Scalable Computing: Practice and Experience
Volume 10, Number 1, pp. 25–34. http://www.scpe.org
ISSN 1895-1767
© 2009 SCPE

IMPACT OF THE DYNAMIC MEMBERSHIP IN THE CONNECTIVITY GRAPH OF THE WIRELESS AD HOC NETWORKS

ARTA DOCI,[*] WILLIAM SPRINGER,[†] AND FATOS XHAFA[‡]

Abstract. Mobility Metrics of the connectivity graphs with fixed number of nodes have been presented in several research studies over the last three years. More realistic mobility models that are derived from real user traces challenge the assumption of the connectivity graph with fixed number of nodes, but they rather show that wireless nodes posses dynamic membership (nodes join and leave the simulation dynamically based on some random variable). In this paper, we evaluate the mobility metrics of the dynamic connectivity graph on both the nodes and the links. The contributions of this paper are two-fold. First, we introduce the algorithm that computes the Maximum Node Degree. We show that the Maximum Node Degree, which characterizes the dynamic membership property, impacts the connectivity mobility metrics of the Link and Path Durations of the dynamic connectivity graphs. Second, we provide the lower and the upper bounds for these mobility metrics.

Key words: wireless ad hoc networks, mobility metrics, connectivity, simulation.

1. Introduction.

Many studies have demonstrated that mobility has a significant impact on the performance of ad hoc network protocols. Specifically, the authors of [5] provide a framework, which is helpful in understanding *how* and *what* mobility metrics affect the protocol performance. For example, the mobility metrics that effect the performance are average shortest path hop count ($A_{sp}Hops$) [11], average link durations [2] and average path durations [13]. In addition, other path and link metrics have been proposed by the authors of [7, 9, 15] and have been shown to effect performance. All these mobility metrics are derived from synthetic mobility models, which face two main issues.

First, the connectivity graph of the synthetic mobility models, where the mobile nodes are the vertices and the communication links are the edges, have been assumed to *have static number of mobile nodes for the simulation duration.* The number of vertices $|V|$ represent the order and the number of edges $|E|$ represent the size of the connectivity graph. In general, the running time of the algorithms are measured in terms of the order and size of the graph, which is relatively easy to estimate when the number of nodes does not change, but becomes a cumbersome task when the graph is dynamic on both the vertices and edges.

Second, the current implementations of the synthetic mobility models place the wireless nodes to start the simulation at time 0 and remain in the simulation until the allotted simulation time is over. On the other hand, real mobility models [16], which are extracted from real user traces, show that wireless nodes posses dynamic membership, that is, they join and leave the simulation based on an exponentially distributed random variable. Under dynamic membership of the nodes, the connectivity graph is *dynamic on the number of vertices and edges.* When adding the new dimension of the dynamic membership to simulation mobility models, then the mobility metrics need to be re evaluated under the new dimension. The focus of this paper is to assess the mobility metrics of the dynamic connectivity graph.

Maximum Node Degree mobility metric, which represents the maximum number of neighbors for each node (in graph theory terms it is the number of edges incident to it), can be used to account for the dynamic membership of the wireless nodes. The contributions of this paper are:

1. Design and implement algorithms that compute the Maximum Node Degree, Average Links, and Average Path Durations mobility metrics for dynamic topology connectivity graphs.
2. Present the lower and upper bounds for the Maximum Node Degree, Link Durations, and Path Durations mobility metrics.

2. RealMobGen: A More Realistic Mobility Model.

The main characteristics of a mobility model are speed, pause distribution, and direction of movement. For example, the most used synthetic mobility model is Random Walk Model (RWM) [3]. In RWM each node is assigned a randomly distributed initial location $(x0; y0)$. Then, each node randomly picks up a destination independent of their initial positions and moves toward it with speed chosen uniformly on the interval $(v0; v1)$. Nodes pause upon reaching each destination. The process is repeated until the allotted simulation time is reached. There are several issues with RWM, which are addressed in RealMobGen.

[*]Colorado School of Mines, Colorado, USA (adoci@mines.edu).
[†]Colorado State University, Colorado, USA (wmspring@CS.ColoState.EDU).
[‡]Technical University of Catalonia, Barcelona, Spain (fatos@lsi.upc.edu).

RealMobGen is a hybrid model that is based on Dartmouth's model of mobile network traces [10] and USC's WWP [6] survey collected from the students on campus. The model closely mimics the environments where ad hoc networks will likely be deployed, since it borrows its characteristics from models derived from real user traces. Another feature of RealMobGen, that is not existent in any other current mobility model, is the classification of nodes as stationary (46% of the nodes) and mobile (54% of the nodes). The ratio of stationary vs. mobile nodes was borrowed from the Dartmouth model.

The stationary nodes select a location based on a transition matrix that defines the probabilities for moving from one point to another. Once a location is selected, a node is turned on for a time drawn from the exponential distribution of start time for the stationary nodes. Stationary node stays at the selected location until the allotted stationary end time. The mobile nodes, also, select a start location based on the transition matrix. The mobile node enters the simulation at a time drawn from the mobile node start time. The node pauses at the selected location until the allotted pause time from mobile pause time exponential distribution. After the pause time is up, the mobile node selects the next location based on the transition matrix and moves there not in a straight line but following data that supports movements along popular roads and turns. The mobile node repeats the pattern 'pause-select next location - move there' until allotted mobile simulation end time.

RealMobGen shows that wireless nodes tend to cluster around popular locations, i. e., cafeteria, gym, classes, and library. We believe that RealMobGen is the first mobility model (we are not aware of any other models) that implements the dynamic characteristic of wireless devices in NS 2, devices join and leave the network at different times. RealMobGen addresses the drawbacks present in the RWM by implementing the following new features:

Feature 1: Wireless Nodes are clustered around popular hotspots. For example, Figure 2.1 shows a snapshot of RealMobGen with 100 nodes, which are clustered around 14 hotspots.

Feature 2: Wireless Nodes posses dynamic membership. For example, Figure 2.2 shows the dynamic membership of 60 nodes.

Feature 3: Nodes are classified on two flavors, namely stationary and mobile (stationary (46%)of the nodes and mobile (54% of the nodes). The ratio of stationary vs. mobile nodes was borrowed from the Dartmouth model.

Feature 4: Moving from one point to another is done via waypoints, instead of a straight line.

3. Connectivity graph and Mobility Metrics.

In more realistic mobility models the connectivity graph $G = (V, E)$ is a dynamic graph on vertices that join and leave and edges that appear and disappear due to mobility. In graph theory terms, when taking into account the dynamic membership of the nodes, there is an edge between any two nodes (i, j), if the following two conditions are met:

1. $D_{i,j} \leq R$, where D is the euclidian distance between nodes (i, j) and R is the transmission range of the wireless nodes.
2. $I_i \cap I_j \neq \emptyset$, where I represents the active intervals of the nodes.

Thus, when interested on the mobility metrics, i. e., path and link durations, we are basically posing the shortest path problem in the case of the dynamic topology and edges. Basically, this can be represented as a graph with edges that are added and removed, as the vertices turn 'on' and 'off'. While the method of calculating the full graph is polynomial in the size of the input, it is quite inefficient as each movement of a node can potentially cause $n-1$ edge insertions and the same number of deletions. We can obtain better bounds by restricting the algorithm to only *processing the nodes, which are currently relevant.*

In order to consider only the nodes that are relevant, first we process the dynamic membership information. For example, consider two nodes i and j. Since all communication is instantaneous, any communication between the nodes must occur while both nodes are turned on. If the communication is indirect, then every node that participates in routing the messages must also be turned on during that period of time. While, we should note that, this can be $O(n)$ nodes in the worst case, in most cases this allows us to consider only a fraction of the vertices of the full graph (See Section *Maximum Node Degree Mobility Metric*).

3.1. Dynamic Membership Graph is an Interval Graph.

Given a set of n intervals (which represent the number of wireless nodes) $I = I_1, I_2, \dots, I_n$, the corresponding interval graph $G = (V, E)$ has the set of vertexes $V = v_1, v_2, \dots, v_n$ and there is an edge E linking nodes (v_i, v_j) if and only if $I_i \cap I_j \neq \emptyset$ (See Figure 3.1). We will use the interval graphs to calculate the $MaximumNodeDegree$.

4. Maximum Node Degree Mobility Metric.

In order to define the upper bounds of the number of nodes that are neighbors, we formulate the problem as a coloring problem, where the goal of the algorithm is

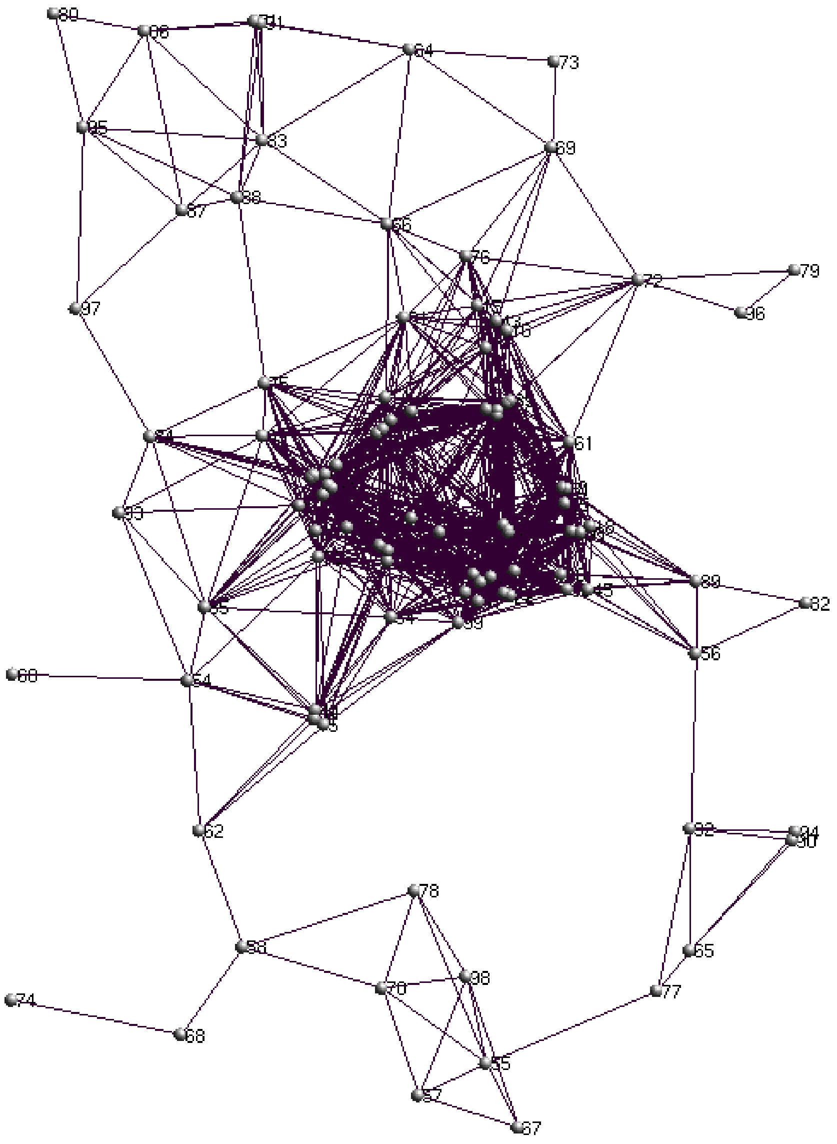
FIG. 2.1. *100 Nodes cluster around 14 Hotspots.*

to color each interval with the minimal number of colors in such a way that no two overlapping intervals are colored with the same color. The minimal number of colors represents the upper bound of the Maximum Node Degree[1] In Figure 4.1, we graph the dynamic membership information, as an interval graph, which is parsed from the output of RealMobGen with 40 wireless nodes.

In this section we present the algorithm for the Maximum Node Degree. The algorithm reads each interval and sorts them by the beginning interval times (See Figure 4.1).

The *MaximumNodeDegree* is derived by the *GreedlyMaximumNodeDegree* Algorithm (See Algorithm 2). The input to the algorithm is the dynamic membership of the nodes, which is a series of intervals that are determined by their beginning and ending time of active nodes. Our goal is to find all the overlapping active time intervals. The number of overlapping intervals gives us the upper bound of the number of neighbors for each node. We set the problem as a coloring problem. The goal is to color each interval with the minimal number of colors in such a way that no two overlapping intervals are colored with the same color.

[1]The algorithm in this section is similar to independent work in the context of channel routing problem [14].

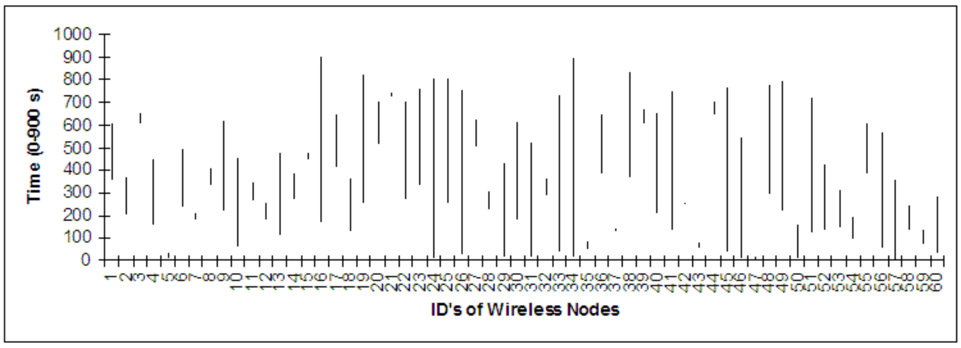

Fig. 2.2. *Dynamic Membership of 60 Nodes.*

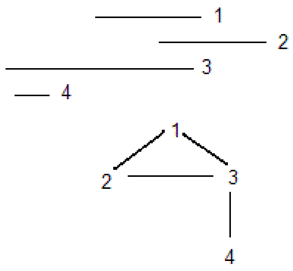

Fig. 3.1. *A set of four intervals and the corresponding interval graph.*

First, the algorithm reads all the intervals and sorts them by their beginning times. After it assigns to the first interval the Color 1, the algorithm looks at the second interval, in order to figure out whether it overlaps with any other intervals. If it overlaps, then it calculates what colors have already been used and assigns the next available color to the interval. On the other hand, if the answer is "no", then the algorithm assigns the color of the overlapping interval.

Algorithm 2 : *GreedyMaximumNodeDegree.*

```
Input:  Active Intervals("TimeOn", "TimeOff"); ($N$) Intervals
 1: sortedIntervals ← sortByTimeOn (Active Intervals).
 2: Greedily IntervalColoring (sortedIntervals).
 3: $loopCounter \leftarrow 1$
 4: $colorCounter \leftarrow 1$
 5: ColorIntervals(sortedIntervals[loopCounter]) ← $Colour[colorCounter]$.
 6: repeat
 7:   loopCounter ← loopCounter + 1
 8:   nextInterval ← sortedIntervals [loopCounter].
 9:   if nextInterval OVERLAPS previousIntervals then
10:     colorCounter ← colorCounter + 1
11:     ColorIntervals(sortedIntervals[loopCounter]) ← (Colour[colorCounter])
12:   else
13:     ColorIntervals(sortedIntervals [colorCounterOfOverlapping]) ← (Colour[colorCounterOfOverlapping])
14:   end if
15: until $loopCounter \leq N$
Output:  Active Intervals: ($''TimeOn'',''TimeOff''$), Interval Color
```

We illustrate the algorithm by running it in the same example of the 40 nodes used in the previous sections. For example, in Table 4.1 we show the partial output of the algorithm. The table illustrates that the first seven intervals overlap, thus the color of the intervals is incremental. On the other hand, the eighth interval

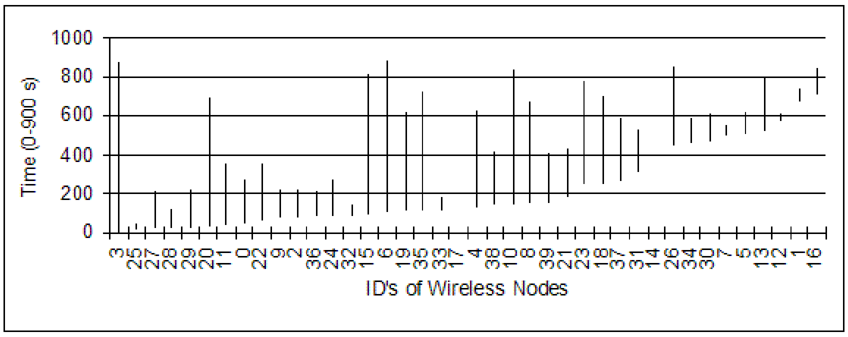

Fig. 4.1. *Dynamic Membership Graph on 40 nodes: Sorted by beginning interval times.*

Table 4.1
The partial output of the Algorithm GreedyMaximumNodeDegree.

Time node turns "ON"	Time node turns "OFF"	Assigned Interval Color
1.28239834	874.2507994	1
9.112384965	35.76696803	2
12.33443828	206.6781021	3
13.96709987	116.2303641	4
15.42727166	212.4065595	5
22.61315476	686.9883725	6
28.87034892	349.2144535	7
39.14919133	269.6973103	2

defined by $(39.14919133, 269.6973103)$ does not overlap with the second one $(9.112384965, 35.76696803)$, thus gets assigned the same color (Color 2).

The algorithm *GreedyMaximumNodeDegree* falls in the greedy class of the algorithms, since it takes the best immediate, or local, solution while finding a global one. The global solution represents the upper bound of the node neighbors degree, while the average of the local optimum(s) and the global one provides us with the average node neighbors degree. It can be shown that the algorithm is optimal (following a well known result from greedy scheduling algorithms [14]).

So far, we have provided the algorithm for the upper bounds of the Maximum Node Degree mobility metric. Now, we will derive the lower bounds for the metric. The input to the algorithm for the lower bounds is the ***Active Intervals***: $("TimeOn","TimeOff")$ and the *Interval Color*. For each color from the Interval Color, we compute the pairwise Euclidian distance of the nodes that belong to that color. If the distance is greater than the transmission range, then the nodes are not neighbors. Else, the nodes are neighbors, thus we increment the *MaximumNodeDegree* counter for this pair of nodes. The algorithm is summarized below:

```
INPUT Active Intervals: $("TimeOn","TimeOff")$, Interval Color
For each Color from the Interval Color do
        • Compute the pairwise Euclidian distance of the nodes that belong to that color.
        • If $D_{i,j} \leq R$, increment the $MaximumNodeDegree$ counter for this pair of nodes.
        • Else, process to the next pair of nodes.
```

We should mention that in NS 2, in order to maintain the lower bounds over the entire simulation time the snapshot of the network should be taken every $0.5s$ and compute the metric for each snapshot. The average of the metric should be computed in the and over the entire simulation time and used as the lower bound of the average maximum node degree.

5. Link and Path Durations Mobility Metrics. We make use of dynamic programming to compute the mobility metrics of link and path durations. In order to compute these metrics, we make optimal solutions sequentially during the simulation, thus ensuring a global solution in the continuous time. In this section again we use the fact that ad hoc network simulations evolve with time; thus, to compute the upper and lower bounds of link and path durations over the entire simulation time we take a snapshot of the network every $0.5s$ and compute the metrics for each snapshot. Then, we average the results of all the individual snapshots and present the average Maximum Node Degree over the entire simulation. For example, if the simulation time is $900s$, then the first step is to divide the simulation time in small time intervals (for illustration purposes we selected to divide it into $0.5s$ time intervals) and take the snapshot of the network for each time interval, which results into $\frac{900s}{0.5s} = 1800 snapshots$.

First, we present the algorithm that provides the lower bounds for the path duration metric. So far, the research on this area considers the graph static on the number of nodes. Therefore, the run time of the shortest path(s) on N nodes for each snapshot is $O(V^3)$ (over the simulation for the $900s$ simulation time it will be $1800 * O(V^3)$ (where $|V| = N$).

In this section we improve the algorithm complexity to $O((V^*)^2 logV^* + V^*E^*)$ for each snapshot (V^* represents the active nodes only and E^* represents the active edges for each snapshot, thus both are dynamic). The algorithm takes as input the dynamic membership of the nodes, which is represented by Active Intervals(“TimeOn”, “TimeOff”). On each snapshot we include only the nodes that are active at that time, thus allowing us to perform the algorithm only for those nodes adjacent to the previously selected nodes [8]. In this paper we implemented Johnson’s sequential single-source shortest paths algorithm (As shown in Algorithm 3).

Algorithm 3 : *JohnsonSingleSourceShortestPaths.*

```
Input:  Active Vertices;Active Edges;s
 1: PriorityQueue=ActiveVertices
 2: for all $v \epsilon ActiveVertices$ do
 3:    $l[v] \Leftarrow \infty$
 4: end for
 5: $l[s] \Leftarrow 0$
 6: while $PriorityQueue \neq 0$ do
 7:    $u \Leftarrow findminimum(PriorityQueue)$
 8:    for each $v \epsilon Adjacent[u]$ do
 9:       if $(v \epsilon PriorityQueue)$ and $(l[u] + w(u,v) < l[v])$ then
10:          $l[v] \Leftarrow l[u] + w(u,v)$
11:       end if
12:    end for
13: end while
Output:  ShortestPathsMatrix
```

Let’s illustrate the output of the algorithm through an example. Let’s assume that we are given the undirected graph and its weights (See Figure 5.1).

We run the algorithm *JohnsonSingleSourceShortestPaths*, which outputs the matrix

$$ShortestPathsMatrix = \begin{pmatrix} 0 & 53 & 53 & 45 & 21 & 82 \\ 53 & 0 & 51 & 66 & 32 & 40 \\ 53 & 51 & 0 & 15 & 32 & 29 \\ 45 & 66 & 15 & 0 & 36 & 38 \\ 21 & 32 & 32 & 36 & 0 & 61 \\ 82 & 40 & 29 & 38 & 61 & 0 \end{pmatrix}.$$

In the implementation phase we designed matrix multiplication as addition, while the addition as a minimization operators. In addition, the product of adjacency matric with itself returns shortest pair of length 2 for any pair of nodes. Thus, to get the n shortest paths we compute by doubling powers of the *Adjacency*, $Adjacency^2$, $Adjacency^4$,... $Adjacency^n$. Furthermore, we have added the parallel implementation of Johnson’s Algorithm, which extracts simultaneously p nodes from the priority queue. Each of the nodes will update the neighbors costs, as well as, augment to the shortest path. The improvement with the parallel algorithm is $\frac{|V^*|}{p} * log\frac{|V^*|}{p}$ on p processors for each snapshot.

The Link Durations is computed by the algorithm *LinkDurations* (See Algorithm 4).

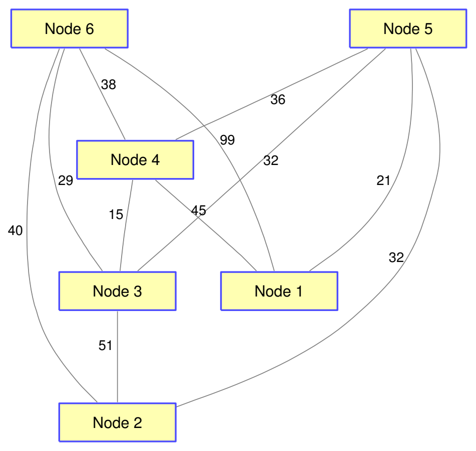

Fig. 5.1. *Illustrate all-pair shortest path algorithm.*

Algorithm 4 : *LinkDurations.*

```
Input: Active Vertices;Active Edges;
1: for all $v \epsilon ActiveVertices$ do
2:    for each $v \epsilon Adjacent[u]$ do
3:       if $(v overlaps u)$ and $(D(u,v) < R))$ then
4:          $link[u,v] \Leftarrow OverlapTime[u,v]$
5:       end if
6:    end for
7: end for
Output: LinkDurationsMatrix
```

So far in this section, we have provided the lower bounds for the Link and Path durations mobility metric. In order to get the upper bounds of the maximum node degree the algorithm takes as input the $GreedyMaximumNodeDegree$ algorithm's output. For each overlapping interval we compute the amount of the time they overlap. The overlapping time represents the link durations between the two wireless nodes. We repeat the computation until there are no more overlapping intervals. For the path durations we compute all the possible paths of the overlapping intervals and add the corresponding link durations. Lastly, we average all the individual snapshot results and present the average Link and Path Durations over the entire simulation.

6. Case Study: Efficiency using Maximum Node Degree.

In order to demonstrate how to use the Maximum Node Degree mobility metric to improve algorithm efficiency we illustrate it by a case study that is the incentive protocols in ad hoc networks.

6.1. Incentive Ad Hoc Protocols.

Ad hoc networks are self-organizing and multi-hop networks with no central authority. Thus, every aspect of the configuration and operation of an ad-hoc network is distributed. Another characteristic is that nodes are power and energy constrained. Thus, each node running a distributed protocol must make its own decisions (possibly relying on information from other nodes). Those decisions maybe constrained by the rules or algorithms of a protocol, but ultimately each node would have some leeway in setting parameters or changing the mode of operation. These nodes are autonomous agents, making decisions about transmit power, packet forwarding, back off time, and so on.

In the presence of the selfish nodes, the goal of each wireless device is to maximize its welfare: $WELFARE = Profit - Costs$, where profit is the payments received for forwarding traffic and the costs are the incurred energy loss of the node by transmitting packets for other nodes and sending its own traffic; and the payments to others that forwarded its own traffic. On the other hand the goal of the incentive protocol designer is to provide

Table 6.1
RealMobGen Simulation Parameters.

Parameter	Value
Simulation Duration	900 s
Simulation Area	900×1200 m
Number of Hotspots	14
Number of Nodes (nn)	$nn \in (40, 60, 90)$

incentives to the wireless nodes to relay traffic, such that nodes will have no incentives to deviate from the protocol, since doing so will not bring them a higher welfare.

In the presence of selfish nodes most of the proposed protocols are based on the well known Vickrey-Clarke-Groves (VCG) mechanism. Below we describe one such protocol based on VCG. Vickery auction is most familiar because it is the foundation of eBay's auction design. In the Vickrey auction the high bidder wins, but pays the second-highest bid. This is why the Vickrey auction is called a second-price auction: the price is not the highest bid, but the second-highest bid. Desirable properties of auction protocols are given in [12], which include Strategy-Proofness, Pareto Efficiency, Individual Rationality, and Budget-Balance.

In [1, 4, 17] the authors propose incentive protocols that ensure that the participating wireless nodes will have no reason to deviate from the protocol, since doing so will not bring them a higher welfare. For example, in [1] the authors propose the Ad-Hoc VCG protocol, which is a reactive routing protocol that achieves the design objectives of truthfulness. Reactive protocols seek to set up routes on-demand, thus topology information is only transmitted by nodes on-demand. In this protocol vertexes represent the nodes and weighted directed edges represent the payments an emitting node has to receive. Nodes are awarded payments for forwarding a message, thus cover the cost for forwarding a unit-size packet: $Payment = c_i * P^{emit}$, where c_i is the cost-of energy of $\$/Watt$ and P^{emit} is emission signal strength in $watt$.

$$Payment = DeclaredCost + (LCP_{withthenode} - LCP_{withoutthenode}). \tag{6.1}$$

The protocol can be thought as it is run on two phases. Firstly, route discovery, where nodes communicate to destination P^{emit} and c_i. Then, the destination computes Lowest Cost Path (LCP). Secondly, data transmission, packets are forwarded along the shortest path route and payments are made to the intermediate nodes.

The algorithm complexity for the proposed protocols [1, 17] is $O(N^3)$, for each time snap of the network during simulation time. In [4] the authors try to reduce the complexity to $O(M^2 * d)$, where d is the diameter and M is some upper bound for the node degree. However, the value of M was not computed. We computed M by using the $GreedyMaximumNodeDegree$ Algorithm. Next, we present the efficiency introduced by taking into account the value of M, which is the maximum node degree.

6.2. Efficiency. We propose the metric *Efficiency*, as a metric to evaluate the gains in the algorithm complexity of incentive protocols.

In this section, we run the *GreedlyDynamicMembership* with several scenarios. The scenarios are generated using RealMobGen mobility model. The inputs to the RealMobGen are the simulation duration set to 900 seconds; simulation area of 900×1200 meters; number of hotspots set to 14, while the number of nodes was varied $40, 60, 100$ (the parameters are summarized in Table 6.1).

For each set of nodes, we run the simulation 10 times and present the results in Table 6.2. We define a new metric, namely *Efficiency*, that calculates the improvement of real mobility metric algorithms over the synthetic ones, i. e., in terms of relevant comparisons (See Eq.(6.2)).

$$Efficiency = \left(\frac{\frac{\sum_T MaxNodeD_{egree}}{T}}{N} \right) * 100. \tag{6.2}$$

For example, from the table we have the $Efficiency$ for $40, 60, 100$ number of nodes to be respectively $54.75, 54.33, 52.30$, which demonstrates that we need to compare only half of the nodes, instead of the full graph when computing the mobility metrics (N^2).

TABLE 6.2
Number of necessary comparisons on 40, 60, 100 nodes.

$MaxNodeD_{egree}$ on 40 Nodes	$MaxNodeD_{egree}$ on 60 Nodes	$MaxNodeD_{egree}$ on 100 Nodes
24	33	50
21	27	55
16	28	49
25	35	55
21	32	53
25	34	55
20	37	48
23	32	51
21	29	54
23	39	53
Efficiency = 54.75%	*Efficiency* = 54.33%	*Efficiency* = 52.30%

7. Conclusions. In this paper we demonstrated that when introducing dynamic topology in wireless *adhoc* networks, the current mobility metrics need to be re evaluated. We introduced Maximum Node Degree as a mobility metrics, which can be used to improve algorithm efficiency in wireless *adhoc* networks. In addition, we re-evaluate the link and path durations mobility metrics and provided their lower and upper bounds. The usefulness of the Maximum Node Degree mobility metric for improving the efficiency is illustrated by a case study of incentive protocols in ad hoc networks. We have defined a new metric, namely *Efficiency*, to characterize the improvement of real mobility metric algorithms over the synthetic ones. The results are validated trough simulations.

REFERENCES

[1] L. Anderegg and S. Eidenbenz, *Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents*, in MobiCom, Septmber 2003.

[2] J. Boleng, W. Navidi, and T. Camp, *Metrics to enable adaptive protocols for mobile ad hoc networks*, in International Conference onWireless Networks, 2002, pp. 293-298.

[3] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, *Multihop wireless ad hoc network routing protocols*, in MobiCom 1998: Proceedings of the Fourth Annual ACM International Conference on Mobile Computing and Networking, 1998, p. 8597.

[4] S. Eidenbenz, G. Resta, and P. Santi, *Commit: A sender-centric truthful and energy-efficient routing protocol for ad hoc networks with selfish nodes*, in IPDPS, April 2005.

[5] F.Bai, N. Sadagopan, and A. Helmy, *Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks*, in INFOCOM, 2003, pp. 825–835.

[6] W. J. Hsu, K. Merchant, H. W. Shu, C. H. Hsu, and A. Helmy, *Weighted waypoint mobility model and its impact on ad hoc networks*, SIGMOBILE Mobile Computer Communications Review, 9 (2005), pp. 59–63.

[7] S. Jiang, *An enhanced prediction-based link availability estimation for manets*, IEEE Transactions on Communications, 52 (2004), pp. 183–186.

[8] D. B. Johnson, *Efficient algorithms for shortest paths in sparse networks*, J. ACM, 24 (1977), pp. 1–13.

[9] H. M. Jones, S. Xu, and K. Blackmore, *Link ratio for ad hoc networks in a rayleigh fading channel*, in WITSP, 2004, pp. 252–255.

[10] M. Kim, D. Kotz, and S. Kim, *Extracting a mobility model from real user traces*, in INFORCOM: Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies, 2006, pp. 1–13.

[11] S. Kurkowski, W. Navidi, and T. Camp, *Constructing manet simulation scenarios that meet standards*, in MASS, 2007, p. To Appear.

[12] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic theory*, Oxford University Press, (1995).

[13] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy, *Paths: analysis of path duration statistics and their impact on reactive manet routing protocols*, in MobiHoc, 2003, pp. 245–256.

[14] M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design*, McGraw-Hill Higher Education, 1996.

[15] W. Su, S. Lee, and M. Gerla, *Mobility prediction and routing in ad hoc wireless networks*, International Journal of Network Management, 11 (2001), pp. 3–30.

[16] C. Walsh, A. Doci, and T. Camp, *A call to arms: its time for real mobility models*, vol. 12, no. 1, pp. 3436, 2008.

[17] S. Zhong, L. E. Li, Y. G. Liu, and Y. R. Yang, *On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretical and cryptographic techniques*, in MobiCom, 2005, pp. 117–131.

Edited by: Fatos Xhafa, Leonard Barolli
Received: September 30, 2008
Accepted: December 15, 2008