Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol II
IMECS 2008, 19-21 March, 2008, Hong Kong

Multiple Disjoint Paths Routing Using Implicit Source Routes in Wireless Ad Hoc Networks

Su-Kit Tang, and Dongyang Long

Abstract—**In this paper, we propose a Multiple Disjoint Path (MDP) routing protocol to maximize network throughput and minimize the protocol overheads in wireless ad hoc networks. MDP consists of two components: virtual source routing and core heuristic. Virtual source routing constructs virtual paths that do not suffer from scalability, privacy and efficiency problems caused by long paths for DSR in large networks. Core heuristic is to ensure the efficiency of request packet propagation in route discovery operation. Our simulation results reveal that MDP performs at a satisfactory level in dense networks in terms of connectivity and transmission efficiency. Network resources can be utilized so that network throughput can be significantly increased at a cost of minimal protocol overheads comparatively.**

Index Terms*—*Network, Ad hoc, Multipath, Routing

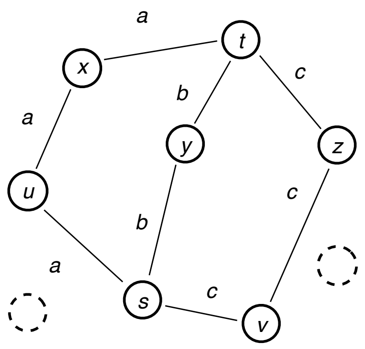

Figure 1. Transmission of Data Packets by Disjoint Paths.

I. Introduction

On-demand routing protocols, such as AODV [1] and DSR [2], have been proposed for efficient communication in wireless ad hoc networks. These protocols discover the most feasible path and maintain routes between two nodes when communication is needed. Single path is constructed by broadcasting request packets from a source node to a sink node. However, in any network, there may be more than one route to sink nodes. To utilize the network resources, many on-demand multipath routing protocols have been proposed [3-8]. They build maximal multiple disjoint paths between two nodes for performance improvement, such that network throughput can be increased. Figure 1 shows a transmission of data packets by three disjoint paths from source node *s* to sink node *t*. Three data packets can be sent by three different paths. In the meantime, the amount of energy consumed for the delivery of same amount of data by one single path can be shared by more nodes in different paths. This optimizes the usage of paths discovered in path discovery and maintenance operations. Therefore, the routing costs imposed to routing protocols for topology changes can be minimized.

In this paper, we propose a Multiple Disjoint Path (MDP) routing protocol. MDP constructs maximal disjoint paths by extending Split Multipath Routing (SMR) [3] with implicit source routing [9]. MDP consists of two components: virtual source routing and core heuristic. Virtual source routing constructs virtual paths that do not suffer from scalability, privacy and efficiency problems caused by long paths for DSR in large networks. Core heuristic is to ensure the efficiency of request packet propagation in route discovery operation. In section II, we review some routing algorithms for wireless ad hoc networks that are crucial to MDP. In section III, MDP will be proposed and discussed. We also have a simulation in section IV to verify the correctness of MDP. From the simulation result, we observe that MDP can dramatically increase the network throughout at a satisfactory level. The amount of overheads it imposes is significantly reduced. Lastly, we conclude this paper.

This work was partially sponsored by the Natural Science Foundation of China (Project No. 60273062, 60573039) and the Guangdong Provincial Natural Science Foundation (Project No. 04205407, 5003350)

Mr. Su-Kit Tang is a Doctor candidate of the Sun Yat-Sen University, GuangZhou, China. He is also with the Macao Polytechnic Institute, Macao, China (phone: 853-599-6440; email: sktang@ipm.edu.mo)

Professor Dongyang Long is with the Sun Yat-Sen University, GuangZhou, China (email: issldy@mail.sysu.edu.cn)

II. Related works

MDP maximizes network throughput in wireless ad hoc networks, by extending SMR with implicit source routing. SMR, based on DSR, builds maximal disjoint paths. Therefore, DSR is also crucial to MDP. DSR [2] is a single path routing algorithm that setups a path from sender to destination by forwarding the address of each node in a path to the destination. Sender will maintain a route cache that it has learned. If a route is found in the cache, it uses the route to transmit the packet. If a route is not found, it initiates a route request and the route request will be propagated in the ad-hoc network until it reaches the destination. When the destination receives the route request, it will return the route record (The path), in which it contains all intermediate nodes' addresses along the trip, to the sender. Significantly, DSR suffers a scalability problem that the propagation of route record may involve unexpected size of source routing information. This overhead creates impact on limited wireless bandwidth. Therefore, Hu proposed the use of implicit source routing for single path routing in wireless ad hoc networks that preserves the advantage of source routing while avoiding the associated per-packet overhead in most cases [9]. It introduced a cache table in each participating nodes. This cache table maintains a list of next hop addresses for each particular routing path, indexed by a flow identifier. A source may then send any packets headed by a flow identifier in lieu of a source route. This avoids of the overhead caused by the source route. In SMR, an extension to DSR, disjoint paths are constructed by spreading request paths in a network. Its objective, which is similar to ours, is to build maximally disjoint paths for load balancing transmissions over the network. Data traffic is split into multiple routes to avoid congestion and to use network resources efficiently. Similarly, a number of solutions that build multiple disjoint paths for performance improvement have been proposed too [3-8]. They use various approaches to construct disjoint paths. However, they are extensions to DSR. They also suffer from unexpected length of route record in large networks.

III. THE PROTOCOL

In this paper, we propose a Multiple Disjoint Path (MDP) construction algorithm to construct disjoint paths from a source node to a sink node. MDP constructs disjoint paths by virtual source routing and the core heuristic. Virtual source routing delivers maximal disjoint paths running from a source node to a sink node. Core heuristic is to ensure the efficiency of request packet propagation in a network.

When a source node initiates communication with a sink node, request packets will be generated and broadcasted by the sink node. Request packets are spread across the wireless network until they reach the sink node. As request packets travel, intermediate nodes, receiving these packets, are required to process and rebroadcast the packets. A simple processing job on request packets involved in a node is to extend a virtual path by assigning sequential virtual address of a virtual path to the node. Since communication in wireless network is done by broadcasting, duplicated request packets of a sink node is expected to happen. Nodes have to make correct and accurate decision on request packet processing and forwarding.

A. Virtual Source Routing

The virtual source routing (VSR) constructs virtual paths that contain virtual addresses of intermediate nodes a request packet has visited. Each virtual path is identified by a unique virtual path id, *vid*, which is generated by the first node after source node. Along with a hop counter of a virtual path, *c*, virtual addresses of a node in a path can be determined. A node is identified by a virtual address in a virtual path. No address privacy issues would be raised among nodes if this is a concern. As a request packet travels, *c* in virtual address is incremented at each node. This maintains the sequential order of virtual addresses in a path virtually. A virtual path is always determined by *vid* so nodes in the same path would always have the same *vid*. Virtual paths of same session will denote by *b*, which groups the paths by source and sink nodes for one communication.

VSR creates a virtual path by spreading request packet *r*. When a source node, *s*, initiates a transmission session, it broadcasts a request packet *r*. A node first receiving *r*, as indicated by *vid*=0, will randomly generate a large number for *vid*. *vid* is assumed to be unique as it is sufficiently large and collision-free. The hop count, *c*, indicates the position of this node in a virtual path as well as the length of the path at source nodes. Neighbor nodes receiving *r* will verify whether *r* is new at current time unit by *vid* and *c*. If it is new, *c* are incremented and *r* will be rebroadcasted. Otherwise, *r* will be ignored. This avoids the broadcast storm problem discussed in [10]. This processing repeats all the way in a virtual path at nodes receiving *r* until the sink node, *t*, is reached. Thus, a virtual path *p* is created. Figure 2 shows an overview of the virtual source routing operation.

In Figure 2, a virtual path is constructed and initiated by a source node *s* in session *b*. First, *s* broadcasts a request packet *r* that containing *vid=0,* and *c=0*. Node *u* receives *r* and rebroadcast *r* after generating *vid=a* and incrementing *c* by one. Subsequent node *v* receiving *r* will increment the

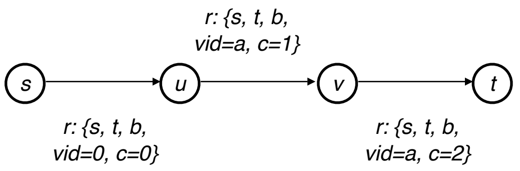

Figure 2. Virtual Source Routing Operation

path variable *c* by one, and rebroadcast *r*. Gradually, sink node *t* will receive *r*, from node *v*, containing *vid=a*, and *c=2*, which indicates a 2-hop virtual path identified by *vid=a*. The virtual path $\{t \rightarrow u \rightarrow v \rightarrow s\}$ is constructed.

A node receiving a request packet *r* will propagate and maintain an entry for *r* in its virtual path table, *VPT*, until *r* expires. Entries in virtual path table are maintained in cache for a period of time. It records all virtual paths that the node resides until timeouts. Since *vid* and *c* can uniquely identify a node in a path, quick routing decision can be made.

B. Core Heuristic

We define disjoint paths as non-intersecting paths, running from a source node to a sink node. These paths do not consist any common intermediate nodes in the same session if they are parallel (See Figure 1). To ensure that paths are disjoint with each others for a sink node, a path discovery heuristic is required to verify the originality of request packets. This enables a node to determine whether a request packet should be accepted. Note that virtual paths for different source or sink nodes are assumed to be of no conflict of interest, so they would not be considered in the heuristic. The heuristic includes the following rules.

Rule 1 (Non-broadcast storming) A request packet will only be accepted if virtual path table does not show an entry of the same source and sink nodes at the same session.

A request packet is used to initiate a communication from a source node to a sink node. When a node receives the request packet, it can determine if the request packet should be accepted by looking up its virtual path table. If an virtual path entry is not found, the request will be accepted, processed and rebroadcasted.

Rule 2 (Non-intersecting) A request packet will be accepted if virtual path table shows an entry of the same source and sink nodes at the same session that is for different virtual paths but of bigger hop counts.

Request packets are running directionless and a node *v* may receive two request packets initiated by the same source node and destined at the same sink node at same session. Since *vid* is collision-free, these two request packets would have their own unique *vid*. Running both request packets through node *v* would lead to a single delivery path only. Therefore, node *v* can only take either one of the two request packets. Otherwise, this evaluation of request packets raises an optimization issue in this situation.

Rule 2.1 (Selection criteria) A request packet will be accepted if virtual path table shows an entry of the same source and sink nodes at the same session for the different virtual path that its hop counter is not better than the newly received one. Virtual path table will be updated with this request packet.

In this paper, the term *optimal* is defined to be minimal hop count as the amount of energy consumed in a transmission would be less, assuming that the energy consumption for one broadcast operation is same for all nodes. The shorter the path is, the less the energy it consumes, in one transmission. Based on this rationale, a node will accept a request packet of smaller hop counter. This will trigger a rebroadcast of request packet with updated information for the change of the path segment constructed previously if better request comes after. In case of same hop counters, a request packet from a farthest distance is in preference comparatively. This can encourage the construction of shortest path in a dense area. However, determination of the distance of neighbor nodes is out of scope of this research.

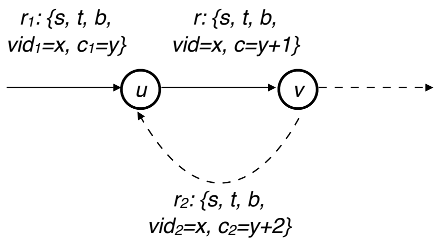
r_1: {s, t, b, vid_1=x, c_1=y}
r: {s, t, b, vid=x, c=y+1}
u
v
r_2: {s, t, b, vid_2=x, c_2=y+2}

Figure 3. A node *u* rejects a loop-back request packet *r* from node *v* during path construction in MDP algorithm

Rule 3 (Loop-free) A request packet will not be accepted if virtual path table shows an entry of the same source and sink nodes at the same session for the same virtual path with smaller hop counter. Suppose that node *u* is somewhere after node *v* in a path construction from source node *s* to sink node *t* at time session *b*, shown in Figure 3. Request packet r_2 of ($vid_2=x$, $c_2=y+2$) from node *v* is running back to node *u*. Virtual path table of node *u* shows r_1 of ($vid_1=x$, $c_1=y$) where $vid_1 = vid_2$ and $c_2 > c_1$. From Rule 2.1, node *u* will not accept *r*. In addition, a condition $vid_1 = vid_2$ holds. We can see that both requests are of same originality and it is a loop.

Rule 4 (Shadow path) A shadow path is a path that is a segment of another path. Suppose that there are two request packets running the same virtual path, only one with smaller hop counter will be accepted. This minimizes the length of a virtual path.

```
Algorithm MDP(r)
1.  if this is a sink node then
2.    t ← this
3.    s ← source node (destination)
4.    b ← now
5.    vid ← 0
6.    c ← 0
7.    broadcast r = (t, s, b, vid, c)
8.  else if this is an intermediate node then
9.    result = VPT_lookup(r)
10.   if result = null then
11.     VPT_add(r)
12.   else if result = path_found then
13.     if r and result are from diff paths then
14.       if r is better than result then
15.         VPT_replace(r)
16.       end if
17.     else if r and result are same path then
18.       if r.c > result.c then
19.         do nothing...loop
20.       else if r is better than result then
21.         VPT_replace(r)
22.       end if
23.     end if
24.   end if
25.   broadcast r = (t, s, b, vid, c + 1)
26. else if this is a source node then
27.   result = VPT_lookup(r)
28.   if r and result are not shadow then
29.     VPT_add(r)
30.   end if
31. end if
```

Based on the rules discussed above, we have the pseudo-code of MDP presented below. The MDP will take a request packet *r* as a parameter to implement the Rule 1 by lines 10 to 11; Rule 2 by lines 12 to 16; Rule 3 by lines 12, 17 to 19 and Rule 4 by lines 12, 17, 20 to 22. Depending on the type of a node, MDP(r) will do corresponding tasks. If it is a sink node, it initiates *r* and broadcast *r*. If it is an intermediate node, it follows the core heuristics to process *r*. If it is a source node, it accepts request packet *r* if it is not from a shadow path.

IV. Performance Evaluation

To evaluate the performance of MDP, we implemented MDP in ns2 [11], an open source network simulator, and conducted a set of simulations using the following settings: IEEE 802.11b standard at MAC layer implementation; 100 nodes randomly placed on areas of $500\times500m^2$, $1500\times1500m^2$, $3000\times3000m^2$ for different network density environments. This demonstrates the efficiency of MDP against DSR in terms of overheads; 100 randomly generated scenarios for each area. In our simulation, a run was conducted for each scenario at each area and collected data was averaged over these 100 scenarios.

In our simulation, we consider four metrics to evaluate the performance of MDP.

(a) Packet Delivery Ratio (*PDR*): The rate of delivering data packets successfully to sink nodes, which is defined as:

$$PDR = \frac{R}{S} \times 100\%$$

where *R* is the number of received packets by the sink nodes and *S* is the number of sent packets by the source nodes.

(b) Packet Delivery Latency (*PDL*): The amount of time required to deliver a data packet from the time it is generated in a source node to the time it arrives at a sink node, which is defined as:

$$PDL = \frac{D}{R}$$

where *D* is the total amount of packet delivery time and *R* is the number of received packets sent by source nodes.

(c) Overheads: The total amount of header data required by the routing protocols for the delivery of a data packet along a routing path.
For the delivery of a data packet, the overheads of DSR is the number of times a routing path that are required to transmit. The overheads of DSR is calculated by the product of average length of routing path, the length of address node and the average hop counts. The average hop counts is determined by the number of sent and forwarded packets divided by the number of sent packets. Since the destination node address is also included in a path, the length of a path is determined by adding one to the average hop counts. Therefore, we define the overheads of DSR as:

$$Overheads_{DSR} = \left(\frac{S+F}{S} + 1\right) \times A \times \left(\frac{S+F}{S}\right)$$

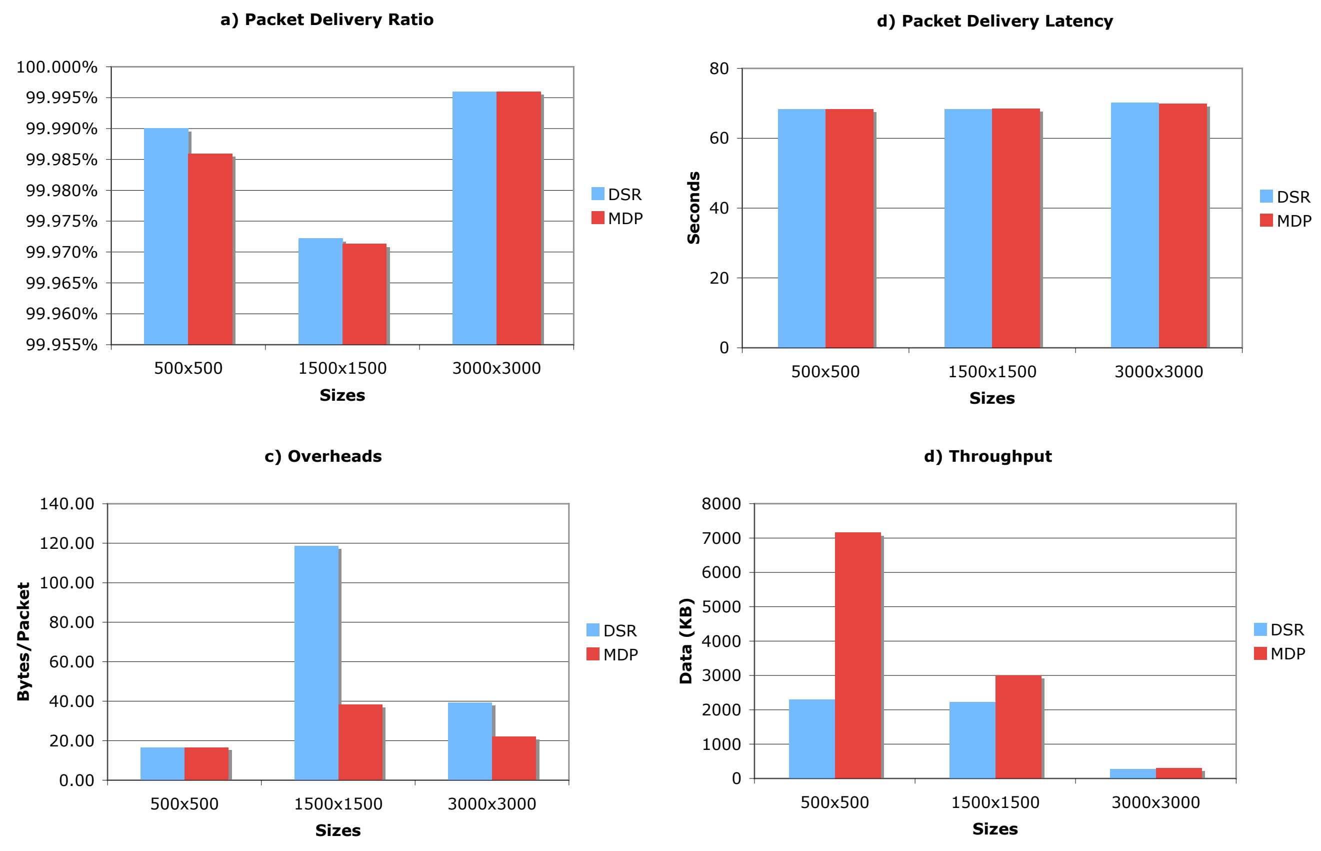

Figure 4. Simulation results of MDP against DSR in NS2

where F is the number of forwarded packets by intermediate nodes and A is the length of node address. The overheads of MDP is the number of times routing information, which includes a time session, a virtual path id, a hop counter and a packet type, that are required to transmit. The overheads of MDP is calculated by the length of routing information times the average hop counts, which is defined as:

$$Overheads_{MDP} = H \times \left(\frac{S + F}{S} \right)$$

where H is the routing information required by MDP.

(d) Throughput: The total amount of data delivered to sink nodes successfully in a certain period of time. In our simulation, we use the same transmission rate for MDP and DSR. To obtain a throughput gained by disjoint paths, MDP will generate packets for all available paths in each transmission.

We observe that MDP does work as efficient as DSR in terms of packet delivery ratio and packet delivery latency. Recall that both of them deliver packets by routing paths in on-demand manner. In Figure 4a, even though *PDR* of two protocols reaches 99.9%, but DSR could perform better than MDP. It is because the utilization of network capacity in high dense networks by MDP may create a traffic congestion as the number of disjoint paths increases. Therefore, packets may be dropped. In sparse networks, this problem is resolved. The number of disjoint paths that can be constructed is limited. Connectivity among nodes is comparatively lower and the amount of traffic generated is limited too. But in Figure 4b, *PDL* does not show any difference between two protocols. In Figure 4c, MDP outperforms in terms of overheads required in routing operation. The amount of overheads MDP generated is

Network size (m^2)	500x500		1500x1500		3000x3000	
Routing Protocols	DSR	MDP	DSR	MDP	DSR	MDP
Packet Delivery Ratio (%)	99.99	99.986	99.972	99.971	99.996	99.996
Packet Delivery Latency (s/pkt)	68.3	68.32	68.26	68.46	70.13	69.87
Overheads (Bytes)	16.7	16.69	118.73	38.36	39.38	22.23
Throughput (Kilobytes)	2305.26	7157.01	2229.99	2997.51	273.28	302.84

Table 1. Summary of MDP Performance against DSR

significantly less than DSR does. In dense networks, as the length of routing paths is short, the amount of overheads from DSR and MDP is just slightly different. Once the network size increases, the length of routing path gets long and the amount of overheads increases. Overheads of DSR increase. Since the overheads of MDP is constant, its overheads remains at low level. In the meantime, we observe that the transport capacity of a network increases by network connectivity. Disjoint paths maximize the network throughput. In Figure 4d, in high dense networks, MDP shows a good result that three times more than DSR in throughput is achieved. As the connectivity of network nodes determines the number of disjoint paths, in sparse networks the throughput of MDP gets close to DSR. A summary of our simulation results is shown in Table 1.

V. Conclusion

In this paper, we proposed a multiple disjoint paths construction algorithm, called Multiple Disjoint Path (MDP), to utilize network resources. MDP constructs maximal disjoint paths using implicit source routes. It resolves scalability, privacy and efficiency problems caused by long paths for DSR in large networks. Our simulation reveals that MDP performs at satisfactory level. From the aspects of connectivity and transmission efficiency, efficiency of MDP is significant in dense networks due to high connectivity. Network resources is utilized as network throughput is significantly increased. In the meantime, the amount of overheads in packet header required by the routing protocol is significantly reduced comparatively.

References

[1] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, “Ad Hoc On-demand Distance Vector (AODV) Routing”, *IETF RFC 3561*, 2003.

[2] D. Johnson, D. Maltz & Y. Hu, “Dynamic Source Routing in Ad Hoc Wireless Networks”, *Internet Draft*, draft-ietf-manet-dsr-10.txt, work in progress, July 2004.

[3] S. J. Lee, and M. Gerla, “Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks”, *IEEE International Conference on Communications*, Helsinki, Finland, Vol. 10, 2001, pp.3201-3205.

[4] M. K. Marina and S. R. Das., “Performance of Route Caching Strategies in Dynamic Source Routing”, *Proceedings of the 2nd Wireless Networking and Mobile Computing (WNMC)*, Mesa, AZ, USA, April 2001, pp.425-432.

[5] Wei, W.; Zakhor, A., “Robust multipath source routing protocol (RMPSR) for video communication over wireless ad hoc networks”, *IEEE International Conference on Multimedia and Expo*, Taipei, Taiwan, Vol. 2, June 2004, pp1379-1382.

[6] N. Wisitpongphan, and O. K. Tonguz, “Disjoint Multi-path Source Routing in Ad Hoc Networks: Transport Capacity”, *Proceeding of IEEE Vehicular Technology Conference*, Florida, USA, Vol. 4, Oct. 2003, pp.2207-2211.

[7] B. Yan, and H. Gharavi, “Multi-Path Multi-Channel Routing Protocol”, *Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications*, Massachusetts, USA, 2006, pp. 27-31.

[8] A. Srinivas and E. Modiano, “Minimum energy disjoint path routing in wireless ad-hoc networks”, *Proceedings of the 9th annual international conference on Mobile computing and networking*, California, USA, Sept. 2003, pp.122-133.

[9] Y.-C. Hu & D. Johnson, “Implicit Source Routes for On-Demand Ad Hoc Network Routing”, *Proceeding of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, Long Beach, CA, USA, pp.1-10, October 2001.

[10] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, “The Broadcast Storm Problem in a Mobile Ad Hoc Network”, *ACM/IEEE Mobicom*, Seattle Washington, USA, August 1999, pp. 151-162.

[11] Network Simulator (NS-2), Available at http://www.isi.edu/nsnam/ns/index.htm