On the Set Cover problem for Broadcasting in Wireless Ad Hoc Networks

Spiros Agathos, *Student Member, IEEE* and Evangelos Papapetrou *Member, IEEE*
Department of Computer Science & Engineering, University of Ioannina, e-mail: sagathos,epap@cs.uoi.gr

***Abstract*—The set cover problem has been extensively used in broadcast algorithms to model the process of selecting the nodes that should forward a message. Surprisingly, none of the research efforts focuses on solving the set cover problem. Instead, virtually every broadcast scheme uses the well known and generic purpose Greedy Set Cover (GSC) algorithm to tackle the problem. We focus on solving the set cover problem in the context of wireless ad hoc networks and show that providing an efficient solution has a significant performance impact on broadcast algorithms.**

I. Introduction

A plethora of mechanisms in wireless ad hoc networks, ranging from routing to resource discovery, utilize broadcasting for collecting and/or disseminating network information. An implementation of broadcasting that minimizes the number of transmissions is of paramount importance due to the sparsity of resources such as bandwidth, power, processing, etc [1]. The proposed algorithms follow two major approaches [2]. In the first, each node uses a heuristic to decide when to refrain from forwarding a message. A more elaborate approach is to use a connected dominating set (CDS) for determining the forwarders of a message. In most CDS-based algorithms, each node decides which neighbors should forward a message by modeling the selection process as a *set cover problem* [3]–[7], which is known to be NP-hard [8]. However providing a good approximate solution is essential for reducing the number of transmissions. Surprisingly, almost every broadcast scheme relies on the generic purpose Greedy Set Cover (GSC) algorithm [8] to solve the problem, obviously due to the well-established view that there is not enough room for improving GSC. The only exception is MPR [5], in which GSC is enhanced with a heuristic that is known to improve performance [9].

In this letter, we make the observation that providing a good approximate solution to the set cover problem, in the context of CDS-based broadcast algorithms for wireless ad hoc networks, may result in significant performance gains. More specifically, we propose a greedy algorithm that extends GSC as well as the MPR heuristic by preprocessing the list of candidate forwarders in order to narrow it down. The intuition is that such a strategy will increase the efficiency of the choices made by both algorithms. The proposed modification can be ported to every CDS-based algorithm that uses GSC with or without the MPR heuristic. We show that it can provide a significant reduction of transmissions with a low computational cost.

In the following, in Section II, we provide some background knowledge while in Section III, we discuss our motivation and delineate the proposed algorithm. Its evaluation is presented in Section IV and conclusions are drawn in Section V.

II. Preliminaries

We first provide some background on broadcast algorithms that utilize the set cover model. In the following, the network is modeled as an undirected graph $G(V, E)$, where V is the set of network nodes and every edge $(v,u) \in E$ indicates that node u lies in v's transmission range and vice versa. Furthermore, $N(v)$ denotes the neighborhood of v, i.e. the set of nodes that lie within v's range while $N(N(v))$ is the 2-hop neighborhood, i.e. the set of nodes that lie within 2 hops from v.

A. The Set Cover Problem in Broadcasting

As discussed previously, in a plethora of broadcast algorithms for wireless ad hoc networks, each node v calculates a CDS for its 2-hop neighborhood by modeling the problem as a set cover problem [8]. To understand this approach, note that the CDS consists only of v's direct neighbors since all strictly 2-hop neighbors, i.e. nodes in $N(N(v))–N(v)$, are neighbors of at least one node in $N(v)$. Let:

- $B(v)$ denote the set of nodes that are candidates for the CDS, thus $B(v)=N(v)$
- $U(v)$ denote the set of strictly 2-hop neighbors, i.e. $U(v) = N(N(v))–N(v)$

Then, v should select as a CDS the minimum subset $F(v) \subseteq B(v)$ such that every node in $U(v)$ has at least one neighbor in $F(v)$. In other words, if every node in $F(v)$ forwards a message from v then all nodes in $U(v)$ will receive that message. Therefore, $F(v)$ is called the *forwarding set of* v. The problem of calculating $F(v)$ can easily be modeled as a set cover problem if $B(v)$ is seen as a set of sets by replacing each node $w \in B(v)$ with the set:

$$C(w) = N(w) \cap U(v) \tag{1}$$

which is the part of $N(w)$ that lies in $U(v)$ (see fig. 1 where dashed lines indicate $C(w), \forall w \in B(v)$). We say that:

Definition 1: A node $w \in B(v)$ covers a node $z \in U(v)$ iff $z \in C(w)$.

Definition 2: Node w is said to cover k nodes in $U(v)$ if $|C(w)|=k$.

It is clear that $U(v) = \bigcup_{\forall w \in B(v)} C(w)$. The solution to the set cover problem is to define the minimum set $F(v) \subseteq B(v)$ such that:

$$U(v) = \bigcup_{\forall f \in F(v)} C(f) \tag{2}$$

Finally, note that although the simplest approach in defining $B(v)$ and $U(v)$ is to set $B(v) = N(v)$ and $U(v) = N(N(v))-N(v)$, there exist algorithms [3], [4], [6], [7] that utilize knowledge about the nodes that have already received a message in order to reduce $B(v)$ and $U(v)$.

B. Solving the Set Cover Problem

Minimizing $|F(v)|$ is of paramount importance for minimizing the number of transmissions in the network. However, producing the minimum set cover, i.e. the minimum forwarding set, is known to be NP-hard [8]. In the context of broadcasting, there exist two approaches for providing solutions

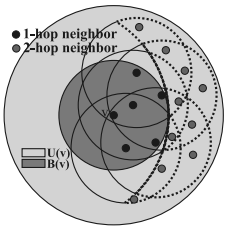


Fig. 1. The set cover problem for a typical definition of $B(v)$ and $U(v)$.

that approximate the minimum sized forwarding set. The first utilizes the well known *greedy set cover (GSC)* algorithm [8]. GSC uses a simple heuristic for electing the forwarding nodes. More specifically, GSC elects from $B(v)$ the node that covers more nodes in $U(v)$, i.e. it elects the node f for which $|C(f)|$ is maximum. When a node f is elected, the covered nodes, i.e., the nodes in $C(f)$, are removed from $U(v)$. The process repeats until no node is left in $U(v)$. The second approach utilizes the algorithm proposed in the context of MPR [5], hereafter called MPR for simplicity. The MPR algorithm extends GSC by introducing another heuristic in which the algorithm first elects as forwarders the nodes that are unique in covering a node in $U(v)$, i.e. a node $f \in B(v)$ is elected if $\exists z \in U(v) : [z \in C(f)] \wedge [z \notin C(w), \forall w \neq f \in B(w)]$.

III. Eliminating Fully Covered Nodes

It is clear that both heuristics that have been proposed so far by GSC and MPR focus on identifying the best criterion for selecting forwarders from the set of candidates $B(v)$. The motivation for this work has been the observation that the performance of the aforementioned heuristics can be improved if we narrow down the candidates. This can be done by identifying and ruling out the nodes that should not be elected as forwarders. To this end, we propose an algorithm that extends the operation of MPR (recall that MPR implements both aforementioned heuristics), by introducing a preprocessing of $B(v)$. We call the proposed algorithm *greedy set cover-eliminate fully covered nodes (GSC-EFCN)*. The observation made by GSC-EFCN (hereafter called EFCN) is that a node $f_1 \in B(v)$ should not be elected as a forwarder if there exists another node $f_2 \in B(v)$ such that $C(f_1) \subseteq C(f_2)$ (recall that $C(i)=N(i) \cap U(v)$ is the set of nodes in $U(v)$ which node i covers). The rationale is that f_2 is a more suitable candidate since it can forward the message to more nodes lying in $U(v)$, i.e. $|C(f_2| \geq |C(f_1)|$. We say that:

Definition 3: A node $f_2 \in B(v)$ *fully covers* node $f_1 \in B(v)$ iff $C(f_1) \subseteq C(f_2)$.

Correspondingly, we say that a node f_1 is *fully covered* if there exists at least one node f_2 that fully covers f_1.

The pseudocode of EFCN is illustrated in fig. 2. In the first phase (lines 3-7) all fully covered nodes are eliminated, i.e.:
First phase: Every node $f_1 \in B(v)$ for which:

$$\exists f_2 \in B(v) : C(f_1) \subseteq C(f_2) \quad (3)$$

is identified and eliminated from $B(v)$. In the case that $\exists\, f_1, f_2 \in B(v) : C(f_1) = C(f_2)$ (nodes f_1 and f_2 are mutually fully covered), then only one node is maintained in $B(v)$. The node id may be used to break the tie. After the elimination of fully covered nodes, EFCN continues with the second phase where it focuses on candidate forwarders that are unique in covering a node in $U(v)$.

Second phase (lines 8-13): Each node $f \in B(v)$ for which:

$$\exists z \in U(v) : z \in C(f) \; and \; z \notin C(w), \; \forall w \neq f \in B(v) \quad (4)$$

is identified and elected immediately as a forwarding node. Then $U(v)$ is updated by removing all the covered nodes and the process is repeated until no node can be elected as forwarder. Note that this is the heuristic proposed by the MPR algorithm [5]. Its rationale is that a node, which is the only one to cover another node in $U(v)$, will, in any case, eventually be elected. Therefore, electing such nodes first reduces $U(v)$, which may potentially lead to the election of less forwarders [9]. After completing the first two steps, EFCN executes the steps defined by GSC (lines 14-19). The algorithm terminates when the set $U(v)$ becomes empty. One would expect that the first step of EFCN is trivial since the basic operation of GSC would naturally lead to the elimination of fully covered nodes. However, we make the observation that eliminating fully covered nodes in advance, increases the probability that the heuristic of the second step will be effective in identifying forwarders, thus minimizing the size of the forwarding set. To illustrate this, let us consider the example depicted in fig. 3. When GSC is used, each one of nodes 2, 3 and 4 may be elected first since they all cover the same number of nodes in $U(k)$. Suppose that node 3 is elected first, then GSC will elect either node 2 and 4 or node 1 and 5. The result is a forwarding set of three nodes. In the case that MPR is used, the result will be the same since there is no node in $U(k)$ that is uniquely covered by a node in $B(k)$. On the contrary, EFCN in its first step will eliminate nodes 1 and 5 since they are fully covered by nodes 2 and 4 respectively. In the second step, nodes 2 and 4 will be elected since they are the only ones to cover some nodes in $U(k)$. At this point the algorithm will terminate since all the nodes in $U(k)$ are covered. It is clear from the previous example that eliminating the fully covered nodes paves the way for the second step of the algorithm, therefore resulting in smaller forwarding sets.

A. Recursive EFCN

As previously discussed, the rationale behind the elimination of fully covered nodes from $B(v)$ is that this procedure increases the probability that the heuristic of the second phase will result in identifying a forwarder. However, note that in the latter case the set $U(v)$ is modified. Since this has an impact on $C(w), \forall w \in B(v)$, it is possible that some of the nodes in $B(v)$

```
1: procedure EFCN((B(v), U(v)))
2:     Fw(v) ← ∅
3:     for all f1 ∈ B(v) do
4:         if ∃f2 : C(f1) ⊆ C(f2) then
5:             B(v) ← B(v) − f1
6:         end if
7:     end for
8:     for all z ∈ U(v) do
9:         if ∃f∈B(v) : [(z∈C(f)) ∧ (z∉C(w), ∀w≠f∈B(v))] then
10:            U(v) ← U(v) − N(f)
11:            Fw(v) ← Fw(v) ∪ f
12:        end if
13:    end for
14:    while U(v) ≠ ∅ do
15:        f = argmax_{w∈B(v)} {|N(w) ∩ U(v)|}
16:        U(v) ← U(v) − N(f)
17:        Fw(v) ← Fw(v) ∪ f
18:    end while
19:    return Fw(v)
20: end procedure
```

Fig. 2. The pseudocode of the EFCN algorithm

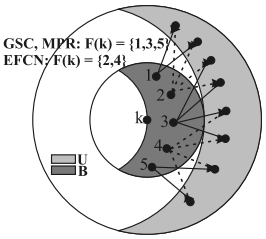


Fig. 3. Example of forwarding set construction for EFCN, GSC and MPR.

are now fully covered by other nodes. Therefore, in order to maximize the algorithm's performance, we propose a modified version of EFCN in which the first two steps are repeated until no forwarder is elected by the heuristic implemented in the second phase of the algorithm since in that case $U(v)$ is not modified. We call this algorithm *Recursive EFCN (R-EFCN)*.

B. Complexity Considerations

It is clear that eliminating fully covered nodes from $B(v)$ comes at a processing cost. Note that identifying fully covered nodes in $B(v)$ is equivalent to the problem of finding maximal sets among $C(f_i), \forall f_i \in B(v)$ (a set is maximal if it is not contained in any other set). Such an algorithm can be implemented in time $O(m \cdot \sum_{\forall f \in B(v)} |C(f)|)$ [10], where $m \leq |B(v)|$ is the number of the maximal sets. It is also interesting that eliminating the fully covered nodes from $B(v)$ has a positive impact on the complexity of the following steps of the algorithm. To explain this, keep in mind that the time complexity of both the second phase of the algorithm as well as the GSC part depends on $|B(v)|$. However, after the first phase, the size of $B(v)$ is only m. Finally, to reduce the time complexity, Bloom filters [11] may be used for representing sets. The tradeoff is that the result of the subset and the set equality operations depend on the false positive rate (P_f) of the filter. However, a negligible P_f is possible by appropriately selecting the size of the filter and the number of hash functions [11]. In this work, we do not consider the use of Bloom filters.

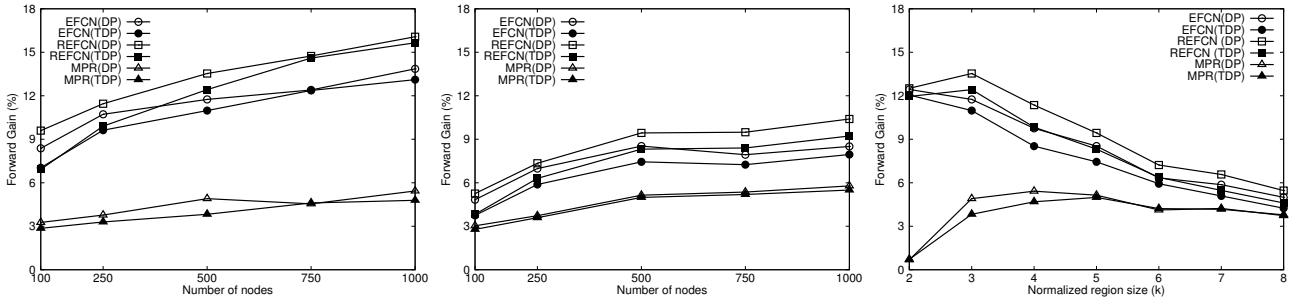
The processing introduced by EFCN can also be justified if we keep in mind that, in the context of wireless ad hoc networks, bandwidth is the resource with the most stringent limitations mostly due to the contention-based nature of access protocols. On the other hand, the constraints on processing power are less severe since there exist powerful and efficient mobile processors [12]. Moreover, reducing the number of transmissions alleviates the processing related to the transmission and reception of packets, e.g. transfer a packet to/from the wireless interface, read packet, etc. Regarding power consumption, researchers have provided evidence [13] proving that the energy consumption in a wireless interface, either for transmitting, receiving or even discarding packets, is in the order of tens of μW. At the same time, state of the art mobile processors need as low as 0.625 nW/cycle [12].

IV. Simulation Framework & Results

To assess the performance of EFCN, we compare it with GSC and MPR. To this purpose, we implemented all three algorithms in the context of two well known broadcast schemes that use the set cover problem to model the selection of forwarders. More specifically, we implemented the DP [3] and TDP [4] algorithms from the dominant pruning class of algorithms. In dominant pruning, a node determines the forwarding set and appends it to the packet. Then, each node in the forwarding set repeats the process unless a termination criterion is met. We implemented the modified relayed/unrelayed termination criterion [14] for both algorithms since it results in the best performance. The difference between DP and TDP is that in DP $U(v)=N(N(v))-N(u)-N(v)$, where u is the previous hop node, while in TDP $U(v)$=$N(N(v))$–$N(N(u))$. In both algorithms, $B(v)=N(v)-N(u)$. In the following, we will use a notation of the form X(Y), where X indicates the algorithm used for selecting the forwarding set and Y indicates the broadcast algorithm, e.g. EFCN(DP) means that EFCN is used to construct the forwarding set while DP is the broadcast algorithm. The simulation study has been carried out with ns2 [15]. The simulation model consists of N nodes distributed in a square area AxA, where A=kR, k∈$[2, 8]$ and R is the nodes' transmission range. Since the objective is to reduce transmissions, we introduce a metric called *forward gain (FG)*:

$$FG_{_{EFCN}} = 1 - fn_{_{EFCN}}/fn_{_{GSC}} \quad (5)$$

where $fn_{_{EFCN}}$ is the number of transmissions when EFCN is used while $fn_{_{GSC}}$ is the same quantity for GSC. Clearly, $FG_{_{EFCN}}$ captures the improvement achieved by EFCN compared to GSC. Similarly, we define $FG_{_{MPR}}$ for the MPR algorithm. Furthermore, we adopt the methodology in [2] and do not consider transmission failures in order to rule out their impact on FG and capture the true performance of each algorithm. For the same reason, we do not consider mobility in the first two experiments. Instead, we determined the positions of nodes using the distribution that is produced when nodes move according to the Random Waypoint algorithm [16]. In the following, we will use RW to denote this distribution. We also report results obtained using a uniform distribution of nodes. In each experiment, we considered 500 trials. In each trial, we produce a different deployment of nodes, according to the aforementioned distributions. Then, a node is randomly selected as the source node and broadcasts a single packet. In the first experiment, we compare the three algorithms for networks of different size. More specifically, we vary the number of nodes ($N \in [100, 250, 500, 750, 1000]$). Two different sizes of the square area are used, namely $k=3$ (i.e. each side of the square area is three times the transmission range) and $k=5$. The latter value has been chosen in order to simulate a low density network but without partitions even when N=100. On the other hand, k=3 corresponds to a relatively dense network with a network diameter more than one hop. Fig. 4(a) and 4(b) illustrate the forward gain with respect to the network size for k=3 and k=5, respectively. EFCN effectively reduces transmissions and achieves a forward gain of up to $\sim 13\%$ when k=3 and $\sim$9% for k=5. Note also that the improvement is evident for both DP and TDP, which proves the generality of EFCN. As expected, the improvement is greater when the network size increases. On the other hand, MPR only manages an improvement of only $\sim 5\%$ in the best case. This is a confirmation of our intuition that the elimination of fully covered nodes from $B(v)$ increases the effectiveness of the MPR heuristic. This is also confirmed by the recursive version of EFCN which further increases the gain to $\sim$16% for k=3 and $\sim$11% for k=5. In the second experiment, we vary the



(a) (b) (c)

Fig. 4. Forward gain (for RW node distribution) versus: a) number of nodes for $k=3$, b)number of nodes for $k=5$, and c) region size for $N=500$.

TABLE I
DELIVERY RATE (%) (k=5, N=100, NODE DIST.:RW)

Max speed (m/sec)	DP				TDP			
	GSC	EFCN	REFCN	MPR	GSC	EFCN	REFCN	MPR
1	99.54	99.53	99.52	99.55	99.53	99.50	99.50	99.51
5	99.44	99.39	99.40	99.43	99.37	99.25	99.25	99.35
10	99.27	99.16	99.14	99.23	98.98	98.83	98.81	98.95
15	99.00	98.95	98.95	98.96	98.56	98.41	98.36	98.43
20	98.80	98.67	98.68	98.72	98.21	97.81	97.77	98.08

size of the square area ($k \in [2,8]$) while N=500. The results (fig. 4(c)) confirm the previous findings. It is worth noting that the gain is smaller when the network density is small. This is reasonable since in that case each node has a small number of neighbors and almost every neighbor is elected as a forwarder due to the sparse connectivity. As a result, there is a little space for improvement. On the contrary, on high density networks, where reducing the number of transmissions is more critical, EFCN works better. Qualitative similar results are obtained when the aforementioned experiments are repeated with a uniform distribution of nodes. For example, the forward gain for EFCN(DP)[REFCN(DP)] ranges from $4.46[4.55]\%$ to $11.96[12.09]\%$ for $k \in [2,8]$ and $N = 500$, while for EFCN(TDP)[REFCN(TDP)] it ranges between $3.52[3.57]\%$ and $10.30[11.88]\%$. Another important aspect of the presented results is related to the average energy consumed per node. Recall, that according to the discussion in Section III-B, energy consumption is dominated by the number of packet transmissions and receptions. Both EFCN and REFCN, besides the reduction of transmissions, manage a similar performance for packet receptions. For example, EFCN(DP)[REFCN(DP)] reduces packet receptions, with respect to GSC, by up to $13.2\%[16.21\%]$ when $k=3$ and $N \in [100,1000]$, while the reduction is up to $7.69\%[9.85\%]$ when k=5 and $N \in [100,1000]$ and up to $10.34\%[10.65\%]$ when N=500 and $k \in [2,8]$. As a result, we expect an analogous performance as far as energy consumption is concerned.

Finally, we evaluate the performance of all algorithms under various mobility levels for k=5 and N=100. More specifically, nodes move according to the random waypoint algorithm with a randomly selected speed in the range $(0, max\_speed]$, where $max\_speed \in [1,20]\ m/sec$. Hello messages with a period of 1 sec are used for collecting neighborhood information. The values of k and N represent a scenario with the lowest density but without partitions. In this way, we minimize packet redundancy therefore we capture the highest possible impact of mobility. Note that, although the algorithms that solve the set cover problem are not responsible for mitigating the impact of mobility, such an evaluation is useful for quantitatively identifying the impact on the delivery rate and fine tuning the underlying broadcast algorithm. Table I presents the delivery rate for all algorithms. It is clear that, when EFCN and REFCN are used, the reduction of the delivery rate compared to GSC is minimal even in high mobility (0.45% in the worst case). At the same time, the forward gain is very close to the values recorded when mobility is not present (a small reduction of 0.36% (0.74%) for EFCN (REFCN) in the worst case).

V. CONCLUSIONS

In this letter we tackled the problem of selecting the forwarding set in the context of broadcast algorithms that utilize the set cover model. We proposed a low cost extension to the well known GSC algorithm, which can be ported to all broadcast algorithms that use GSC, and proved that a significant reduction of the transmissions is possible, especially in networks of high density.

REFERENCES

[1] S.Y.Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom*, August 1999, pp. 151–162.

[2] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proc. of MobiHoc '02*, 2002, pp. 194–205.

[3] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, pp. 353 – 363, 2001.

[4] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 111–122, 2002.

[5] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proc. of HICSS'02*, 2002, pp. 3866–3875.

[6] L. Li, R. Ramjee, M. Buddhikot, and S. Miller, "Network coding-based broadcast in mobile ad-hoc networks," in *Proc. of INFOCOM 2007. IEEE*, 2007, pp. 1739–1747.

[7] S. Agathos and E. Papapetrou, "Efficient broadcasting using packet history in mobile ad hoc networks," *Communications, IET*, vol. 5, no. 15, pp. 2196–2205, 2011.

[8] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[9] S. S. Skiena, *The algorithm design manual*. New York, NY, USA: Springer-Verlag New York, Inc., 1998.

[10] D. M. Yellin, "Algorithms for subset testing and finding maximal sets," in *Proceedings of SODA '92*, 1992, pp. 386–392.

[11] J. Blustein and A. El-Maazawi, "Bloom filters. a tutorial, analysis, and survey," Technical Report CS-2002-10. [Online]. Available: http://www.cs.dal.ca/research/techreports/2002/CS-2002-10.pdf

[12] (2011, Sep.) Cortex-a9 processor. [Online]. Available: http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[13] L. Wang and J. Manner, "Energy consumption analysis of wlan, 2g and 3g interfaces," in *Proceedings of GREENCOM-CPSCOM '10*. IEEE Computer Society, 2010, pp. 300–307.

[14] L. Zhu, B. Lee, B. Seet, K. Wong, G. Liu, S. Huang, and K. Lee, "Performance of new broadcast forwarding criteria in manet," in *ICOIN '04*, 2004, pp. 34–43.

[15] K. Fall and K. Varadhan, "The ns manual," VINT Project, Univ. California, Berkeley, CA, 2001.

[16] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, pp. 257–269, 2003.