BRVST: Efficient and Content-Expressive Information Matching Overlay in Wireless Networks

Ying Li and Xin Wang
Department of Electrical and Computer Engineering
Stony Brook University
Email: {yingli, xwang}@ece.sunysb.edu

Abstract—**Efficient and flexible information matching over wireless networks has become increasingly important and challenging with the popularity of smart devices and the growth of social-network-based applications. Some existing approaches designed for wired networks are not applicable to wireless networks, due to their overwhelming control overheads. In this paper, we propose a reliable and scalable binary range vector summary tree (BRVST) infrastructure for flexible information expression support, effective content matching and timely information dissemination over the dynamic wireless network. A novel attribute range vector structure has been introduced for efficient and accurate content representation and a summary tree structure to facilitate information aggregation. For robust and scalable operations over dynamic wireless network, the proposed overlay system exploits a virtual hierarchical geographic management framework. Extensive simulations demonstrate that BRVST has a significantly faster event matching speed, while incurs very low storage and traffic overhead, as compared with peer schemes tested.**

I. Introduction

With the drastic growth of social and wireless application information data generated and consumed, it is emergent to establish a bridge infrastructure that can timely and accurately discovers and delivers the information to various parties of interests.

As an example of new era information service, a smartphone user in a downtown block wants to obtain a recommendation for some restaurants while people close-by may be also searching for the same type of information. Another user just stepping out of a Thai cuisine is satisfied with the dining experience and would like to share this place with others. Other applications include traffic information posting and retrieval where users cooperatively contribute to and benefit from the real-time traffic reports.

These applications can be better met by a "contribute-and-benefit" pattern system. Publish/Subscribe (Pub/Sub) system is one of this type, in which subscribers specify their interests and publishers post advertisements. The system matches subscriptions with publications. Unlike client/server models, the Pub/Sub model decouples time, space, and flow between publishers and subscribers to provide flexibility in information distribution.

Gryphon [1] and SIENA [2] were once popular Pub/Sub models in wire-line networks, however, their tree-based structure are not scalable in dynamic wireless network whose topology may constant change due to mobility and connection broken.

Many later attempts have been made to apply Pub/Sub infrastructure for wireless networks [3] [4] [5], where the information in the systems is roughly divided into several basic types. These platforms cannot efficiently support heterogeneous user application needs.

Different from conventional Pub/Sub systems which mainly categorize information into a few types for ease of implementation, the modern information system is expected to better meet the customized information needs of individual users. Besides the difference in categories, the heterogeneity of information is more generally resulted from different values or contents for the same type of information. In the restaurant recommendation example, the difference in the service time of a day or the average price level would totally distinguish restaurants and draw the interests of different groups of consumers, even when they provide the same type of foods. Simply ascribing information into coarse types (food, movie, car, etc.) cannot meet most application needs. On the other hand, completely expressing every detail of the information in words and matching over them is not feasible in reality. We need an information system that supports rich and accurate information content expression while efficiently reducing the representation complexity.

In this paper, we propose a reliable and scalable content-expressive information matching and dissemination infrastructure in a large-scale mobile wireless network, which utilizes novel and efficient components as well as a location-based virtual management infrastructure for efficient storage, lightweight communications, and quick information match.

The main contributions of our work are:

- We propose a mechanism to flexibly and efficiently represent information with the combination of a set of elementary tuples for numerical expression of the content.
- We propose a novel Attribute Range Vector that allows flexible vector length adjustment based on the information accuracy requirement, and supports a unique simple bitwise operation for quick content matching check, to facilitate accurate content representation as well as low-overhead in storage and transmission.
- We propose a Summary Tree structure to facilitate efficient aggregation of information, which significantly reduces the overhead for storing and transmitting infor-

mation updates.

- Different from Pub/Sub systems, which generally match the publication over predefined or existing subscriptions, where subscribers usually have to wait, our scheme makes the matching process bidirectional so that all information can be promptly processed for matching.

The rest of the paper is organized as follows. In Section II, some related works are discussed. Section III gives the system basic conventions and an overview of the overlay structure. Section IV outlines the detailed design and algorithms for information matching of BRVST. Extensive simulations are evaluated in Section V. We conclude the work in Section VI.

II. Related Work

There are lots of studies on developing information matching mechanisms, among which Publish/Subscribe systems are once prevalent. However, the work on Pub/Sub systems over wireless networks is far less mature than that in wired networks.

Very few efforts have been made to support flexible content-based information matching and dissemination over wireless networks. One of the challenges is to accurately represent the content which often has a value range and to support efficient query on the ranges. R-Tree [6] supports range query for a single content attribute, but the structure consumes too large space when the information is composed of multiple attributes. Bloom Filter can also be modified to support range query. MDSBF [7] combines multiple bloom filters with each one representing one attribute of the content. However, this can easily get into computational bottleneck as information volume increases, because the query on each attribute bloom filter requires several hashing operations. TAMA [8] has its own design to express numeric ranges. Its fixed granularity-level design, however, lacks the ability to balance between content representation accuracy and storage efficiency. Besides, TAMA maintains information in tables without aggregation, which is not efficient in both space and time complexity. Instead, our novel variable-length attribute range vector, which supports convenient aggregation, can not only flexibly represent numeric range of content to any desired accuracy level with low storage space, but also take advantage of simple bit-wise operations to facilitate efficient information matching.

Other types of systems such as [9] by Picco et al. assume tree-based topologies, which are hard to maintain and vulnerable to network topology changes. To avoid this drawback, the wireless network can be divided into regions for more efficient management and information distribution. DRIP [3] groups nodes registered to different broker nodes into Voronoi regions whose shape and size could change over time. However, it may involve a high overhead to maintain the topology region especially over a mobile network. Based on virtual infrastructure, our design avoids the high overhead of region maintenance and also facilitates information aggregation to minimize information update changes.

III. Model Basics and System Overview

In this work, we adopt the notion of Publication and Subscription to distinguish information from the generators and to the consumers. The whole information space is built up with the basic element - attribute ($A_i, i = 1, 2, ...$), which contains attribute name (a_n) specifying the identification of an attribute (numeric ID in realization), and attribute value (a_v) that specifies the content and is usually a numeric point or range. i.e. $A_i = \{a_n, a_v\}$.

A subscription **s** is a conjunction of n attributes: **s** = $\{A_1 \wedge \cdots \wedge A_n\}$. A publication **p** is a disjunction of attributes: **p** = $\{A_1 \vee \cdots \vee A_n\}$, and is also referred to as an event. Conventionally the attribute value of a subscription could either be a numeric point or a range, while that for publication is assumed only to be a numeric point, and many literature studies [8] [10] have followed this convention. However, very often some attributes of the information, when generated, are not absolute point values. For example, the video surveillance data could have its *time* attribute as a range which confines the start and end points of a video segment. So our design also supports range value for a publication attribute.

We assume all data published are trustful, and there is no fraud or spam. Detecting malicious data is not our focus.

For users to get more precise information, we consider a publication and a subscription to match each other iff: for each attribute existing in the subscription, the same attribute must also exist in the publication; and for the common attributes, those from the publications must have their value ranges contained by the value ranges of the corresponding attributes in the subscription. i.e.$\forall A^s \in s, \exists A^p \in p : (a_n^p = a_n^s, a_v^p \subseteq a_v^s)$, where the superscript ^{s} denotes the subscription, while ^{p} denotes the corresponding terms for a publication.

In order to make the infrastructure scalable and more robust to the network dynamics, we introduce a virtual management infrastructure where the network space is mapped into virtual zones each consisting of a set of virtual grids (Fig. 1). With many wireless devices equipped with GPS receivers or having other methods of localization [4], the grid and zone which a node belongs to can be easily calculated based on node location in reference to a reference virtual origin and a pre-determined grid or zone size [11]. There is no need of a complicated scheme to create and maintain the virtual grids or zones. The grid size can be determined by the system based on the application scenarios and performance tradeoffs. Its effects is studied in Section V.

Each grid can elect a Grid Manager (GM) for Pub/Sub message collection, aggregation and matching within the grid. Each zone also has a Zone Manager (ZM) responsible for Pub/Sub aggregation, matching, data catching over grids within the zone. The schemes for leader election and maintenance have been proposed by many literature work [11] which can be leveraged in our system . The managers can be static or mobile, depending on the system application scenarios.

Event matching and Pub/Sub message update are both performed on demand. Subscriptions and publications in a grid

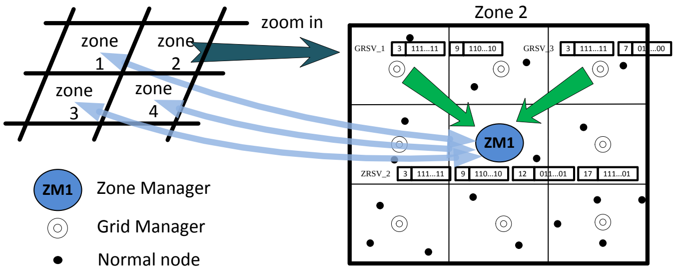

Fig. 1. An example system where each zone has 9 grids. The zone manager collects Pub/Sub messages from grids within the zone and aggregates them into control messages to exchange with other zones.

are collected and aggregated. Although nodes may frequently move in and out of a grid, the aggregate filter may stay unchanged. Messages are sent to the upper level ZM only upon the change of aggregate filter. This will significantly reduce the overhead for Pub/Sub message transmission and matching in a dynamic wireless network. A ZM maintains the Pub and Sub information of the grids within its zone with efficient data structures to be introduced in Section IV, and the Pub/Sub information of the whole zone can be similarly further aggregated. As many mobile users have interests in close-by information, the aggregate filters only need to be shared among nearby zones or zones identified with Pub/Sub relationship.

Any new subscription or publication will trigger the event matching process within its own zone first, then matching at other zones whose aggregate filters imply potential chance of match will initiate. This will significantly reduce the data matching and distribution overhead. Once a publication is matched with one or more subscribers, the overlay structure will then deliver the data to these destinations using the stateless geographic multicasting, RSGM [11], for reliable and low overhead transmissions. The detailed routing process is beyond the scope of this paper.

IV. Bidirectional Content Matching

In many conventional Pub/Sub systems, the subscriptions are specified before the publications. However, some subscribers may indicate their interests on some data that have been published before. Simply throwing away the published data when they cannot match the current subscriptions would waste the system resources consumed for the information matching and distribution. Instead, if the publications can be stored even if they did not find match, the system could immediately deliver the data to later subscribers once they have interests.

In this work, we propose an efficient bi-directional content matching infrastructure, so that newly published data will be timely distributed to existing subscribers matched and new subscriptions can also trigger the retrieving of interested data already published quickly. In face of the challenge of representing the rich contents while not significantly sacrificing system performance, we novelly propose simple binary bit vectors and summary tree structure to facilitate flexible content-expressive information matching and dissemination processes at low overhead for storage, transmission and computation.

A. Binary Vector and its Operations

Content-based information system can potentially support flexible user information need, but at the same time poses high challenges for information representation and matching. We introduce simple *Attribute Range Vector* to facilitate light-weight content-expressive management while not compromising the accuracy of information matching.

1) Attribute Range Vector (ARV): We propose a binary bit vector named Attribute Range Vector (ARV) to flexibly represent the numeric range values of an attribute, referred as the target range. The target range could be a single point value as well. An ARV has a small size and is easy to process. The numeric value of an attribute is generally limited within predefined boundaries, which can be determined in advance by the system based on some common knowledge. For example, the temperature of the weather has an lower and upper limit in physical world. A subscriber could indicate her interest by setting a target range within the limit defined by the system. To facilitate flexible range matching, the predefined limit range is divided into N smaller equal segments, while the value of N can vary based on the matching accuracy requirement. An N-bit ARV is formed by representing whether a segment matches a content range, following the steps below:

- **Step0**: Set the initial segment to be the whole predefined limit range.
- **Step1**: Check if the target attribute value range falls into some existing segments with each occupied more than α (percentage) of the segment range, an accuracy threshold desired. If so, goes to the next step; otherwise divide each of the current segments into equal halves, and continue this step.
- **Step2**: Make an N-bit vector with N equal to the current number of segments, with each bit indicating if the attribute range overlaps the corresponding segment range, 1 yes, and 0 no.

From the above ARV construction process, we can see that the number of bits of the vector can only be the power of 2, i.e., $N = 2^i$, $i = 0, 1, 2, 3...$, and the length of ARV can be continuously doubled until a desired representation accuracy is achieved. The threshold α trades off between accuracy and simplicity of the message representation.

For example, the attribute *Age*, often involved in social network applications, is limited within 0 to 100. Three subscriptions that contain the attribute *Age* are: Age_{Sub1} 1-48, Age_{Sub2} 26-47, and Age_{Sub3} 38-60. Their corresponding ARVs are obtained by constructing a split tree following the above steps as shown in Figure 2, with the level i having 2^i segments. Suppose the threshold α is set to 90% in this example. Age_{Sub1} falls into the segment 0-50 and the fitting ratio of the target range 1-48 is $48/50$, which is larger than the threshold $\alpha = 90\%$. So this segment is accurate enough to represent the target range and the ARV for Age_{Sub1} is 10.

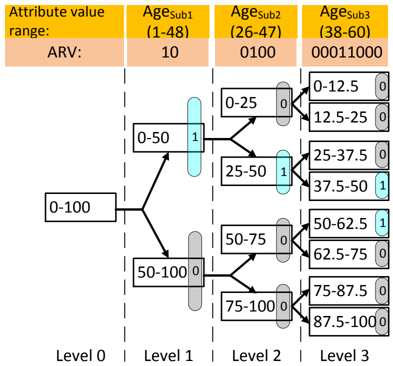

Fig. 2. The segment division procedure in constructing an ARV.

Age_{Sub2} apparently falls into the 0-50 segment of level 1, however, this range is not very accurate. We further divide the overall range into 4 new segments at the level 2, so the range 26-47 falls into the segment 25-50. We can use 4-bit vector 0100 to represent this 4-segment coverage, with the left most bit standing for the segment of the lowest value. The target range 38-60 of Age_{Sub3} spans across the 0-50 segment and the 50-100 segment at the first-level of the split tree, but these two segments are inaccurate in representing the target range. If we go deeper into the level 3, the segment 37.5-50 & 50-62.5 will be accurate enough with the resulting ARV 00011000.

A shorter ARV is always preferable to reduce the transmission and storage overhead. The ARV bit vector is checked after each modification for the potential of simplification. Except level 0, the number of bits in an ARV is always even and in the power of 2. When the length of ARV is larger than 1, starting from one side of the vector, if *every* consecutive 2-bit has the same value (both '1' or both '0'), the length of the vector can be reduced into half by taking every other bit to form a new ARV. For example, 1100 can be reduced to 10, but not 0110 nor 0111 which does not have the same value for consecutive 2-bit. The simplification operation will continue without losing the accuracy of the information until the vector cannot be further simplified.

Likewise, a given vector could also be extended by 2^i (i=1,2,3...) times when needed by simply doubling the bit patterns. This feature is extremely useful in the matching process we will discuss later, where two or more ARVs need to be adjusted to have the equal length before they can be compared or merged.

The proposed ARV is the elementary component of Subs and Pubs, and some other aggregated management structures at different hierarchical levels are composed of ARVs.

Under our matching rule specified in Section III, the attribute of a publication is considered to match with that of a subscription when its value range is contained by that of the same attribute of the subscription. When the same attribute for Pub and for Sub are represented with 2 ARVs respectively, and are scaled to the same length of bits, the matching rule now translates into: the bit positions where the Pub ARV has '1' need also to be '1' for the Sub ARV. There will be a slight chance for two ARVs satisfying this criteria to be actually not matched due to improper choice for the afore mentioned accuracy level threshold α. As these attribute-level false positives get significantly large, an event-level false positive will happen which causes erroneous matching result and thus unwanted traffic overhead. The impact of threshold α on the event-level false positive rate is studied in Section V.

*2) **ARV Merge**:* A merge operation is needed for information aggregation. As the length of the vectors could only be the power of 2, two vectors of different lengths can always be made equal by doubling the length of the shorter one several times as previously mentioned. Suppose we want to merge the same attribute vector of Sub1 & Sub2 which are 0100 & 10 respectively, we only need to scale up 10 by repeating each of its bit once to get 1100, and the merge can be completed by only a simple bitwise "OR" between 0100 & 1100 to get the result 1100. This number can in turn be simplified into 10 without losing the accuracy. This indicates that the segment 0-50 can represent the merge of the ranges 26-47 and 1-48. As the accuracy level for each segment is ensured to be higher than α, the accuracy of the ARV will not be impacted when it is scaled up or down. The merge operation is always carried at the length of longest ARV thus over the finest level of segments, and the merge of ARV will maintain the accuracy level. *The ARV's merit for convenient merge operation is critical to information aggregation which contributes to very low storage and transmission overhead.*

*3) **Match of ARVs**:* Our purpose of introducing ARV is to facilitate fast information matching, which could be easily achieved with fast bit-wise operations under the following conditions:

A subscription, represented by conjunctions of attributes like A∧B∧C, where A, B and C are three different attributes, is considered to be matched only if all the attributes are satisfied. A publication is allowed to have additional attributes than A∨B∨C, i.e. A∨B∨C∨F∨G, to still be considered as matching the subscription, as long as all the attributes of the subscription (A, B and C in this example) are satisfied on their values. This convention intuitively means that subscribers will always accept information that is more elaborate than their expectations.

For differentiation and ease of referral, an subscription and publication attribute range vector are called respectively an S-ARV and P-ARV. If one or more attributes of the subscription are not included by the publication, we can immediately claim they do not match each other, given the conditions above. Otherwise they are further checked. First all the S-ARVs and P-ARVs are respectively concatenated following the corresponding order as shown in Figure 3, with all the redundant P-ARVs ignored and each corresponding pair of P-ARV and S-ARV scaled to the same length. Then the Sub and Pub are considered to match each other if and only if all bits after the following operations are 0: The cascaded P-ARVs vector and S-ARVs vector first have the bitwise AND operation, and the result XOR with the original cascaded P-ARVs vector.

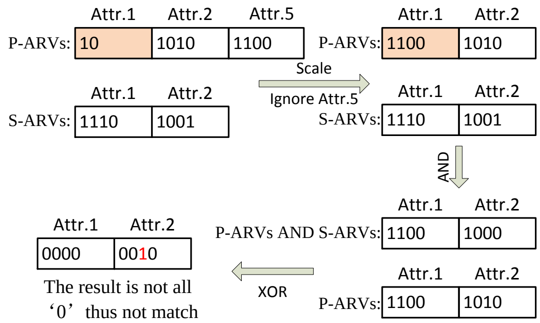

Fig. 3. The bit-wise matching evaluation of a Pub and Sub.

Figure 3 gives an example. The subscription has 2 attributes, and the publication has 3 attributes. To perform the matching, the attribute 1 (Attr.1) of the publication is scaled to 4 bits, while the Attr.5 is omitted because it is not involved in the subscription. Then bitwise operations are carried out: (P-ARVs AND S-ARVs) XOR P-ARVs, and the result is not all '0' thus is not a match. Because Attr.2 of the publication has a '1' in the bit position where the subscription Attr.2 does not, which means the attribute 2 value range of the publication is out of that of the subscription.

B. Basic System Architecture and Maintenance

As introduced in Section III, we take the virtual grid as the lowest-level management unit for a multi-hop wireless network. Each grid has a Grid Manager (GM), and a set of grids form a zone that is under the control of a Zone Manager (ZM). The example system in Figure 1 is split into multiple zones with each zone being composed of 9 grids. In this section, we present the functions at each level of our infrastructures.

*1) **Subscription Maintenance at the Grid Manager**:* A subscriber sends its subscription to its grid manager on demand. Each subscription message is a concatenation of all its attributes, i.e. all the corresponding ARVs. There are possibly many subscriptions in an information-dense area. Simply storing and transmitting all subscriptions would not only incur a large overhead in traffic and storage but also difficult to track the frequent subscription changes due to the user mobility and frequent user interest changes. On the other hand, selectively ignoring some of the subscriptions would compromise the system performance. In our system, the GM will aggregate the subscriptions by finding the minimum representative subscription set to represent all the subscriptions within the grid before recording them and sending them to the upper level.

Two subscriptions could share some common attributes, and the attribute set of a subscription could contain all the attributes of another subscription. In the second case, if the value ranges of the common attributes overlap each other to some extent, we could take the subscription which has all its attributes contained by the other subscription as the representative subscription of both subscriptions. However, if the value ranges of the common attributes do not have any intersection, then using one subscription to represent the other is not appropriate. We use an example to illustrate this aggregation principle. Suppose there are 2 subscriptions in a grid, SUB1: A and SUB2: $A \wedge B \wedge C$, where A, B and C are different attributes. According to our scheme, since all publications that contain the attribute A including the ones that also contain B and/or C will all be routed to this grid for further matching, thus taking SUB1 as the representative subscription, compared to otherwise having both SUB1 and SUB2, will help reduce the subscription information storage and control traffic without sacrificing the completeness of subscription information in this grid. Once receiving the information based on the aggregate filter, the GM will further match the information with individual subscription to determine if the information matches all the criteria of a subscriber. Thus aggregation reduces the message and data transmission between the ZM and GM, but does not sacrifice the accuracy requirement of each subscriber. The subscription aggregating process can be realized through

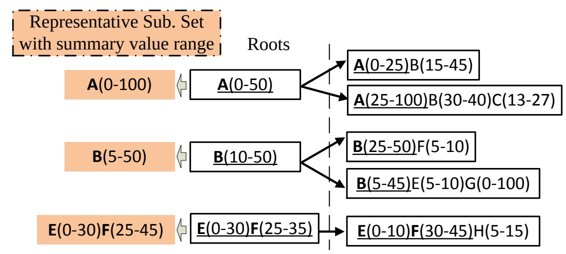

Fig. 4. Summary forest with attribute summary value range in shade.

a summary tree, which is actually a forest containing several separate trees as shown horizontally in Figure 4. All the subscriptions of a tree will be represented by its root, and a tree node will contain all attributes of the root. There is also a summary range attached to each root shown as the shaded block in Figure 4, obtained by merging ('OR' operation) the value range of common attributes (underscored in Figure 4) of all the subscriptions on a tree. When determining if a node should be inserted into a tree, we will check if some of its attributes are the same as the root and if the attribute ranges overlap the current summary ranges. The summary ranges of all trees form the representative subscription set of the grid as shown on the left side of the dash line in Figure 4.

Algorithm 1 shows how to add a subscription into the current summary forest. On lines 3-10, a new subscription will become either the child or the parent of an existing root, depending on whether it contains all the attributes of a root or all of its attributes are contained by a root of the forest, with the value ranges of corresponding common attributes overlapping each others. Otherwise, the subscription will be made a new stand alone root, as shown on lines 12 and 16. On line 18, after inserting the new subscription, the summary value range attached to the root of the affected tree will be updated. Line 19 checks whether trees can be merged to one another to reduce the number of trees in the forest, i.e., the size of the forest, every time the summary value range of a tree is changed, by examining whether one tree root can be inserted as the child of another tree root following the similar criteria.

Algorithm 1 Adding a subscription s into the summary forest

```
if there are already nodes in the forest then
    for each root node R_i of the forest do
        if the subscription s contains all the attributes in R_i then
            if the summary value range of each attribute in R_i overlaps that of s then
                insert s as the child of R_i into the summary tree;
            end if
        else if R_i contains all the attributes of s then
            if each attribute value range of s overlaps the summary value range of the same attribute in R_i then
                make s the parent of R_i as the new root;
            end if
        else
            make s a new root of the forest;
        end if
    end for
else
    make s a new root of the forest;
end if
Adjust the summary value range of the affected tree.
Check whether the forest can be reduced by merging trees.
```

For illustration, suppose a grid has the following subscriptions with letters representing different attribute names: A(0-50), B(10-50), A(0-25)B(15-45), A(25-100)B(30-40)C(13-27), B(25-50)F(5-10), E(0-30)F(25-35), B(5-45)E(5-10)G(0-100), E(0-10)F(30-45)H(5-15). Applying them one after another with Algorithm 1 will generate a summary forest as shown in Figure 4.

Algorithm 2 Removing a subscription s from the forest

```
1:  if s is a root of the forest then
2:      delete the tree originated from root s;
3:      for each children node of s do
4:          apply Algorithm 1;
5:      end for
6:  else
7:      delete s from the summary tree;
8:  end if
9:  Adjust the summary value range for each affected tree.
10: Check whether the forest can be reduced by merging trees.
```

Algorithm 2 works to remove a node in response to unsubscription. On lines 1-5, if the subscription to be deleted is the root of a tree, then this whole tree is removed with all the non-root nodes reinserted into the forest by applying algorithm 1 one by one. If the subscription is not a root, it is simply deleted from the tree as shown on lines 6-7. Then the affected trees will have their summary value ranges updated accordingly on line 9. Line 10 works similarly as the last line of Algorithm 1 to reduce the forest size.

Each GM will maintain a subscription summary forest, and updates the trees in response to the changes of subscription from individual subscribers within the grid. When a node wants to send a new subscription, modify or unsubscribe its existing subscription, it will send a message with the affected sub through on-demand light-weight geographic routing [12] to the GM. The GM will either insert or delete the subscription following the Algorithm 1 or 2. A new action may change the representative set. In many cases, however, *individual subscription changes will not lead to the change of the aggregated information summary at the root level of the tree*. This feature is very important. It helps to increase the stableness and significantly reduce the information maintenance overhead in a wireless environment with possible constant node movement and thus frequent subscription changes. The representative set is forged into a vector, named Grid Representative Set Vector (GRSV) as shown in Figure 5 by cascading each subscription from the representative set. The GRSV will be sent to the ZM upon its change to reduce the update overhead.

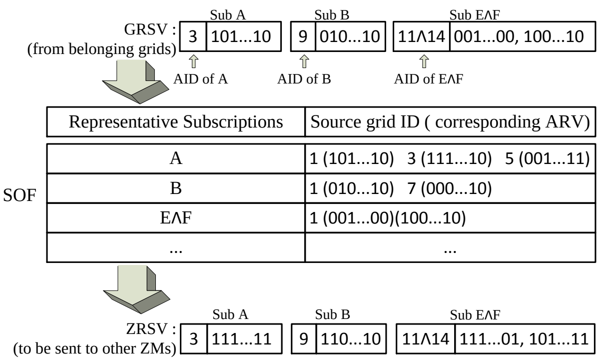

Fig. 5. The ZM converts the GRSVs received from belonging grids into SOF, then converts it into ZRSV by summary tree scheme.

*2) **Subscription Maintenance at the Zone Manager**:* Each zone manager maintains a subscription origin form (SOF) generated based on the GRSVs sent by grids with subscriptions within its zone, as shown in Figure 5. The representative subscriptions from the grids will again be aggregated through the summary tree scheme similar to that at the grid level. We cascade each subscription of the resulting representative set to form a long vector - Zone Representative Set Vector (ZRSV). The ZRSVs are exchanged among ZMs to guide the publication distributions. The SOF will be updated if there is a GRSV update, but similar to the grid level aggregation, an individual update in SOF may not lead to ZRSV change. The aggregation helps to reduce the message distribution and simplify the information matching process, which is more critical for dynamic wireless networks. The ZRSV only needs to be distributed to relevant zones upon changes, after a given period, or when the zone receives unwanted traffic. Each ZM maintains the ZRSVs received from neighboring zones and zones interested in its publications (learned from previous successful match processes) to guide the distribution of published data.

C. Match a Publication over Subscriptions

When a node generates a publication, it will send the data along with the publication ARVs describing the data to its GM. GM will perform a match within its grid by comparing the publication ARVs with its representative Sub set, i.e., the roots with summary ranges of the summary forest, using the matching rule defined in Section IV-A3. If a root is matched, each of its tree node is further examined to precisely find the subscribers. The data will be forwarded to the identified subscribers through on-demand stateless geographic multicast scheme [11]. No matter local matches are found or not, GM will forward the data along with the P-ARVs and the grid ID to the zone manager.

The ZM will match the P-ARVs against its SOF, to decide which grids within the zone to forward the data to for further matching at GM level. It also matches against all ZRSVs for other zones it maintains. The data along with the publication P-ARVs and the zone ID will be multicasted towards the centers of the zones that match this publication. Once the data reach a target zone, they will be forwarded to the ZM which will match the Pub with each item of the SOF. The data will be multicast to the matched grids, where the GMs will again multicast the data finally to the matched subscribers.

As mentioned earlier, each ZM only actively maintains the ZRSVs of its neighbors. However, other zones may also have subscribers to its publication. If the publication ARVs associated with a publisher are seen by the ZM the first time or after a given time period since its last global distribution, the ARVs will be multicasted to all zones to inform them the existence of new publications. A zone x with the matched ZRSV will send to the ZM its ZRSV, which will be maintained by the ZM along with other ZRSVs. ZM will multicast publication data to the zones with matched ZRSVs. A zone will update its ZRSV to the publication zone following the ZRSV update rules described earlier. A ZRSV will be removed if there are no data match with it for a predefined timeout period. To further reduce the overhead, for a large system, the period of sending the publication ARVs to farther-away zones can be made larger as generally the information has location constraints. In addition, a zone without any data matched with some subscriptions could also actively search for publishers by broadcast a query message within certain range or query the ZMs within certain zone-hop distance.

D. Publication Caching and Match

Publications may not match any subscription in a single attempt, and a subscriber may want to retrieve earlier published data. Conventional studies generally assume publications always get matched; if not, the unmatched publications are simply discarded. This would waste the system resources that haven been used in generating, matching and distributing these published data, and also cannot meet the users' urgent needs for previously published data if discarded. In this work, we introduce publication caching to facilitate bidirectional matching which also supports matching a subscription over cached publications. A zone manager receiving a publication will cache the data at the ZM or designated storage server for a predefined duration, and records the ARVs of this pub along with its source node's ZID and GID. In case that the caching space is running out, data with least matching-hit records will be removed.

A ZM holds SOFs of its own zone and ZRSVs of other relevant zones. Upon the update of the SOF or ZRSV, the ZM will compare the changed SOF or ZRSV with the ARVs of the cached publication so that the matched subscribers can get the interested data right away.

V. Simulation and Performance Evaluation

We implement BRVST using NS2.34. The focus of BRVST is on information content matching and forwarding mechanisms, and the underlying routing scheme follows SOGR [12] and RSGM [11] for on-demand robust unicast and multicast respectively. 400 nodes are randomly distributed initially in a network region of size 1000m x 1000m to reflect the real-world mobile user density. In our default setting, the network is divided into 4 equal zones with 4 equal grids inside each. These numbers will vary when studying the impact of grid size on system performances. The node movement follows the improved Random Waypoint model [13]. All the nodes including the autonomously elected GM and ZM could move following the model. The wireless channel propagation model is set to be TwoRayGround, and 802.11a is adopted as the MAC protocol with an average transmission range of 80m. Publications and subscriptions are generated by randomly selected nodes. Each publication or subscription has one to three attributes, which are randomly selected from a predefined set of 15. The range of an attribute is also randomly generated within a predefined range limit based on the attribute type. If not otherwise specified, the average node moving speed is set to 5 m/s, the Pub and Sub generation rates are both set to 200/minute, and the accuracy threshold α is set to 90%.

There is very limited number of studies closely related to ours. For performance references, we select two existing Pub/Sub schemes, DRIP and TAMA, that are partly comparable to our work. DRIP [3] (INFOCOM'08) is proposed for wireless networks which group nodes into Voronoi regions to manage the network, while BRVST introduces geographic zones to facilitate management and information distribution. TAMA [8] (ICDCS'11) is a middleware for content matching, but is not specified for wireless networks. To be fair, we compare the impact of node mobility on the matching time for DRIP and BRVST in wireless environment, without including TAMA. The number of Voronoi regions for DRIP is also set to 16 under the same region area and node density. Since TAMA also considers using attribute range to describe contents, we compare it with BRVST on the false positive rate. The management overhead involved for storing and transmitting publication and subscribe filters are compared among all three schemes.

A. Matching Time

It is equally important for both the information provider and consumer to be served as fast as possible, so we evaluate the time for an emergent publication and an emergent subscription to get matched separately.

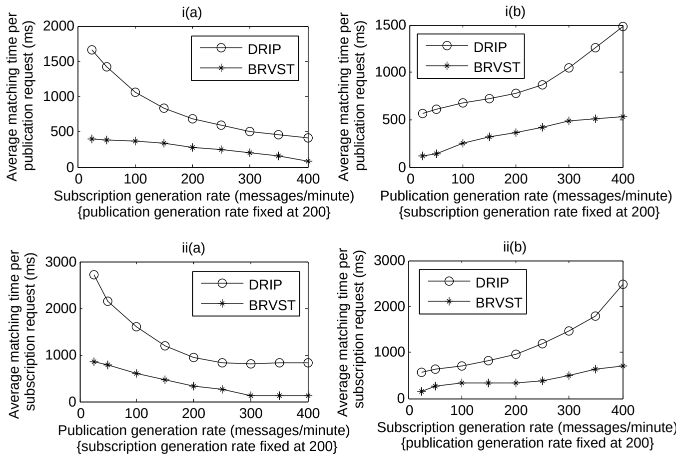

Fig. 6. i(a) Matching time per Pub request as Sub rate increases; i(b) Matching time per Pub request as Pub rate increases; ii(a) Matching time per Sub request as Pub rate increases; ii(b) Matching time per Sub request as Sub rate increases.

For each newly published event, we evaluate the average time taken to match it with the subscribers. We allow publication to be matched with a later generated subscription and vice versa, so the delay is also affected by the subscription and publication generating frequency, as shown in Figure 6-i. In Figure 6-i(a) the publications rate is fixed at 200/min, while the subscription rate is varied. In Figure 6-i(b), the subscription rate is fixed at 200/min, while the publication rate is varied. Similarly, we evaluate the average time duration for a newly generated subscription to match the publication in Figure 6-ii(a) and (b), with the subscription and publication rate fixed at 200/min respectively.

We can observe that BRVST has a much shorter average matching time as compared to DRIP under all test scenarios. A publication (or subscription) request has a shorter time to be matched when there is a higher subscription (or publication) rate as shown in Figures 6-i(a) and ii(a). The reduction of matching time reaches a limit, beyond which the matching time may slightly increase as a result of higher processing overhead.

On the contrary, as the publication (or subscription) rate becomes larger, the time to match a publication (or subscription) increases as a result of competitions, which deteriorate the average matching time, as shown in Figures 6-i(b) and ii(b). As DRIP involves network-wide broadcast to establish and maintain Voronoi regions, the matching time increases exponentially, while BRVST has only a sub-linear increasing time, which indicates its better scalability to system load.

Figure 7-(a) tests and compares the reliability of BRVST and DRIP in terms of matching time performance under high node mobility, with the average node speed varying from 0 to 20m/s. The average matching time per message (including either the publication match or subscription match) of DRIP increases significantly as a result of its broadcast-based management overhead. The delay becomes more severe when the average moving speed is higher than 10m/s, where

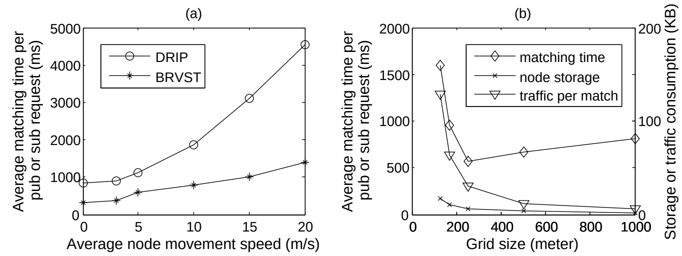

Fig. 7. (a)Mobility impact on average matching time; (b)Grid size impact on BRVST's average matching time per message, system average node storage consumption and traffic volume incurred per match. The setting of grid size variation corresponds to the number of grids varying from 64 downto 1.

nodes could move across regions within the average matching duration. Based on light-weight virtual management infrastructure, BRVST has much more stable performance in the mobility case.

In Figure 7-(b), the matching time is seen to first reduce with grid size and then increase. As the grid size increases, the number of grids decreases so does the number of zones, while the number of nodes in a grid increases. In a larger grid, messages are more likely to get matched within the grid or zone, and there are fewer other zones to check with. However when the grid size gets too large, messages need to interact over longer distance with GMs and ZMs. In addition, a large number of nodes also result in more filters in a grid which incurs a longer matching time.

B. System Maintenance Overhead

We compare the overhead for storing and transmitting management messages at broker nodes and regular network nodes respectively. In Figure 8, the publication and subscription rates increase at the same speed.

In Figure 8-i(a), TAMA and BRVST both have lower storage overhead at regular nodes, as these nodes do not store publication and subscription information. Specifically, BRVST only requires each node to keep a few ID numbers which are very small in volume. With the need of storing a delay list of brokers and neighboring information, DRIP has much higher storage overhead, and the overhead increases quickly with the load.

In Figure 8-i(b), the storage overhead at brokers for all three schemes increase linearly with the load. DRIP has a much higher increasing rate with its need of maintaining information of both non-broker nodes and other brokers, as well as the subscriptions and publications of all the nodes in the network. Both TAMA and BRVST exploit range-based content representation to reduce the storage space. BRVST exploits space efficient aggregate scheme, so its storage space is 60% lower than that of TAMA.

We compare DRIP and BRVST on the overhead for transmission of management messages. In Figure 8-ii(a), the overhead of DRIP increases exponentially due to its inefficient broadcast mechanism. BRVST does not requirement significant overhead to maintain its zone and grid infrastructure, and

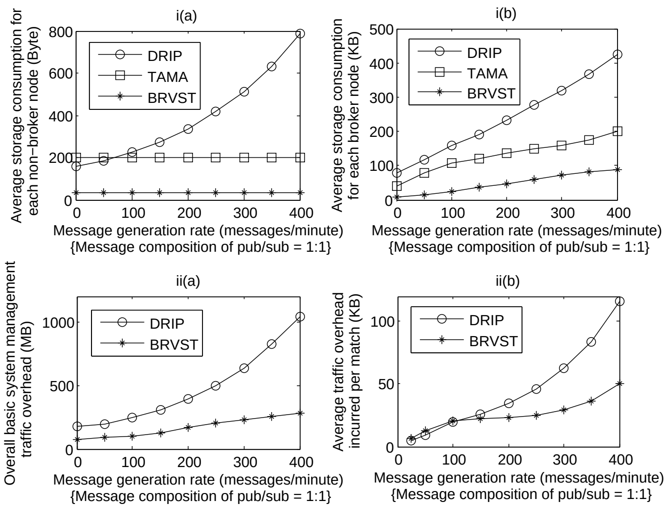

Fig. 8. Storage consumption for i(a)non-broker node; i(b)broker node; ii(a)Basic system traffic overhead; ii(b)Traffic overhead incurred per match.

only sends highly aggregated publish or subscribe information, thus it has a much lower transmission overhead.

In Figure 8-ii(b), when the message rate is low, BRVST and DRIP have similar matching overhead. At a higher load, however, the overhead of DRIP increases exponentially, while the overhead of BRVST is compensated as each publication can match multiple subscriptions with its aggregate subscription mechanism.

In Figure 7-(b), as grid size increases, both the average node storage space and the traffic volume incurred for each match reduce. With a larger grid size, nodes are less likely to move out of the grid, thus the overhead associated with grid change will be lower. A larger grid also allows better information aggregation, thus reducing the matching traffic.

C. False Positive Rate

As TAMA and BRVST represent contents with certain range granularity to reduce complexity, it would also introduce some false positive rate and forwards some unwanted traffic to nodes.

Figure 9-(a) shows that the false positive rate is inversely proportional to the accuracy threshold α, and approximately bounded by $1 - \alpha$. There is an obvious tradeoff between the accuracy level of representing information and the length of the ARV vector thus the overhead. The higher the accuracy level, the more storage and traffic volume incurred. Figure 9-(b) shows as the false positive rate rises, the traffic overhead of both system increases. However at the same false positive rate, BRVST would waste much less traffic than TAMA due to its efficient ARV representation.

VI. Conclusion

In this paper, we present BRVST, an information content matching and forwarding engine in wireless network, which supports maximum flexibility in the expression of information

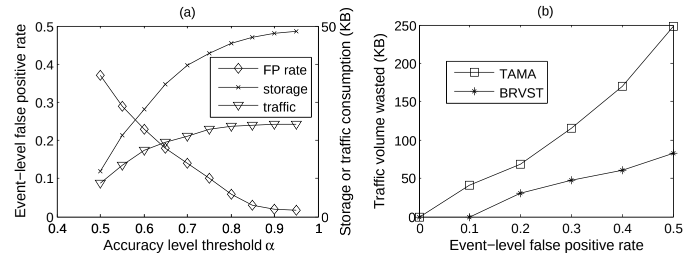

Fig. 9. (a)Threshold α impact on BRVST's event-level false positive rate, average broker storage consumption and traffic volume incurred per match; (b)Traffic volume wasted due to false positive match for BRVST and TAMA.

content. The most valuable contributions of BRVST are its introduction of a novel attribute range vector that can accurately represent information content with extreme efficiency both in space and computationally, and the summary tree concept that enables effective extraction and aggregation of information. All these proposed structures help significantly reduce storage and communication consumption as well as computation overhead, and ensure stable performance. Extensive simulations demonstrate that BRVST is reliable and scalable in large and dynamic wireless network conditions even under very high information load.

References

[1] R. E. Strom, G. Banavar, T. D. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. C. Sturman, and M. Ward, "Gryphon: An information flow based approach to message brokering," *CoRR*, 1998.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, pp. 332–383, August 2001.

[3] Q. Yuan and J. Wu, "Drip: A dynamic voronoi regions-based publish/subscribe protocol in mobile networks," in *INFOCOM 2008*, april 2008, pp. 2110 –2118.

[4] N. Carvalho, F. Araujo, and L. Rodrigues, "Reducing latency in rendezvous-based publish-subscribe systems for wireless ad hoc networks," ser. ICDCSW '06. IEEE Computer Society, 2006.

[5] J. Mocito, J. A. Briones-García, B. Koldehofe, H. Miranda, and L. Rodrigues, "Geographical distribution of subscriptions for content-based publish/subscribe in manets," in *Proceedings of the ACM/IFIP/USENIX Middleware'08*. ACM, 2008, pp. 102–103.

[6] A. Guttman, "R-trees: a dynamic index structure for spatial searching," ser. SIGMOD '84, 1984, pp. 47–57.

[7] Y. Hua, D. Feng, and T. Xie, "Multi-dimensional range query for data management using bloom filters," ser. CLUSTER '07, 2007, pp. 428–433.

[8] Y. Zhao and J. Wu, "Towards approximate event processing in a large-scale content-based network," ser. ICDCS '11, pp. 790–799.

[9] G. Picco, G. Cugola, and A. Murphy, "Efficient content-based event dispatching in the presence of topological reconfiguration," in *ICDCS 2003*, may 2003, pp. 234 – 243.

[10] Z. Jerzak and C. Fetzer, "Bloom filter based routing for content-based publish/subscribe," in *Proceedings of the second international conference on Distributed event-based systems*, ser. DEBS '08, pp. 71–81.

[11] X. Xiang, X. Wang, and Y. Yang, "Stateless multicasting in mobile ad hoc networks," *IEEE Transactions on Computers*, vol. 59, no. 8, pp. 1076 –1090, aug. 2010.

[12] X. Xiang, X. Wang, and Z. Zhou, "Self-adaptive on-demand geographic routing for mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 1, p. 99, 2011.

[13] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model," *IEEE Transactions on Mobile Computing*, vol. 3, no. 1, pp. 99–108, 2004.