 20 hours. The DF for $\mathbb{T} = 10$hours is set to $0.138/min$, or decremented by 1 every 7.2 minutes, which is obtained by counting the number of different nodes met in 10 hours. When the DF is 0, it means that we do not reduce counters' values. Thus, the messages just get flooded into the network. The delivery ratio is only limited by the messages' TTL values. Higher DF values reduce the scope of the interests propagation, which in turn reduces the delivery ratio. Fig. 9(a) echoes with the results in Fig. 7 when DF = 0.138, which is the DF for TTL = 10 hours. Note that the DF does not limit the messages' life time, only limits the scope of interests propagation. The DF is effective in reducing unnecessary transmission, which will be discussed later.

### C. Delay

We only consider the delay of delivered messages, which is measured as the time interval from the time when the messages are generated to the time they are delivered. The results are given in Fig. 7(b) and Fig. 8(b). Again, PUSH shows the best performance. B-SUB has similar results. The delay of PULL is considerably larger than B-SUB and PUSH. Since we use a $log_{10}$ scaled time-axis, the changing of delay with the TTL is actually quite slow. This is because most of the messages can be delivered within a relatively short time. Thus, the messages being delivered because of increased TTLs, which have much larger delays, do not increase delay siganificantly.

The changing of delay with the DF in two traces is given in Fig. 9(b). Because the DF limits the scope of interests propagation, the delay also decreases with the DF. And due to the two traces' properties, the results in the Haggle (Infocom 06) are a little better than the results in the MIT Reality trace. Again, when DF = 0, the delay is only limited by the TTL, which is set to 20 hours.

### D. Overhead

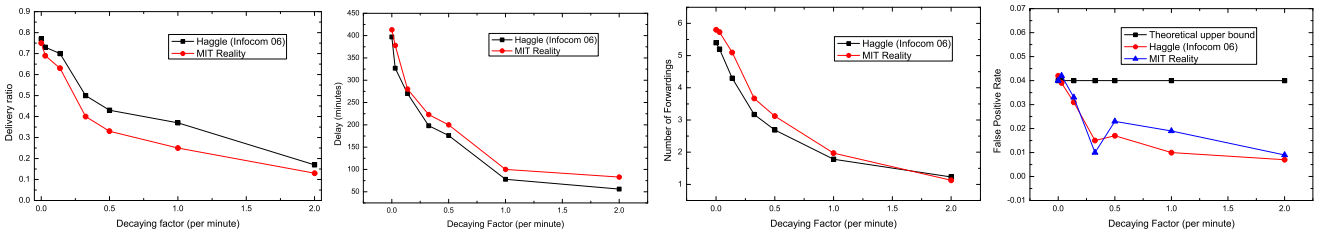
We first look at the number of forwardings for each delivered message. The results are obtained by dividing the number of forwardings in the network by the number of messages that have been delivered. The results are presented in Fig. 7(c) and Fig. 8(c). PUSH has the most forwarding numbers. B-SUB is able to maintain a relatively stable forwarding counts since it uses decaying to limit the scope of interests propagation, and the TCBF's preferential query to limit the forwarding counts. PULL actually has the best performance because it is the most conservative protocol. However, PULL's delivery ratio and delay are much worse than B-SUB.

The changing curve of the number of forwardings with the DF is given in Fig. 9(c). The TTL of messages is 20 hours. Because we are able to reduce the scope of interests propagation by using a higher DF, the number of forwardings decrease with the DF. This is because: 1) the scope of message forwarding to interested users is reduced, which means that "distant" producers tend to not forward packets; 2) the brokers are more aggressive in selecting users in forwarding, since only users that frequently meet will get opportunities to propagate their interests to other nodes in the network. When the DF is too large, the delivery ratio is quite low. In this situation, B-SUB works like PULL, where only very close producers get the chance to forward their messages. Thus, the average number of forwardings reduces to nearly 1.

We then look at the FPR of the delivered messages. The ratio of the number of falsely delivered messages to the total number of delivered messages is computed as the FPR. We present the results in Fig. 9(d). As analyzed before, the worst case FPR, in theory, is 0.04 in our settings. In practice, the FPR can be much lower than this value. This is because the elements stored in each TCBF are usually quite less than the maximum number of keys. The FPR reaches the largest value for the DF = 0, where brokers are highly likely to collect the maximum number of elements. Additionally, due to the uneven distribution of the keys, the FPR can actually be larger than the maximum theoretical value. When the DF increases, the elements stored in brokers' TCBFs decreases, along with the FPR. The results of B-SUB's false positives show that it is practical to achieve adequate performance using much less storage with the TCBF.



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.




(a) Delivery ratio (b) Dealy (c) Number of forwadings (d) False positive

Fig. 9. The changing curve of the four metrics with the decaying factor.

## VIII. Conclusion and Future Work

In this paper, we proposed B-SUB, a content-based pub-sub system for HUNETs. B-SUB uses a novel data structure called, Temporal Counting Bloom Filter, to perform content-based networking tasks. Using the TCBF reduces memory and bandwidth consumption in storing and propagating users' interests, which makes B-SUB simple and efficient. We systematically analyze the impact of several parameters of B-SUB on its behaviours and performance. Extensive real-world-trace-based simulations are conducted to verify B-SUB's performance. The results have proven that B-SUB has a similar delivery ratio, and delay, to the optimal method (PUSH), whereas B-SUB consumes much less resources than PUSH.

However, more simulation studies are needed to fully justify B-SUB's design. With the development of wireless communication technologies, HUNETs are becoming a promising communication platform for the future wireless applications. A prototype HUNET system will be our future work.

## Acknowledgement

This work is supported in part by NSF grants CNS 0422762, CNS 0626240, CCF 0830289, and CNS 0948184.

## References

[1] Ioannis Aekaterinidis and Peter Triantafillou. Substring matching in p2p publish/subscribe data management networks. *Data Engineering, International Conference on*, 0:1390–1394, 2007.

[2] BBC. New mobile message craze spreads. http://news.bbc.co.uk/2/hi/technology/3237755.stm.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. of ACM*, 13(7):422–426, 1970.

[4] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.

[5] Antonio C., David S. R., and Alexander L. W. Content-based addressing and routing: A general model and its application, 2000.

[6] I. Carreras, De P. Francesco, D. Miorandi, D. Tacconi, and I. Chlamtac. Why neighborhood matters: interests-driven opportunistic data diffusion schemes. In *Proc. of CHANTS '08*, pages 81–88, New York, NY, USA, 2008. ACM.

[7] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay tolerant networking architecture. Internet draft, draft-irrf-dtnrg-arch.txt, DTN Research Group.

[8] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. *IEEE Journal on Selected Areas in Communications*, 26(5):748–760, Feburary 2005.

[9] A. Chaintreau, P. Hui, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007. Fellow-Crowcroft, Jon.

[10] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 26(5):748–760, June 2008.

[11] G. Cugola, A. L. Murphy, and Gian P. Picco. Content-based Publish-subscribe in a Mobile Environment. In Paolo Bellavista and Antonio Corradi, editors, *Mobile Middleware*, pages 257–285. Auerbach Publications, 2006. Invited contribution.

[12] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proc. of MobiHoc '07*, pages 32–40, New York, NY, USA, 2007. ACM.

[13] N. Eagle and A. (Sandy) Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[14] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[15] W. Gao, Q. Li, B. Zhao, and G. Cao. Multicasting in delay tolerant networks: a social network perspective. In *Proc. of MobiHoc '09*, pages 299–308, New York, NY, USA, May 2009. ACM.

[16] P. Hui and J. Crowcroft. How small labels create big improvements. In *Proc. of PERCOMW '07*, pages 65–70, Washington, DC, USA, 2007. IEEE Computer Society.

[17] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proc. of MobiHoc '08*, pages 241–250, New York, NY, USA, 2008. ACM.

[18] Twitter Inc. Twitter. http://www.twitter.com.

[19] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. of ACM SIGCOMM '04*, pages 145–158, New York, NY, USA, 2004. ACM.

[20] Z. Jerzak and C. Fetzer. Bloom filter based routing for content-based publish/subscribe.

[21] P. Jokela, András Zahemszky, Christian E. Rothenberg, S. Arianfar, and P. Nikander. Lipsin: line speed publish/subscribe inter-networking. In *Proc. of SIGCOMM '09*.

[22] F. Li, Pei C., J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *Networking, IEEE/ACM Transactions on*, 8(3):281–293, Jun 2000.

[23] F. Li and J. Wu. Mops: Providing content-based service in disruption-tolerant networks. *Proc. of ICDCS '09*, 0:526–533, 2009.

[24] C. Liu and J. Wu. An optimal probabilistic forwarding protocolin delay tolerant networks. In *MobiHoc '09: Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 105–114, New York, NY, USA, 2009. ACM.

[25] T. Pongthawornkamol, K. Nahrstedt, and G. Wang. The analysis of publish/subscribe systems over mobile wireless ad hoc networks. In *Proc. of ACM MobiQuitous '07*, pages 1–8, Aug. 2007.

[26] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD data set cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[27] G. Sollazzo, M. Musolesi, and C. Mascolo. Taco-dtn: a time-aware content-based dissemination system for delay tolerant networks. In *Proc. of ACM MobiOpp '07*, pages 83–90, New York, NY, USA, 2007. ACM.

[28] V. Srinivasan, M. Motani, and Wei T. Ooi. Analysis and implications of student contact patterns derived from campus schedules. In *Proc. of ACM MobiCom '06*, pages 86–97. ACM, 2006.

[29] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *Proc. of MSWiM '07*.

[30] Y. Zhao and J. Wu. Socially-aware publish/subscribe system for human networks. In submission. Available at: http://www.cis.temple.edu/˜wu/pubsub.pdf.



Authorized licensed use limited to: Temple University. Downloaded on August 11,2010 at 17:30:08 UTC from IEEE Xplore. Restrictions apply.