Design of a Multiple Bloom-filter for Distributed Navigation Routing

Ping Jiang, Yuanxiang Ji, Xiaonian Wang, Jin Zhu and Yongqiang Chen

***Abstract*— Unmanned navigation of vehicles and mobile robots can be greatly simplified by providing environmental intelligence with dispersed wireless sensors. The wireless sensors can work as active landmarks for vehicle localization and routing. However, wireless sensors are often resource-scarce and require a resource saving design. In this paper, a multiple Bloom-filter scheme is proposed to compress a global routing table for a wireless sensor. It is used as a look-up table for routing a vehicle to any destination, but requires significantly less memory space and search effort. An error expectation based design for a multiple Bloom-filter is proposed as an improvement to the conventional false positive rate based design. The new design is shown to provide an equal relative error expectation for all branched paths, which ensures a better network load balance and uses less memory space. The scheme is implemented in a project for wheelchair navigation using wireless camera motes.**

***Index Terms*— Bloom-filter, mobile robots, navigation, routing, wireless sensor networks**

I. Introduction

Autonomous navigation is a traditional research topic in intelligent vehicles and robotics. An autonomous robot perceives the environment from its on-board sensors, such as cameras and laser scanners, and drives to a destination by centralized information processing. Most research to date has focused on the development of a brain large and smart enough to gain autonomous capability for robots[1]. The techniques have been developed with success, but they are nonetheless faced with difficulties and ambiguity in understanding the environment due to the limitations of only using on-board sensors. If the environment is provided with pervasive intelligence using a wireless sensor network, the difficulties can be significantly alleviated with less environmental uncertainty for the robot. Firstly, the distributed sensor network can provide a topological map of the environment. The task of routing to a geographic goal now becomes a sensor-querying sequence in the sensor network. Secondly, distributed sensors become active landmarks for robot localization. This is much more reliable and efficient than map based probabilistic localization[1-3]. Thirdly, with static cameras mounted in the environment rather than on a mobile robot, each sensor will be in charge of a local region; as a result, there is less uncertainty and higher reliability in both environment sensing and motion control. Under the control of such a wireless sensor network, mobile robots with less on-board intelligence can be expected to have superior mobility. The solution is feasible in applications where a large number of robots need to be controlled in a certain area, such as the navigation of wheelchairs in a care centre, trading environment intelligence for expensive on-board intelligence in every robot.

Manuscript received October 21, 2011. This work was supported in part by the Royal Society of UK and the NSF of China.

P. Jiang is with the Department of Computer Science, University of Hull, Hull, HU6 7RX, UK (corresponding author: phone: 0044 1482 465680; fax: 0044 1482 466666; e-mail: p.jiang@hull.ac.uk).

Y. Ji, X. Wang, and J. Zhu are with the Department of Information and Control, Tongji University, Shanghai, 200092 China(e-mails: jyx851110@hotmail.com, dawnyear@tongji.edu.cn, zhujin@tongji.edu.cn).

Wireless sensor networks can distribute sensing and processing units within an environment for effective and efficient information gathering. There are a wide range of applications in which wireless sensor networks outperform traditional centralized sensing systems. Examples include monitoring the vital signs of patients in hospital[4], managing land in agriculture[5], creating virtual fencing for animals using acoustic stimuli[6], and monitoring sub-glacier environments to better understand the Earth’s climate[7]. Nowadays, wireless sensor networks are no longer solely used for data acquisition and information gathering. Active decision making and control with pervasive intelligence has led to a new wave of research on cyber-physical systems[8, 9]. Cloud robotics recently became a new research field to expand a robot's knowledge beyond its physical body, so that a robot can become smaller, cheaper, and smarter[10]. Autonomous navigation is an application which would benefit greatly from external intelligence. Wireless sensors can be adopted by intelligent transport systems to provide vehicles with information about the conditions in its surrounding environment[11], and they can even provide vehicles with navigational services[12]. Wireless sensors can also actively locate and guide the visually impaired to avoid risk[13] and navigate people out of dangerous areas[14].

However, individual sensors in a wireless sensor network are often tiny and resource-constrained with limited energy and on-board computational capacity [15, 16]. Energy-efficient designs have been a vital challenge for wireless sensor network research[17, 18]. The resource constraints in terms of storage as well as communication create additional challenges to routing other than those found in traditional ad-hoc wireless networks.

This paper investigates vehicles routed to a destination using a wireless sensor network, which shares massive centralized computing with distributed sensors. Traditionally, routing or global path planning for navigation is accomplished by an on-line shortest path search on a road map, such as Wavefront Expansion and Dijkstra's search algorithm[1], or by multiple hops communication in a wireless sensor network, such as generating a navigation field[19]. However, on-line searching or network propagation searches should be avoided due to the constrained energy and bandwidth available in a wireless sensor network. In this paper, we take a routing table based approach as opposed to the on-line search. The shortest paths in the network are planned off-line and saved in every sensor node to direct a vehicle to a destination. In order to be efficient in communication and storage, a Bloom-filter based scheme is proposed for storing the routing table.

Bloom-filters are an efficient and lossy way to describe membership of elements belonging to a set[20]. They are broadly used for efficient membership determination in many fields, such as the detection of an attack in network security[21], packet classification[22], spell-checkers[23], and dictionaries of passwords[24]. Since the 1990s, Bloom-filters have become popular in many Web applications. Fan and Cao et al. used a Bloom-filter to develop a summary cache for web cache sharing[25], where a proxy used a Bloom-filter based query to determine if another proxy cache held a desired Web page. Czerwinski, Zhao et al. described a routing protocol in which the resource lists were represented by Bloom-filters[26]. Recently, more research efforts have been reported for improving the conventional Bloom-filters. Fan and Cao introduced counting Bloom-filters to support dynamic modification of content[25]. The possibility of a false negative occurring in counting Bloom-filters was analyzed in [27]. Mitzenmacher demonstrated a compressed Bloom-filter to reduce transmission bits[28]. Xiao and Hua presented a parallel Bloom-filter solution for querying multi-attribute items[29]. Rhea and Kubiatowicz introduced attenuated Bloom-filters for probabilistic location and routing of peer-to-peer location mechanisms[30].

To compress a set of strings, a Bloom-filter hashes the strings into a bit vector. To determine the membership of any query string, the Bloom-filter uses the same hash functions to check if the corresponding bits have been set when they were encoded, although there does exist a false positive possibility. The traditional design of a Bloom-filter considers a trade-off between the false positive probability and the memory usage. In order to represent several branched paths at a node, a multiple Bloom-filter is proposed in this paper, with each Bloom-filter in it saving a compressed routing table for each path.

This paper first introduces multiple Bloom-filters for robot navigation routing. The paper then analyzes the multiple Bloom-filters, designed with the conventional approach to have a constant false positive probability, and points out that different branches may have unbalanced relative errors. A path with less encoded nodes will exhibit a higher relative error than a path with more nodes, which should be avoided in navigation. An error expectation based design is then proposed to overcome this issue. It is proved that the new design can ensure equal expectation of relative errors for all branches and use less memory than the conventional design. The proposed multiple Bloom-filter scheme is implemented in a wireless camera network for wheelchair navigation and verified by experiment.

II. A Distributed Visual Sensor Network for Wheelchair Navigation

In order to develop a technical solution for navigation of a wheelchair using pervasive intelligence, a system with wireless visual sensors distributed in the environment and connected by wireless communication was developed in the project WiME(Wireless Mosaic Eyes)[31]. The system consists of three hardware systems, 1) networked wireless visual sensors, 2) wireless controlled robots, and 3) a remote console. The networked wireless visual sensors are the main elements to accomplish distributed localization and navigation, with path routing as one of the functions. In the project, we modified an off-the-shelf wheelchair to be controlled by the wireless sensors. The only processor on-board the wheelchair was an 8-bit Atmega 128L, which was used to link the visual sensor network wirelessly through the IEEE 802.15.4 protocol and to drive the two differential wheels of the wheelchair. Such a low-performance processor is not powerful enough to carry out global navigation. However, the visual sensors deployed in the environment can provide enough distributed intelligence to control the wheelchair.

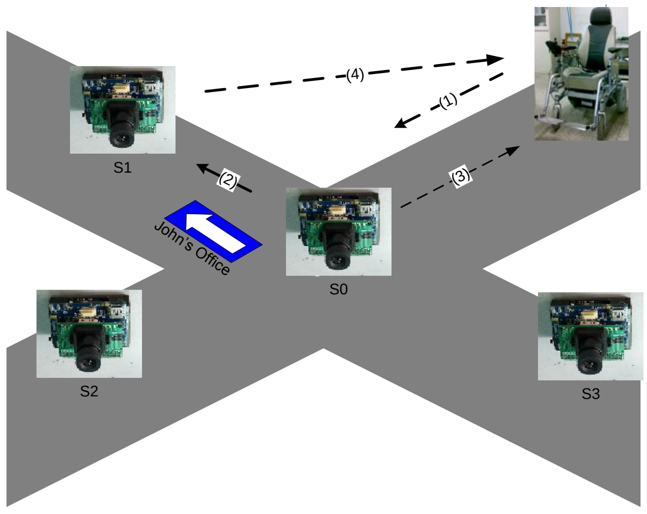

Fig. 1. Control flow of the WiME system

The control flow of navigation data can be illustrated in Fig.1. A wheelchair with John's office as its destination first sends "John", (1) in Fig.1, to the nearest sensor S0. It queries S0 which path it should take in order to navigate to the destination with the shortest total distance. The wireless sensor S0 then checks routing tables on-board, which are the multiple Bloom filter presented in this paper, to find out and inform sensor S1 as a partner for the navigation service, (2) in Fig.1.

Sensor S0 then works with S1 together to plan a path and control the wheelchair going through S0 to S1, (3) and (4) in Fig.1, which are achieved by the snake based path planner and tracker presented in [32]. The process continues along the route, sensor-by-sensor, to direct the wheelchair to John's office. In order to achieve this scenario, each distributed sensor needs to be provided with unambiguous semantics and assigned a determinate role for navigation. A configuration software kit was developed for this purpose.

The configuration kit provides a tool for creating a geographic map and locating wireless sensors. It links the geographic map with the sensor topology as its road map for navigation. It then generates the routing tables for each node, where each routing table records a set of locations able to be reached with the shortest total distance by following a branched path. An example of the map configuration for an indoor environment is shown in Fig.2. The red dots indicate the deployed sensor nodes in the environment, and the blue lines indicate the communication links as well as the geometric pathways, together forming a topological map for navigation. There are three sorts of sensor nodes deployed:

Goal nodes: Goal nodes are mounted at designated locations that can be queried as a destination, e.g. (node 21: "store room"), (node 22: the office of "John"), (node 23: "exit"), (node 15: "laboratory 1"), (node 17: the office of "Jerry"), (node 25: "laboratory 2"), and a group of research students in the laboratories, for example (node 11: "Alice").

Junction nodes: Junction nodes are mounted at the junctions of passages for the purpose of routing, e.g. node 2, node 11, node 15 and node 18.

Way nodes: The remaining nodes are way nodes to support navigation. They are placed by following a sensor deployment strategy, e.g. to achieve minimum set of sensors with bounded tracking errors [33].

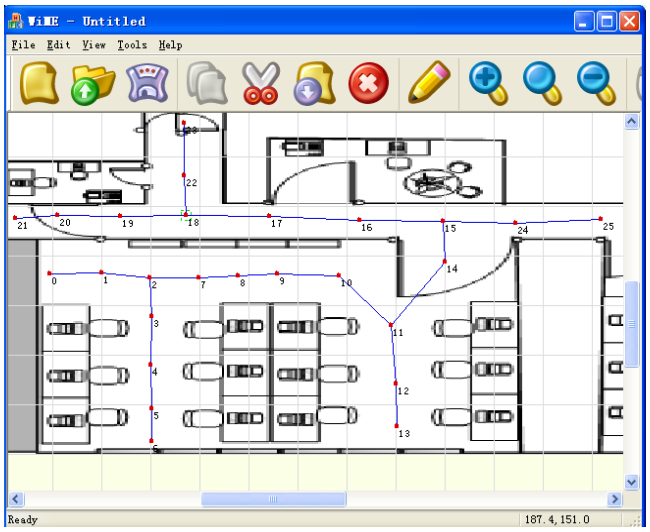

Fig. 2. Map configuration software for sensor node deployment, where wireless visual sensors are in red and communication links are in blue.

With the aid of the routing tables, all wireless visual sensors are organized into a network for navigation. Each sensor can answer queries about direction, such as "what is the best way to the John's office from here?". This will be discussed in the following sections.

III. Navigation by a Multiple Bloom-filter

Global routing or global path planning is the first task of navigation. It is located at the top of a tiered navigation architecture[1]. Given a destination, a vehicle needs to know the most cost-effective sequence of abstract locations leading to it. It is traditionally implemented by a centralized search in a topological map. For navigation by distributed sensors, an intuitive approach is to adopt the routing techniques used in communication networks[34], e.g. searching by broadcasting potential values[35] or flooding a navigation field to a destination[19]. The drawback of these methods appears to be with slow response, heavy communication burden and high memory requirements. A wireless sensor network has limited on-board capacity and low communication rates. It is expected that routing can be accomplished with less on-line communication and computation, at the same time using moderate amounts of on-board memory space. A technique to improve performance of repetitive-mode queries from vehicles is to pre-compute some information [36]. One extreme is to pre-compute the entire finite set of all possible queries and store them in a routing table, providing rapid access time but with a very large storage overhead. In this section, a multiple Bloom-filter scheme is proposed to provide routing tables using less memory space.

With wireless sensors deployed in an environment for navigation, we assume that the sensors have been organised into a network topologically equivalent to the geometric road map, as Fig.2. The topological map can be represented as a graph of $\{S, E\}$, where $S=\{s_1, s_2,\ldots,s_c\}$ is a collection of sensor nodes in the network; $E = \{e_{i,j} : i \in [1 \cdots c]; j \in [1 \cdots c]; i \neq j\}$ is a collection of edges, i.e. geometric passages, which connect nodes in pairs. For each node $s_i=\{I_i, E_i\}$, I_i is the identity of the i^{th} node and E_i is the set of edges connecting s_i with other nodes. The routing problem is to find a sequence of edges from the current location to a goal with the shortest total distance. When a vehicle is under the control of node s_i, it will query for the next edge $e_{i,j}$ to follow in order to reach desired destination s_g. The query is expected to use a human friendly identity I_g such as "John's office", "David's office", rather than an artificial code, which will allow a user without any knowledge about the coding method to use the navigation service. It also simplifies the program needed for a vehicle or PDA to access the wireless sensor network.

Due to limited communication and computation power of each wireless node, an on-line shortest route search should be avoided. Therefore, we take a routing table approach, where routing tables are pre-computed by the Dijkstra's search algorithm. A routing table $T_{i,j}$ in a node s_i lists all destinations that can be reached with the shortest distances by following the edge e_{ij}. Therefore, each node has the same number of routing tables as its edges. Taking the map shown in Fig. 2 as an example, we will generate 3 routing tables for node 18 as

below:
$T_{18\to19}$={"store room" }, $T_{18\to22}$={"John", "exit"}, $T_{18\to17}$={"Jerry", "laboratory 1", "laboratory 2", "Alice" and all other students in laboratory 1, all students in laboratory 2}.

With the routing tables saved in each node, a robot at the location of one node can query which passage it should take in order to reach a given destination with the shortest distance. The node can answer by determining which routing table a queried string's membership belongs to. It uses memory to avoid an on-line search. For example, if a robot near node 18 is looking for "Alice", it can query the node for a navigation service. Node 18 checks the tables $T_{18\to17}$, $T_{18\to19}$, and $T_{18\to22}$ to determine which edge should be taken and finds out that "Alice" is in $T_{18\to17}$. The robot will then be directed from node 18 to node 17. Therefore, the routing table of an edge lists all possible destinations by taking the edge. However, the required memory to store those destinations could be huge for a big map, which is often not feasible for a sensor node with scarce on-board memory.

A Bloom-filter can be used as a routing table. To represent n random strings into a table T, k independent hash functions are used to generate digital fingerprints in T, which map elements belonging to the path to k integers in $[0,m]$ and the corresponding bits in bit-vector T are set as shown in Fig. 3. If there are L branched paths for a node, L Bloom-filters, forming a multiple Bloom-filter, are needed for routing.

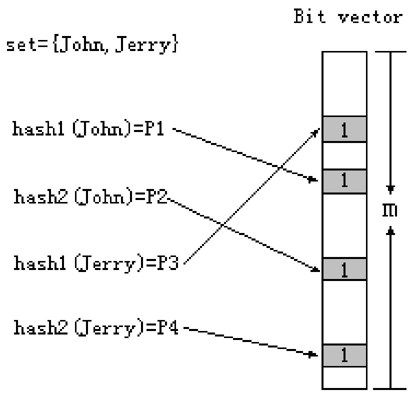

Fig. 3. A routing table in the Bloom-filter for a path with two members, John and Jerry

A vehicle can query the multiple Bloom-filter in the sensor node to determine which path to take. If any of the bits in the Bloom-filter of a path are not set, the corresponding path should definitely not be taken. If all of the bits for a path are set, the vehicle *may* take the path. There is a non-zero probability that the decision is wrong because a bit could be set by other elements when they are hashed. This is known as a *false positive*. The advantage of using Bloom-filters as routing tables can be clarified by considering its compression capability, from n elements with any string length to m bits, and its high query efficiency, from an n elements' search to a k hash functions' check. However, there is a trade-off between the compression rate and the false positive rate, which is dependent on the element number n, the hash function number k and the vector length m[28]:

$$p_{fp} \approx (1-e^{-nk/m})^k \tag{1}$$

The minimum false positive probability can be obtained as $p_{fp}=1/2^k$, when $k=(m/n)\cdot\ln 2$. (2)

Example: For a path with 1000 locations and 4 hash functions, the optimized table length is m=5771bits, i.e. 721 bytes with a false positive rate of p_{fp}=6.25%. It can be seen that the required memory space is reasonable for a wireless sensor to store the routing information of a medium-sized map.

The probability of false positives can be further reduced by a multi-hop approach, where a query is forwarded to the next node in the path to double check the membership. If we use independent hash functions in different nodes, the probability of a false positive decreases as a power of hops $p_{fp}^{\ hops}$. Taking the previous example with p_{fp} =6.25%, we have p_{fp}=0.39% for 1 hop, p_{fp}=0.024% for 2 hops, etc. The false positive rate can be reduced to a very low level with only a few hops and the correct path branch can be determined.

IV. Design of a Multiple Bloom-filter Based on Error Expectation

In section 3, a routing scheme using a multiple Bloom-filter is presented. An immediate question is how to determine the number of hash functions and the table length of each Bloom-filter. From equation (2), the optimal number of hash functions should be selected based on a desired p_{fp}. However, the number of hash functions in a multiple Bloom-filter is fixed for several independent tables and often specified beforehand to simplify implementation, e.g. for reducing on-line computation required for hash checks. Therefore, in order to ensure that queries have a false positive rate of less than p_{fp}, adjusting the number of hash functions is often not feasible. We have to change the length m of each Bloom-filter for a given p_{fp} to carry out a sub-optimal design. From equation (1), we have

$$m=\frac{-nk}{\ln(1-\sqrt[k]{p_{fp}})} \tag{3}$$

From equation (2), the optimal $k_0=-\log_2 p_{fp}$ and $m_0=nk_0/\ln 2$, the actual length in comparison with the optimal length can be obtained as below.

$$m=\frac{k}{k_0}\cdot\frac{\ln\frac{1}{2}}{\ln(1-(\frac{1}{2})^{\frac{k_0}{k}})}\cdot m_o \tag{4}$$

Fig.4 shows the relation. It can be observed that the table length of a Bloom-filter has to be longer than the optimal m_0 in order to achieve the same p_{fp} if a given k is not equal to k_0.

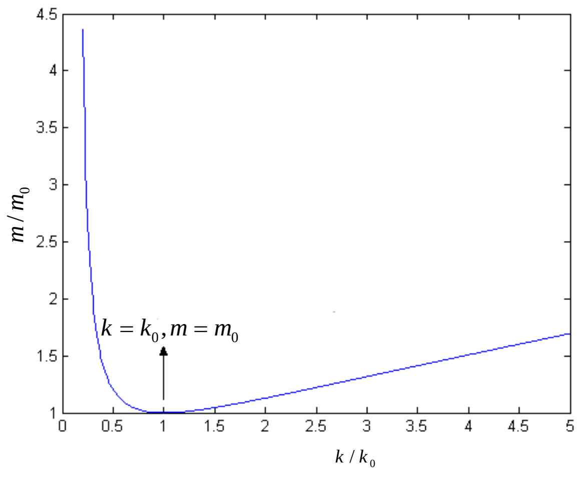

Fig. 4. Table length ratio m/m_0 vs hash number ratio k/k_0, where m_0 and k_0 are optimal

Consider node *s* has *L* edges and as a result there are *L* Bloom-filters in it. Each Bloom-filter is expected to encode $n(i)$, i=1,…,L, nodes with altogether $n = \sum_{i=1}^{L} n(i)$ nodes in a multiple Bloom-filter. In order to design table lengths $m(i)$ for *L* Bloom-filters, an intuitive route is to select $m(i)$ to achieve a common false positive rate p_{fp} for all branched paths based on (3). It will be shown that this approach may result in biased false positive errors for different edges. In a multiple Bloom-filter, the numbers of encoded nodes $n(i)$, i=1,…,L, in individual Bloom-filters could be quite different, e.g. some edges may lead to very few nodes but the others may lead to many. However, all Bloom-filters or sub-tables have to pass an equal number of queries, which could be legal or illegal for an edge, e.g. a query to a location encoded in another table. Equal p_{fp} means equal possibility of errors occurring in all paths, regardless of how many nodes are resident in a path. As a result, a path with fewer nodes will have a higher chance of being wrong than one with a dense population of nodes. This can be verified by its error expectation. Considering there are altogether *N* possible queries, the relative error expectation for edge *i* due to *N* queries is:

$$E_{EPN}(i) = \frac{(N - n(i)) \cdot p_{fp}}{n(i)} \tag{5}$$

It can be found that an equal rate of false positive, p_{fp}, results in a higher relative error for a Bloom-filter with a smaller value of $n(i)$ than one with larger $n(i)$. It could be even worse if all queries are legal, i.e. N=n, and the distribution of nodes in different paths is not uniform. In this case, a path with very few nodes, i.e. $n(i)$<<N, will have much higher error expectation than a path which is densely lined with nodes, i.e. $n(i)$≈N, due to the term N-$n(i)$ in (5). The overall error expectation for checking *L* tables in a multiple Bloom-filter can be calculated as:

$$E_d = \sum_{i=1}^{L} \left(N - n(i)\right) \cdot p_{fp} = \left(L \cdot N - n\right) \cdot p_{fp} \tag{6}$$

Therefore, we need to take the error expectation as the design criterion, as opposed to the false positive rate in order to achieve a uniform relative error expectation for all paths. This is important for navigation, where the routing by any path should have equal relative risk. The communication bandwidth along all paths can then be fairly allocated if the multi-hop approach proposed in section 3 is used for coping with false positives.

In order to ensure a uniform relative error expectation for all edges, we propose a biased approach that assigns a different false positive rate, $p_{fp}(i)$, i=1..L, to each edge *i*, for example, an edge with fewer nodes will be given a lower possibility. Let

$$p_{fp}(i) = p_{fp} / t \quad \text{with} \quad t = \frac{N - n(i)}{N - \bar{n}} \cdot \frac{\bar{n}}{n(i)}, \tag{7}$$

where $\bar{n} = n / L$ represents the average number of nodes in a branch.

From (7), if $n(i) \le \bar{n}, t \ge 1$ and $p_{fp}(i) \le p_{fp}$, a lower false positive rate is set; if $n(i) > \bar{n}, t < 1$ and $p_{fp}(i) > p_{fp}$, a higher false positive rate is set. We expect that this modification can make all edges exhibit an equal relative error expectation regardless of how many nodes in a Bloom-filter.

Theorem 1: If $n(i) \le N / \left(p_{fp} \cdot \left(\frac{N - \bar{n}}{\bar{n}} \right) + 1 \right)$, i) $p_{fp}(i)$ defined in (7) can be used as a false positive probability for a multiple Bloom-filter design; ii) all edges will have equal relative error expectation; iii) the overall error expectation of the new design will be the same as the equal p_{fp} design shown in (6).

Proof: If the $p_{fp}(i)$ in (7) can be used as a probability, $p_{fp} \le t$ must be true. From (7), it can be obtained that $n(i) \le N / \left(p_{fp} \cdot \left(\frac{N - \bar{n}}{\bar{n}} \right) + 1 \right)$. This proves i).

The relative error expectation of edge *i* due to *N* queries can be calculated by using the $p_{fp}(i)$:

$$E'_{EPN}(i) = \frac{\left(N - n(i)\right) \cdot p_{fp}(i)}{n(i)} = \frac{\left(N - n(i)\right)}{n(i)} \cdot \frac{p_{fp}}{t} = \frac{N - \bar{n}}{\bar{n}} \cdot p_{fp}$$

which is a constant for any $n(i)$ in an edge. This proves ii).

The overall error expectation of the new design for all edges can be obtained as:

$$E'_d = \sum_{i=1}^{L} \left(N - n(i)\right) \cdot p_{fp}(i) = \sum_{i=1}^{L} \left(N - n(i)\right) \cdot \frac{p_{fp}}{t} = \sum_{i=1}^{L} \frac{(N - \bar{n})}{\bar{n}} \cdot n(i) \cdot p_{fp}$$
$$= \frac{(N - \bar{n})}{\bar{n}} \cdot L \cdot \bar{n} \cdot p_{fp} = \left(L \cdot N - n\right) \cdot p_{fp}$$

which is identical to the expectation of the equal p_{fp} based design in (6). This proves iii).

Therefore the proposed variable false positive rate (7) for the multiple Bloom-filter design balances the relative error expectations of all edges but without compromising overall error expectation. Since in most applications N>>n and p_{fp}<<1, the condition $n(i) \le N / \left(p_{fp} \cdot \left(\frac{N - \bar{n}}{\bar{n}} \right) + 1 \right)$ is often true. However, it would be difficult to estimate the number of all the possible queries, *N*. In real applications considering a large *N*, (7) can be simplified to:

$$p_{fp}(i) = p_{fp} \cdot \frac{n(i)}{\bar{n}} \tag{8}$$

Therefore, the false positive probability of each edge can be easily obtained from the number of resident nodes in the edge.

V. Memory Usage of the Error Expectation Based Multiple Bloom-filters

Wireless sensor nodes often have less memory capacity. A key factor to be concerned is the amount of memory usage.

Assume there are *L* Bloom-filters in a sensor node and each Bloom-filter encodes *n*(*i*), *i*=1,…,*L*, members. Following the conventional design, we specify the table lengths so that all tables achieve an equal false positive rate, p_{fp}. From (3), we know the table length of the *i*th Bloom-filter should be:

$$m'(i) = \frac{-n(i)k}{\ln(1-\sqrt[k]{p_{fp}})} \tag{9}$$

The overall table length of the multiple Bloom-filter becomes:

$$m_{FPB} = \sum_{i=1}^{L} \frac{-n(i)\cdot k}{\ln\left(1-\sqrt[k]{p_{fp}}\right)} = \frac{-n\cdot k}{\ln\left(1-\sqrt[k]{p_{fp}}\right)} \tag{10}$$

If we follow the new design based on equal error expectation, the p_{fp}(i) will be adjusted by (8). The overall table length becomes:

$$m_{EEB} = \sum_{i=1}^{L} \frac{-n(i)\cdot k}{\ln\left(1-\sqrt[k]{p_{fp}[n(i)/\bar{n}]}\right)} \tag{11}$$

The following theorem tells us the memory usage of the proposed method in comparison with the conventional design.

Theorem 2: The table length (11) of a multiple Bloom-filter to achieve equal error expectation is no longer than that of the false positive rate based design in (10). They have equal length only when $n(1) = n(2) = \cdots = n(L) = \bar{n}$.

Proof: Considering optimization of (11) with respect to variables of *n*(*i*), *i*=1…*L*, subject to the constraint of $\sum_{i=1}^{L} n(i) = n$, we can define the associated Lagrange function as:

$$U(n(i), i=1\cdots L) = \sum_{i=1}^{L} \frac{-n(i)\cdot k}{\ln\left(1-\sqrt[k]{p_{fp}\cdot[n(i)/\bar{n}]}\right)} + \lambda\left(\sum_{i=1}^{L} n(i) - n\right) \tag{12}$$

where λ is a Lagrange multiplier.

Let $g(n(i)) = 1-\sqrt[k]{p_{fp}\cdot[n(i)/\bar{n}]}$, *i*=1…*L*, the optimization can be solved by the partial derivatives of *U* with respect to *n*(*i*) and λ:

$$\begin{cases} \dfrac{\partial U}{\partial n(i)} = -\dfrac{k\cdot g(n(i))\cdot \ln(g(n(i))+1-g(n(i))}{g(n(i))\cdot \ln^2(g(n(i)))} + \lambda = 0, i=1\cdots L. \\ \sum_{i=1}^{L} n(i) - n = 0 \end{cases} \tag{13}$$

From (13), there is a critical point at $n(1) = n(2) = \cdots = n(L) = \bar{n}$. Substituting it into (11), we have an extremum of *U*:

$$m_{EEB}(n(i) = \bar{n}) = \sum_{i=1}^{L} \frac{-n(i)\cdot k}{\ln(1-\sqrt[k]{p_{fp}})} = -\frac{n\cdot k}{\ln(1-\sqrt[k]{p_{fp}})}.$$

It has the same length as the conventional design in (10). We need to further prove it is the maximum for all *n*(*i*). The second derivative of m_{EEB}(*n*(*i*)) in (11) with respect to *n*(*i*) can be obtained as:

$$\frac{\partial^2 m_{EEB}}{\partial n(i)^2} = \frac{\dfrac{\partial g}{\partial n(i)} f(g)}{g^2\cdot \ln^2(g)}, \tag{14}$$

where $f(g(n(i)) = 1 + k\cdot g(n(i)) + 2\dfrac{1-g(n(i))}{\ln(g(n(i)))}$.

Because $\dfrac{\partial g}{\partial n(i)} = -\left[p_{fp}\cdot\dfrac{n(i)}{\bar{n}}\right]^{-\frac{1-K}{K}} \cdot \dfrac{p_{fp}}{\bar{n}} \le 0$, the extremum must be the maximum if *f*(*g*)≥0.

Because of *g*≥0 and *k*≥1, we have

$$f(g(n(i)) \ge 1 + g(n(i)) + 2\frac{1-g(n(i))}{\ln(g(n(i)))} \equiv f_1(g) \tag{15}$$

Taking its derivative with respect to *g*, we have

$$\frac{\partial f_1}{\partial g} = 1 + 2\frac{-\ln(g)-(1-g)/g}{\ln^2(g)} = \frac{f_2(g)}{\ln^2(g)}, \tag{16}$$

where $f_2 = \ln^2(g) - 2\ln(g) + 2 - 2/g$.

Further,

$$\frac{\partial f_2}{\partial g} = \frac{f_3}{g^2}, \text{where } f_3 = 2\cdot g\cdot \ln(g) - 2g + 2. \tag{17}$$

$$\frac{\partial f_3}{\partial g} = 2\cdot \ln(g)\cdot \tag{18}$$

We will track back now to prove $\dfrac{\partial^2 m_{EEB}}{\partial n(i)^2} \le 0$ for $0 \le g(n(i)) \le 1$, because of $g(n(i)) = 1-\sqrt[k]{p_{fp}(i)}$.

From (18), we have $\dfrac{\partial f_3}{\partial g} \le 0$ for $0 \le g \le 1$. Hence f_3 is non-increasing. From f_3(1)=0, we know f_3(*g*)≥0 and therefore $\dfrac{\partial f_2}{\partial g} \ge 0$ for $0 \le g \le 1$. It indicates f_2 is non-decreasing for *g* from 0 to 1. From f_2(1)=0, we have f_2(*g*)≤0 from 0 to 1, hence $\dfrac{\partial f_1}{\partial g} \le 0$ for $0 \le g \le 1$. f_1 is non-increasing for *g* from 0 to 1.

Because $\lim_{g\to 1} f_1(g) = 2 + 2\cdot \lim_{g\to 1}\dfrac{-1}{1/g} = 0$, we have f_1(*g*)≥0 for *g* from 0 to 1, and thus *f*(*g*) ≥0 in (15). Substituting it into (14), we know $\dfrac{\partial^2 m_{EEB}}{\partial n(i)^2} \le 0$ for $0 \le g \le 1$. The critical point at $n(1) = n(2) = \cdots = n(L) = \bar{n}$ reaches the maximum value of m_{EER}, which is the table length of the conventional design. For any other *n*(*i*), *i*=1…*L*, m_{EER} in (11) is always less than m_{FPB} in (10).

From Theorem 2, a multiple Bloom-filter using the expectation based design will not only achieve uniform relative error expectation, but will also save memory usage in comparison with the false positive rate based design.

VI. Experimental Results

A practical WiME system was developed with wireless visual sensors mounted on the ceiling of a building. Software packages and communication protocols were developed for wheelchair navigation, with the proposed multiple Bloom-filter as a global path planner. A demonstration video of wheelchair navigation in the building can be found in [31].

In order to evaluate the performance of the proposed error expectation based multiple Bloom-filter design, we randomly generated a topological map with altogether n=1000 nodes. With k=4 hash functions and a desired false positive rate p_{fp}=0.01, the false positive rate based(FPB) and the error expectation based(EEB) multiple Bloom-filters were developed. For a node with 4 branches (Path1, Path2, Path3 and Path4), which have 23, 193, 332 and 452 nodes respectively, the two multiple Bloom-filters were queried by 4 groups of 10^6 random strings. Table 1 and 2 show the errors for each group of queries.

Table 1: Errors Using the False Positive Rate Based Design

Group	1	2	3	4	Average Errors	Relative Errors
Path 1	9987	10013	9979	10025	10001	434.83
Path 2	9976	9984	9981	10012	9988	51.75
Path 3	10014	9985	9998	10002	9999	30.12
Path 4	9982	9986	10011	9993	9993	22.11

Table 2: Errors Using the Error Expectation Based Design

Group	1	2	3	4	Average Errors	Relative Errors
Path 1	937	945	914	909	926	40.26
Path 2	7702	7734	7724	7727	7722	40.00
Path 3	13178	13297	13206	13308	13247	39.90
Path 4	17882	18123	18105	17934	18011	39.84

From Table 1, the conventional false positive rate based design results in much higher relative errors for a path with fewer nodes than a path with more nodes. For example, path 1 exhibits a 434.83 relative error. From the working process of a multiple Bloom-filter proposed in section 2, any query needs to be checked by all the Bloom-filters. Therefore, a path with fewer nodes will suffer more errors from queries which do not belong to it. The errors will cause more than one Bloom-filter to pass the hash check. As a result, the multi-hop checks have to be carried out for further confirmation, which increases the communication cost in paths with fewer nodes. The proposed expectation based design guarantees a uniform distribution of relative errors. In Table 2, all paths show a similar amount of relative errors, about 40, which means an equal risk for any path. The comparison results verify Theorem 1.

In terms of overall memory usage, we randomly generated 1000 topological maps with 1000 nodes each. The table lengths by using the false positive rate based design and the error expectation based design are shown in Fig. 5.

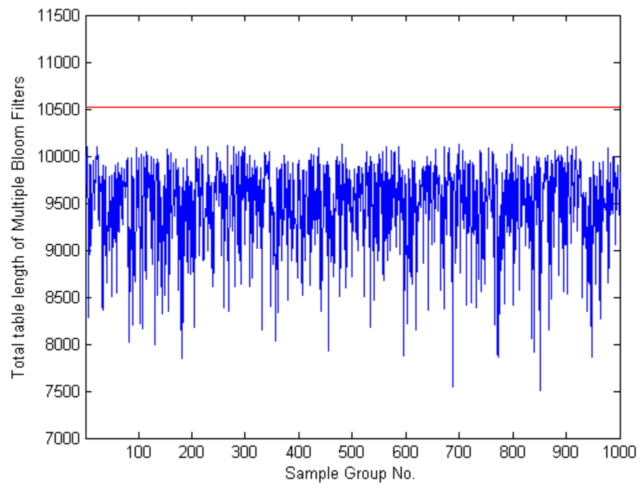

Fig. 5. Memory usages of the EEB design(blue) and the FPB design(red)

It is clear that the proposed method outperforms the conventional design, which verifies Theorem 2. To have a quantitative comparison, we list the memory usage of the Bloom-filter based routing tables and the routing table without using Bloom-filters for the 4 groups of results in Table 3.

Table 3: Memory Usage of routing Tables(Kbits)

Group	*1*	*2*	*3*	*4*	*Percentage*
No Bloom-filter	48	48	48	48	100%
FPB Bloom-filters	10.52	10.52	10.52	10.52	21.9%
EEB Bloom-filters	9.449	9.452	9.483	9.462	19.7%

From Table 3, the memory usage is reduced to 21.9% by introducing Bloom-filters as routing tables, where the Bloom-filters are designed to achieve equal false positive rates. The memory usage is further reduced to 19.7% by using the error expectation based design, which is around 10.1% of the false positive rate based design.

VII. Conclusions

A multiple Bloom-filter technique has been proposed for storing a large amount of routing information into a wireless sensor node, where energy and memory are scarce resources. An error expectation based design, instead of the typical false positive rate based design, was proposed to achieve a uniform false positive expectation for every sub-table. As a result, communication costs, as well as power consumption, were balanced to be in proportion to the number of resident nodes in a path. The table length of a multiple Bloom-filter was shown to be shorter than the conventional design. Therefore, memory usage in the wireless node is reduced. The proposed method has been practically implemented in the WiME project for wheelchair navigation using wireless visual sensors, and can be taken as a general approach for navigation routing using a distributed sensor network. For example, the increasing number of CCTV cameras can also be used to provide navigational services to vehicles on a motorway or wheelchairs in a building.

REFERENCES

[1] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge: USA: MIT Press, 2004.

[2] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization," presented at Proc. IEEE International Conference on Robotics and Automation (ICRA-2000), San Francisco, CA, 2000.

[3] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, pp. 99-141, 2001.

[4] H. Baldus, K. Klabunde, and G. Muesch, "Reliable set-up of medical body-sensor networks," *Lecture Notes in Computer Science*, vol. 2920, pp. 353-363, 2004.

[5] T. Wark, P. Corke, P. Sikka, L. Klingbeil, Y. Guo, C. Crossman, P. Valencia, D. Swain, and G. Hurley, "Transforming agriculture through pervasive wireless sensor networks," *IEEE Pervasive Computing*, vol. 6, pp. 50-57, 2007.

[6] Z. Butler, P. Corke, R. Peterson, and D. Rus, "From robots to animals: virtual fences for controlling cattle," *International Journal of Robotics Research*, vol. 25, pp. 485 - 508, 2006.

[7] K. Martinez, R. Ong, and J. Hart, "Glacsweb: a sensor network for hostile environments," presented at The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, USA, 2004.

[8] R. Poovendran, "Cyber-physical systems: close encounters between two parallel worlds," *Proceedings of the IEEE*, vol. 98, pp. 1363-1366, 2010.

[9] M. D. Ilic, L. Xie, U. A. Khan, and J. M. F. Moura, "Modeling of future cyber-physical energy systems for distributed sensing and control," *IEEE Trans on Systems, Man and Cybernetics, Part A*, vol. 40, pp. 825-838, 2010.

[10] E. Guizzo, "Robots with their heads in the clouds," in *IEEE Spectrum*, 2011.

[11] D. Tacconi, D. Miorandi, L. Carreras, F. Chiti, and R. Fantacci, "Using wireless sensor networks to support intelligent transportation systems," *Ad Hoc Networks*, vol. 8, pp. 462-473, 2010.

[12] V. P. Srini, "A vision for supporting autonomous navigation in urban environments," *Computer*, vol. 39, pp. 68-77, 2006.

[13] A. Mpitziopoulos, C. Konstantopoulos, D. Gavalas, and G. Pantziou, "A pervasive assistive environment for visually impaired people using wireless sensor network infrastructure," *Journal of Network and Computer Applications*, vol. 34, pp. 194-206, 2011.

[14] M. Li, Y. Liu, J. Wang, and Z. Yang, "Sensor network navigation without locations," presented at Proceedings 28th IEEE International Conference on Computer Communications, Rio de Janeiro, Brazil, 2009.

[15] D. Estrin, D. Culler, and K. Pister, "Connecting the physical world with pervasive networks," *IEEE Pervasive Computing*, vol. 1, pp. 59-69, 2002.

[16] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, pp. 102-114, 2002.

[17] X. Lin, Y. Kwok, and H. Wang, "Energy-efficient resource management techniques in wireless sensor networks," in *Guide to Wireless Sensor Networks*, S. Misra, Ed. London: Springer-Verlag, 2009.

[18] A. Rogers, E. David, and N. R. Jennings, "Self-organized routing for wireless microsensor networks," *IEEE Trans on Systems, Man and Cybernetics, Part A*, vol. 35, pp. 349-359, 2005.

[19] M. A. Batalin and G. S. Sukhatme, "Mobile robot navigation using a sensor network," presented at IEEE International Conference on Robotics and Automation, New Orleans, LA, 2004.

[20] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422-426, 1970.

[21] G. Antichi, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, "Counting bloom filters for pattern matching and anti-evasion at the wire speed," *IEEE Network*, vol. 23, pp. 30-35, 2009.

[22] H. Lim and S. Y. Kim, "Tuple pruning using Bloom filters for packet classification," *IEEE Micro*, vol. 30, pp. 48-59, 2010.

[23] M. D. McIlroy, "Development of a Spelling List," *IEEE Transactions on Communications*, vol. 30, pp. 91-99, 1982.

[24] E. H. Spafford, "Opus: preventing weak password choices," *Computer and Security*, vol. 11, pp. 273-278, 1992.

[25] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 281-293, 2000.

[26] S. Czerwinski, B. Y. Zhao, T. Hodes, A. D. Joseph, and R. Katz, "An architecture for a secure service discovery service," presented at Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99), New York, 1999.

[27] D. Guo, Y. Liu, X. Li, and P. Yang, "False negative problem of counting Bloom filter," *IEEE Knowledge and Data Engineering*, vol. 22, pp. 651-664, 2010.

[28] M. Mitzenmacher, "Compressed bloom filters," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 604-612, 2002.

[29] B. Xiao and Y. Hua, "Using parallel Bloom filters for multiattribute representation on network services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 20-32, 2010.

[30] S. C. Rhea and J. Kubiatowicz, "Probabilistic location and routing," presented at Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA, 2002.

[31] P. Jiang, Z. Feng, Y. Cheng, Y. Ji, X. Wang, F. Tian, J. Baruch, and F. Hu, "A mosaic of eyes for wireless navigation and control of mobile robots," *IEEE Robotics and Automation Magazine*, vol. 18, pp. 104 - 113, 2011.

[32] Y. Cheng, P. Jiang, and Y. F. Hu, "A distributed snake algorithm for mobile robots path planning with curvature constraints," presented at IEEE International Conference on Systems, Man and Cybernetics, Singapore, 2008.

[33] C. H. Lin and C. T. King, "Sensor-deployment strategies for indoor robot navigation," *IEEE Trans on Systems, Man and Cybernetics, Part A*, vol. 40, pp. 388-398, 2010.

[34] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, pp. 6-28, 2004.

[35] Q. Li and D. Rus, "Navigation protocols in sensor networks," *ACM Trans. on Sensor Networks*, vol. 1, pp. 3-35, 2005.

[36] Y. Saab and M. VanPutte, "Shortest path planning on topographical maps," *IEEE Trans on Systems, Man and Cybernetics, Part A*, vol. 29, pp. 139-150, 1999.