Opportunistic Detection of Relative Mobility in Wireless Sensor Networks

Ricardo Simon Carbajo, Meriel Huggard, Ciarán Mc Goldrick
School of Computer Science and Statistics
Trinity College Dublin (Ireland)
Email: {carbajor, Meriel.Huggard, Ciaran.McGoldrick}@sccs.tcd.ie

***Abstract*—In highly mobile ad hoc networks the neighbourhood status evolves rapidly. Routing tables which are populated with an "out of date" view of the neighbourhood produce an overhead in the routing process which can affect the QoS of the communication and the stability of the network interconnectivity. To maintain correct network state information, periodic "Hello" messages are often employed. In many scenarios these may be expensive in terms of communication and energy. Providing a node with an ability to estimate mobility has been proposed as a mechanism for regulating the transmission of discovery "Hello" messages and for prompt updates of its neighbourhood view.**

In this paper, a mechanism for gauging the relative mobility of a node with respect to its neighbourhood is presented for mobile Wireless Sensor Networks. The algorithm employs a set of Bloom filters to encapsulate opportunistically eavesdropped data from neighbours. A set of mobility states is defined using a model based on the number of neighbours recently cached, a membership comparison of Bloom filters, and a probabilistic expression. The mechanism detects changes in the neighbourhood using only the available information. The evaluation of the system has been performed (using a gradient-based routing protocol) in the TinyOS simulator, Tossim.

***Index Terms*—Wireless Sensor Networks; Mobility; Neighbourhood Detection; Tossim; TinyOS; Bloom Filter.**

I. Introduction

Dynamically changing ad hoc networks commonly employ a proactive approach to neighbour discovery, where nodes periodically send a "hello" message as an element in the timely maintenance of neighbourhood connectivity information. Thus the network can more rapidly establish routes for data transmissions.

In many scenarios, where wireless transmission ranges are of the order of 50 to 100 meters, the status of the network does not change particularly rapidly. Nodes move at different speeds and some of them will only have local mobility. Moreover a set of nodes, which may be defined as a cluster, may not be moving with respect to each other - even though they are physically changing position, i.e. a relative mobility scenario. If we consider that many applications in ad hoc networks, more specifically in wireless sensor networks, transmit data intermittently then the volume of control traffic required to maintain the network status information, using periodic "hello" messages, is highly expensive in terms of message and energy costs. Therefore in scenarios where the network includes some transient nodes, and data communication will be infrequent, energy may be conserved by opportunistically updating only areas with activity, and avoiding the proactive beacon approach.

In this paper, a memory efficient mechanism to detect if a node has changed its relative position with respect to its neighbourhood, i.e. relative mobility, is presented. The mechanism alerts the system of a possible change in neighbourhood based on eavesdropped data. The approach builds a set of temporal shift Bloom filters [1] to store historical overheard information. A probabilistic technique is employed to assess the mobility state of a node according to the information cached in the set of Bloom filters. A recursive estimation approach enhances the accuracy of the mobility assessment by incorporating previous expressions of the likelihood of mobility. The model accommodates most of the uncertainty scenarios where wireless connectivity might be temporarily disrupted and where the relative position of the node(s) may or may not have changed. The mechanism has been tested with a dynamic routing protocol designed for Peer-to-Peer communication in Mobile Wireless Sensor Networks. The routing protocol employs gradient-based techniques to populate routing tables and implements effective local repair and caching mechanisms for reliability. One of the goals is to avoid proactive periodic beacon messages by assuming that areas with no activity do not require recurrent updating, whilst areas with activity opportunistically exploit communication to update connectivity status.

In the following section, an overview of existing approaches for assessing a nodes's mobility are provided. Section III explains the functioning of the mobility detection mechanism, describes the Bloom filter, and presents the numerical basis for the mobility detection model. An evaluation of the model in the TOSSIM simulator, under different mobility scenarios, is presented to validate the concept. The Conclusions and Future Work section summarises our findings, system improvements and planned practical deployments.

II. Mobility Estimation and Detection

Mobility estimation and detection techniques have been well studied for mobile ad hoc networks. These focus on the prediction of the status of the neighbourhood and improve routing efficiency (in terms of transmission) [2]. Some of these approaches employ a collaborative approach between nodes requiring the use of explicit data communication. These mechanisms make use of distributed information to estimate

978-1-4244-9229-9/10/$26.00 ©2010 IEEE

the level and the direction of mobility of a local node or a neighbour. Many of these approaches exploit non-random mobility patterns and are based on the processing of metrics from the physical and MAC layer, such as RSSI, and the analysis of explicit and implicit packet information received from other nodes in the network. Some of these algorithms depend on historical data for retro-analysis. Moreover, the use of specialised or dedicated layers to estimate mobility might require the use of special packets or the piggyback of localization information to update other nodes.

Much of the work described above focused on estimating the physical mobility of a node and the path being followed. However, the study of the mobility of a node with respect to its neighbourhood, i.e. relative mobility, has recently attracted attention, e.g. [3], [4]. In [5] the problems experienced by routing protocols when a neighbour node disappears and the benefit of detecting that promptly are discussed. A lightweight Bloom filter based mechanism is employed to replicate information of the presence of nodes by using beacon messages. In [3], the authors propose a dynamic timer adjustment to reduce the overhead of "Hello" control packets when discovering the neighbourhood. By measuring the link change rate with respect to its neighbourhood, the mobility status of the node can be estimated and the "Hello" timer tuned. In order to make more accurate forwarding decisions, Xu et al. [6] propose a neighbourhood tracking scheme. Whenever a node needs to route a packet, the view of the neighbourhood is constructed and updated using past location information. This employs mobility prediction models to estimate the positions of neighbourhood nodes at the time when the packet will be forwarded.

In the area of Wireless Sensor Networks, algorithms need to operate in networks with a mix of static and mobile low-power nodes where no location information may be available. Factors like the unpredictable and constantly changing wireless medium behaviour and the effects of the mobility of the nodes, can affect the neighbourhood connectivity status of a node even if there are no relative position changes. Moreover, nodes can travel in groups at constant or individual speeds. Taking this into account, the authors in [4] propose an approach to detect the mobility status of a node's neighbourhood based on calculations from messages received from neighbours. Their proposed algorithm, the SDMS (Self Detection Mobility Status), capitalizes on the definition of mobility states according to the analysis of neighbourhood-related local indicators. However, the algorithm requires neighbours to announce their presence regularly and, the higher the rate of announcements, the greater the accuracy of the algorithm.

Herein, the mobility of a node is assessed with respect to its neighbourhood status through leveraging opportunistic communication. Our approach avoids periodic "Hello" beacon messages and serves as a mechanism to determine when the node is changing its relative position within its neighbourhood. The algorithm, which defines a set of mobility states, works independently and can be incorporated into any routing protocol for mobility detection purposes.

III. Relative Mobility Detection Model

A. Bloom Filter Functionality

A Bloom filter [1], is a probability-based compressing structure which identifies whether a data item is a member of the filter or not. In other words, it stores the presence of the item in the filter structure, rather than storing the data item itself. The Bloom filter structure is represented as an array of bits. By switching bits from 0 to 1 in a set of strategic positions, an element is declared as a member of the filter. Every combination of positions represents a data item stored. In order to calculate the combination of positions which represent a particular data item, hash functions are employed. Each hash function is applied to the data item producing an integer number which is trimmed/scaled to the number of bits in the Bloom filter - thereby generating a position in the bitvector. Applying X hash functions to the same data item produces Y positions in the bitvector. The collection of hash functions applied over a data item will always produce the same combination of positions for every data item. By the same token, a data item can be hashed to check whether all its positions in the Bloom filter are set to 1, i.e. to check membership.

The Bloom filter structure has a drawback. The higher the number of data items to be stored in the Bloom filter, the higher the probability of obtaining false positives. A false positive occurs when a data item has not been stored in the Bloom filter, but the positions which correspond to the data item are set to 1 by the other data items stored. The selection of the number of hash functions, together with the number of bits of the Bloom filter and the number of elements inserted in the Bloom filter, establish the probability of getting false positives. On the other hand, the Bloom filter guarantees that there will be no false negatives, i.e. if there is not a combination of positions set to 1 which satisfies the hashed data item, then it is guaranteed that the data item was not inserted in the Bloom filter. Bloom filters can easily be compared with a "XOR" bitwise operation and merged with an "OR" bitwise operation.

The probability for a false positive error can be calculated with the next equation:

$$E_{fp} = \left(1 - (1 - 1/m)^{kn}\right)^k \tag{1}$$

being "m" the number of bits in the Bloom filter, "n" the number of elements inserted and "k" the number of hash functions [7]. The error E_{fp} can be minimized for the number of hash functions according to the next equation [7]:

$$k = (m/n) \ln 2 \tag{2}$$

For instance, for a bit array 10 times larger than the number of entries, the probability of a false positive is 1.2% for k=4 hash functions, and 0.9% for the optimum case of k=5 hash functions. If the data set is known "a priori" then the Bloom filter can be designed to avoid false positives.

B. Recursive Mobility Estimation with Temporal-Shift Bloom Filters

The creation of a historical record of eavesdropped neighbour addresses is key to evaluating a node's relative mobility, i.e. changing neighbourhood. The core neighbourhood of a node is formed by a group of static (in relative terms) neighbours. Over time, other nodes can come and go including those which (by being at the edge of the communication range) may have local mobility. Nodes seen for a short period of time can distort this representation. To accommodate this, overheard information is split into a number of consecutive chronological intervals. The time window for overhearing packets, prior to launching the mobility evaluation process, is defined from the average speed and the mean of the transmission range of the nodes. This provides an indication of the time that a moving node would take to leave its neighbourhood area radius. This "Evaluation Time" represents the average time that the node takes to leave the neighbourhood. The mobility evaluation interval is given by the equation:

$$\text{Evaluation Time(s)} = \frac{\text{Transmission Range(m)}}{\text{Speed(m/s)}} \tag{3}$$

In the interval prior to launching the mobility evaluation process, a "split factor" is established to distribute the overheard neighbours into a set of Bloom filters which act as independent consecutive memory structures. The "split factor" is a configuration parameter in the protocol which defines how many "Active" Bloom filters (BF) are employed. Initially, an independent Bloom filter, the "PrimaryBF" (PriBF), will contain the neighbourhood. The initial neighbourhood can be established by explicitly requesting the neighbours to reply or eavesdropping for an initial period of time. The source of an overheard packet is hashed into the Bloom filter. Once the "PrimaryBF" (#PriBF) is populated, every overheard packet will be stored in the "Active" Bloom filters (#BF). Provided the current "Active" Bloom filter is not empty, the time to shift to start storing information in the next filter is calculated using:

$$\text{Shift BF} = \frac{\text{Evaluation Time(s)}}{\text{Split Factor}} \tag{4}$$

If no data communication activity occurs, the evaluation mechanism will not be launched as the "ActiveBF" will be empty. A maximum time is preset such that, if no mobility evaluation process takes place, a "Hello" message discovery process starts. Each ActiveBF and PrimaryBF keeps a counter of the number of different neighbour addresses which have been stored. When the last "ActiveBF" is populated, the evaluation process starts by comparing each of the individual "ActiveBF" with the "PrimaryBF". This comparison calculates the similarity of the "PrimaryBF" and the "Active" Bloom filter as:

The "Similarity" in Equation 5 represents the percentage of neighbours seen by an "ActiveBF" which are core neighbours, i.e. neighbours also stored in the "PrimaryBF". "ActiveBF" (BF) is intended to represent nodes that are mobile or appear within the neighbourhood subsequent to the formation of "PrimaryBF". Then the Similarity seeks to capture an indication of the relative proportion of these nodes within the neighbourhood.

$$\text{Similarity(BF, PriBF, \#BF, \#PriBF)}(\%) = \begin{cases} \dfrac{\text{\# bits in "BF" matching "PriBF" 1's}}{\text{\# bits in "PriBF" set to 1}} & \text{if \#BF} \geq \text{\#PriBF}, \\ \dfrac{\text{\# bits in "BF" matching "PriBF" 1's}}{\text{\# bits in "BF" set to 1}} & \text{if \#BF} < \text{\#PriBF}. \end{cases} \tag{5}$$

In addition, the Plausible Similarity (Pl. Sim.) in Equation 6 indicates the minimum level of "Similarity" that must be achieved for a nodes neighbourhood to remain unchanged. Plausible Similarities are precalculated according to Equation 6 creating a predetermined table model to be queried. A predetermined table model (generated for a maximum of 10 neighbours) with their associated percentages of plausible similarity and thresholds for the cardinality of neighbours in the "ActiveBF" and the "PrimaryBF" has been created (see Figure 1).

$$\text{Plausible Similarity(\#BF, \#PriBF, \#Tolerance)} = \begin{cases} 100 - (\frac{\#Tolerance}{\#BF} \times 100) & \text{if \#BF} \geq \text{\#PriBF}, \\ 100 - (\frac{\#Tolerance}{\#PriBF} \times 100) & \text{if \#BF} < \text{\#PriBF}. \end{cases} \tag{6}$$

In Equation 6, the #Tolerance parameter is used to establish the level of allowable ("plausible") change within the Similarity expression. The Plausible Similarity depends on the relationship between #BF and #PriBF, giving rise to two different formulations of the expression.

Equation 7 establishes criteria which map the numeric Similarity and Bloom Filter measures into Mobility States.

- MOBILITY, mobility is estimated.
- STATIC, node has seen sufficient core neighbours to estimate that there is not mobility.
- UNCERTAIN, there is not enough information to evaluate whether the node is moving.

$$\text{Mobility Evaluation State(Sim., \#BF, \#PriBF, Pl.Sim.)} = \begin{cases} \text{MOBILITY} & \text{if \#BF} \ggg \text{\#PriBF}, \\ \text{MOBILITY} & \text{if (\#BF} \gg \text{\#PriBF) \&\& (Sim.} < 100\%), \\ \text{UNCERTAIN} & \text{if (\#BF} \gg \text{\#PriBF) \&\& (Sim.} = 100\%), \\ \text{STATIC} & \text{if (\#BF} \geq \text{\#PriBF) \&\& (Sim.} \geq \text{Pl.Sim}), \\ \text{MOBILITY} & \text{if (\#BF} \geq \text{\#PriBF) \&\& (Sim.} < \text{Pl.Sim}), \\ \text{STATIC} & \text{if (\#BF} < \text{\#PriBF) \&\& (Sim.} \geq \text{Pl.Sim}), \\ \text{MOBILITY} & \text{if (\#BF} < \text{\#PriBF) \&\& (Sim.} < \text{Pl.Sim}), \\ \text{UNCERTAIN} & \text{if (\#BF} \ll \text{\#PriBF) \&\& (Sim.} \gtrsim \text{Pl.Sim}), \\ \text{MOBILITY} & \text{if (\#BF} \ll \text{\#PriBF) \&\& (Sim.} < \text{Pl.Sim}). \end{cases} \tag{7}$$

#BF	#PriBF	Pl. Sim.
1	1	0%
1	2	0%
1	3	0%
1	4	0%
1	5	0%
1	6	0%
1	7	0%
1	8	0%
1	9	0%
1	10	0%

#BF	#PriBF	Pl. Sim.
2	1	100%
2	2	1/2
2	3	1/2
2	4	1/2
2	5	1/2
2	6	1/2
2	7	1/2
2	8	1/2
2	9	1/2
2	10	1/2

#BF	#PriBF	Pl. Sim.
3	1	100%
3	2	1/2
3	3	1/3
3	4	1/3
3	5	1/3
3	6	1/3
3	7	1/3
3	8	1/3
3	9	1/3
3	10	1/3

#BF	#PriBF	Pl. Sim.
4	1	
4	2	100%
4	3	1/4
4	4	1/4
4	5	1/4
4	6	1/4
4	7	1/4
4	8	1/4
4	9	1/4
4	10	1/4

#BF	#PriBF	Pl. Sim.
5	1	
5	2	
5	3	
5	4	1/4
5	5	1/5
5	6	1/5
5	7	1/5
5	8	1/5
5	9	1/5
5	10	1/5

#BF	#PriBF	Pl. Sim.
6	1	
6	2	
6	3	
6	4	100%
6	5	1/5
6	6	2/6
6	7	2/6
6	8	2/6
6	9	2/6
6	10	2/6

#BF	#PriBF	Pl. Sim.
7	1	
7	2	
7	3	
7	4	
7	5	100%
7	6	2/6
7	7	2/7
7	8	2/7
7	9	2/7
7	10	2/7

#BF	#PriBF	Pl. Sim.
8	1	
8	2	
8	3	
8	4	
8	5	
8	6	100%
8	7	2/7
8	8	3/8
8	9	3/8
8	10	3/8

#BF	#PriBF	Pl. Sim.
9	1	
9	2	
9	3	
9	4	
9	5	
9	6	100%
9	7	100%
9	8	2/8
9	9	3/9
9	10	3/9

#BF	#PriBF	Pl. Sim.
10	1	
10	2	
10	3	
10	4	
10	5	
10	6	
10	7	100%
10	8	100%
10	9	2/9
10	10	3/10

Fig. 1. Predefined Model of the Plausible Similarity for each combination of cardinality of neighbours in the "ActiveBF" and "PrimaryBF". The Plausible Similarity value can be given as a percentage (%) or as the fraction *#Tolerance / #BF* (see Equation 6). Shadowed combinations indicate a state of uncertainty (UNCERTAIN) (see Equation 7).

Each relevant "Mobility Evaluation State" (see Equation 7) is established for consecutive temporal states of "ActiveBF" and "PrimaryBF" and is associated with the interval of time in which the state was assessed. In order to improve the reliability of the estimation, the number of temporal Bloom filters ("ActiveBF") determined by Equation 4 represent temporal states which are recursively estimated. Thus a state incorporates a cumulative incremental encapsulation of previous states.

A correlated combination of states increases confidence in the estimation of the Mobility Evaluation State. For instance, if the "MOBILITY" state appears correlated in an evaluation time window then mobility is assumed; a change from "STATIC" state to a set of correlated "UNCERTAIN" statesl indicates the lack of definitive information. In the latter case, mobility could be assumed when the difference between #BF and #PriBF exceeds a threshold value. The combination of the number of states, and the order and correlation required, is defined within the model.

When mobility is detected, a neighbourhood discovery process is launched to update the set of core neighbours in the "PrimaryBF" for future estimations. Based on the result, the distance in number of hops to reach all the nodes in the old neighbourhood can be estimated from preestablished values.

IV. Evaluation

The algorithm has been implemented in TinyOS version 2.1.1 [8] and evaluated in its embedded simulator TOSSIM [9], [10] where TOSSIM has been extended to support mobility [11]. The mechanism has been designed to operate as a separate component and, for the evaluation, it is integrated in a gradient-based routing protocol. The "Split Factor" (see Equation 4), which determines the number of "Active" Bloom filters, has been set to 5. The size of the Bloom filter has been set to 64 bits. This helps to reduce the number of false positives through establishing that the maximum number of elements in the filter can be 12, i.e. 12 different neighbours. According to Equation 1, the probability of getting a false positive error in the Bloom filter with a size of 64 bits, for 12 data items inserted, and 4 hash functions, is 0.07915. The number of hash functions chosen is close to the optimal number for minimizing the error, which according to Equation 2 is 3.69.

Two square grid experiments have been formulated to evaluate the mechanism in a controlled situation. A scenario with 16 nodes, where node 0 moves from position (6,0) to (80,74) at different speeds and a topology with 64 nodes, where node 0 moves from position (6,0) to (180,174), see Figure 2.

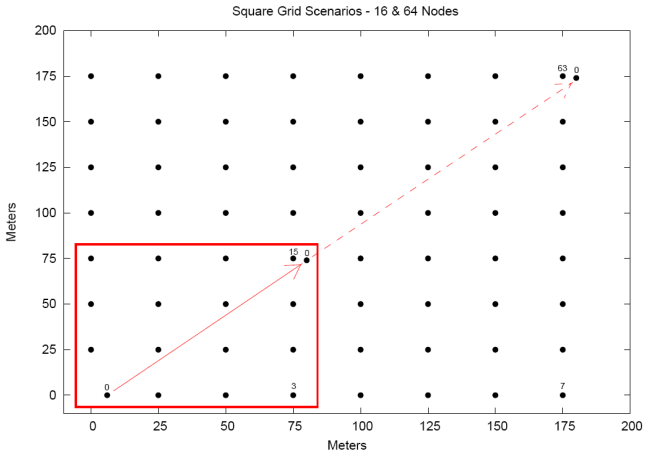

Fig. 2. Topologies with 16 and 64 nodes. Node 0 moves from position (6,0) to (80,74) in the 16 nodes topology and from position (6,0) to (180,174) in the 64 nodes topology.

The results for the experiments conducted are presented in Tables I and II. Node 0 moves at a Speed (Speed) for a time (Sim.T) with a transmitted packets per second rate (Pck/Sec) and a per node Transmission Range (Range(m)). Packets are routed from source to destination on a point to point basis (P2P). Different Evaluation Times (Eval.T(s)) were required depending on the Speed. The delivery ratio (Deliv.Rat) of sent to received Packets has been calculated for each P2P route. The number of times mobility has been detected (Mob.Detec) and the time at which it was detected (Time.Detec) through the node 0 trajectory are used in the assessment of the performance of the algorithm. In addition, the number of false positive detections (False Pos), e.g static nodes which have been detected as mobile, have been presented such that the first parameter is the number of different nodes and the next one is the total number of detections (4/5).

The experimental results in Tables I and II allow the evaluation of the degree of sensitivity of the mobility detection mechanism. The data presented is the result of multiple simulation runs.

A. Interpretation of Results

When calculating the Plausible Similarity, a threshold value of 10% was introduced in UNCERTAINTY cases and a value

TABLE I
RESULTS FOR TOPOLOGY WITH 16 NODES

Speed	P2P	Pck/sec	Range	Sim.T	Eval.T	Deliv.Rat	Mob.Detec	Time.Detec	False Pos
0.8	0→15	1	56	125	5	126/126	1	57s	4/5
0.8	0→15	1	37	125	5	126/126	1	48s	3/3
0.8	0→15	1/2	37	125	5	63/63	1	60s	4/4
0.8	0→15	1/4	37	125	5	32/32	1	47s	4/7
2.8	0→15	1	37	37.5	5	36/36	1	27s	2/2
2.8	0→15	1/2	37	37.5	5	18/18	1	27s	1/1
7	0→15	1	37	15	2	14/14	1	13s	1/1
7	0→15	1/2	37	15	2	7/6	1	13s	1/1
2.8	15→0	1/2	37	37.5	5	19/19	1	23s	3/3
2.8	0→12	1/2	37	37.5	5	18/18	0	Null	0/0
2.8	3→13	1/2	37	37.5	5	18/17	0-1	24-34	0/0
2.8	3→13,1→9	1/2	37	37.5	5	18/17,19/18	0-1	12	0/0

TABLE II
RESULTS FOR TOPOLOGY WITH 64 NODES

Speed	P2P	Pck/sec	Range	Sim.T	Eval.T	Deliv.Rat	Mob.Detec	Time.Detec	False Pos
0.8	0→63	1	56	291	5	290/290	5	61,68,153,164,206s	16/17
0.8	0→63	1	37	291	5	286/286	4-5	56,124,156,203,253s	11/12
0.8	0→63	1/2	37	291	5	148/148	4-5	41,71, 112,144,182,285s	6/6
0.8	0→63	1/4	37	291	5	74/74	4	95,130,214,270s	3/3
2.8	0→63	1	37	87.5	5	86/86	3	24,42,71s	4/4
2.8	0→63	1/2	37	87.5	5	44/44	3	28,50,78s	3/3
7	0→63	1	37	35	5	34/34	2	12,23s	1/1
7	0→63	1/2	37	35	5	17/15	1-2	13,28s	1/1
2.8	63→0	1/2	37	87.5	5	45/45	2	35,66s	1/2
2.8	0→56	1/2	37	87.5	5	44/44	2	56,87s	2/2
2.8	56→0	1/2	37	87.5	5	44/43	1-2	38,83s	1/1
2.8	5→58,3→39	1/2	37	87.5	5	44/44,44/44	1	34s	1/1

of 5% was used in MOBILITY cases (see Equation 6). The parameters provide regulation of the level of tolerance of the mobility mechanism under the differing scenarios. The reliability of the gradient-based routing protocol is evidenced by the high delivery ratio which is close to 100%. The number of packets send is proportional to the simulation time (Sim.T) and the Pck/Sec rate. It can be seen that mobility detection (Mob.Detec) is highly dependent on the amount of network traffic (P2P) in the vicinity of the node. This is expected as the mechanism operates by analysing opportunistic communication. We can see in Table I that mobility is not always detected when the communication occurs in routes at the edges/perimeter which do not send traffic towards route 0→15. The time at which mobility is detected is generally towards the middle-range of the simulation time (Sim.T) (see Table I). This is expected as the transmission range (Range) can extend to roughly half of the network in the 16 nodes scenario. On the other hand, mobility is detected at different points in the larger 64 nodes topology (see Table II). However, the time when the detection occurs also depends on the speed, evaluation time, and the amount of traffic in the vicinity. The mechanism presumes detection of the majority of the core neighbours when populating the "Primary" Bloom filter. This is achieved by launching two staggered neighbour discovery process to reduce the effects of contention and short-term wireless disruptions. Mobility will be detected if the neighbourhood core is not well formed, serving as an indicator of the freshness and completeness of the routing table. In Tables I and II, we can see the number of nodes detecting mobility as a consequence of a bad neighbourhood formation (False Pos). This is verified by the high number of UNCERTAINTY states for these nodes as compared to the set of consecutive MOBILITY states for the moving node 0. The number of false positives is inversely related to the speed which, in turn, is due to the simulation time and the number of evaluation processes (i.e. 1 every Eval.T). For instance, when the speed is 7ms, the evaluation time must be reduced to detect mobility since the simulation time was too low.

V. CONCLUSION AND FUTURE WORK

A mechanism to detect the relative mobility of a node within a neighbourhood has been designed and validated. The algorithm caches eavesdropped routing information in a compressed manner using temporal shift Bloom filters. Numeric filter evaluations are used to identify relative node mobility from within filter sets.

Both 16 and 64 node topologies have been used to evaluate the scheme using a TinyOS implementation. From this we concluded that the algorithm can gauge the relative mobility of a node, subject to eavesdropping neighbourhood traffic. The algorithm can assess the quality of connectivity of the neighbourhood table and is independent of any routing protocol.

Work is ongoing on evaluating the system using larger, more comprehensive and more realistic deployment scenarios in our live testbed. Future activities will see robustness testing and the incorporation of the scheme in other WSN protocols.

VI. ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland.

REFERENCES

[1] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[2] W. Su, S. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *International Journal of Network Management*, vol. 11, no. 1, pp. 3–30, 2001.

[3] N. Hernandez-Cons, S. Kasahara, and Y. Takahashi, "Dynamic Hello/Timeout timer adjustment in routing protocols for reducing overhead in MANETs," *Computer Communications*, 2010.

[4] R. J. Anthony and M. Ghassemian, "Mobility status as dynamic context for behaviour optimisation in self-organised networks," *Complex, Intelligent and Software Intensive Systems, International Conference*, vol. 0, pp. 878–885, 2009.

[5] T. Tran, B. Scheuermann, and M. Mauve, "Lightweight detection of node presence in MANETs," *Ad Hoc Networks*, vol. 7, no. 7, pp. 1386–1399, 2009.

[6] H. Xu and J. Garcia-Luna-Aceves, "Neighborhood tracking for mobile ad hoc networks," *Computer Networks*, vol. 53, no. 10, pp. 1683–1696, 2009.

[7] S. Cohen and Y. Matias, "Spectral bloom filters," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003, p. 252.

[8] UC Berkeley, "TinyOS version 2.x," July 2010. [Online]. Available: http://www.tinyos.net/tinyos-2.x/

[9] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international Conference on Embedded Networked Sensor Systems*, 2003.

[10] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, 2007.

[11] C. Stevens, C. Lyons, R. Hendrych, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "Simulating mobility in wsns: Bridging the gap between ns-2 and tossim 2.x," in *DS-RT '09: Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2009.