SEARCHING FOR RESOURCES IN MANETS:
A cluster based flooding approach

Rodolfo Oliveira, Luis Bernardo, Paulo Pinto
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, P-2829-516 Caparica, Portugal

Keywords: Performance analysis of wireless ad hoc networks, searching service, clustering protocol.

Abstract: In this paper, we propose a searching service optimized for highly dynamic mobile ad-hoc networks based on a flooding approach. MANETs unreliability and routing costs prevent the use of central servers or global infra-structured services on top of a priori defined virtual overlay networks. A flooding approach over a virtual overlay network created on-demand performs better. Flooding is supported by a light-weight clustering algorithm. The paper compares the relative efficiency of two clustering approaches using 1.5-hop and 2.5-hop neighborhood information, and of a non-clustered approach. It presents a set of simulation results on the clustering efficiency and on searching efficiency for low and high mobility patterns, showing that the 1.5-hop algorithm is more resilient to load and to node movement than the 2.5-hop algorithm.

1 INTRODUCTION

The problem of looking for resources on 802.11 Mobile Ad hoc NETworks (MANETs) is complex due to the networks unstable nature. Nodes move around independently creating a very dynamic network topology. It is assumed that no geographic position information is available, which is most of the time true, mainly in indoor scenarios. MANET routing protocols can be seen as resource lookup service that look for IP addresses.

Experience with fast moving nodes (Tsumochi, 03) showed that standard proactive, table-driven, routing protocols perform worst than on-demand routing protocols, which flood the network looking for an address only when it is needed. It also shows that both approaches fail to handle extreme mobility conditions. The problem is that routing information becomes outdated too fast, especially for lengthy paths. Due to bandwidth restrictions, it is not feasible to maintain proactively the tables always updated. On-demand approaches fail due to packet collisions and due to the breaking of the return path in result of intermediate node movement. These conclusions are extensible to generic searching services implemented at application layers. Structured peer-to-peer p2p (services) and directory services have much higher update costs than flooding based services (Bernardo, 04). A flooding approach is more adapted to unstable MANETs due to the null registration costs. All efforts are concentrated during the search phase.

The searching protocol performance depends strongly on the lower layers of the protocol stack, responsible for routing IP packets, and for handling the Medium Access Control (MAC). Traditional flooding peer-to-peer (p2p) services create virtual overlay networks. They are formed by several nodes connected using static TCP links. Their performance drops sharply on a MANET if the virtual overlay topology is not similar to the network physical topology, due to the routing protocol overhead. Crossing a virtual link may lead to a route recovery procedure (usually a network flood) if the MANET topology changes.

The MANET routing protocol overhead can be avoided if the searching protocol's query message is broadcasted, hop by hop, during the searching flood (e.g. ad hoc mode of the JXTA rendezvous protocol (JXTA, 04)). However, two problems may occur: the 802.11 MAC layer is more error prone for multicast/broadcast packets than for unicast packets, and dense networks may suffer from the broadcast storm problem (Tseng, 02). This latter problem can be minimized organizing nodes into clusters, and reducing the number of nodes sending messages to the network.

This paper presents a new searching algorithm, optimized for very dynamic continuous MANETs. It focuses mainly on the clustering algorithm. Section two overviews existing cluster based searching algorithms. The proposed clustering and searching

Oliveira R., Bernardo L. and Pinto P. (2005).
SEARCHING FOR RESOURCES IN MANETS - A cluster based flooding approach.
In *Proceedings of the Second International Conference on e-Business and Telecommunication Networks*, pages 105-111
DOI: 10.5220/0001417001050111

protocols are presented on sections three and four. Section five presents the ns-2 simulation setup, and several performance measurements. Finally, section six draws some conclusions and presents future work directions.

2 CLUSTER BASED SEARCHING

Cluster based searching approaches group nodes into broadcast groups (BGs), and using a set of heuristics select BG leaders (BGLs), responsible for forwarding packets for their BG. Nodes periodically broadcast a beacon packet, that may carry (Wu, 03): local information (1-hop); its BGL (1.5-hop); a neighbor node (within radio range) list (2-hop); the neighbor's list with the BGLs information (2.5-hop); etc. Most MANET protocols adopt a 2-hop or above approach (e.g. OLSR (Jacquet, 01)). 2-hop is the minimum information required to define a set of active nodes that cover all nodes, usually called a Connected Dominant Set (CDS). Although, on an unstable MANET, it is possible that 2-hop and further distant neighbors information is out-of-date, introducing errors on the CDS construction that result in failure to cover all nodes. Additional errors may result from the impossibility of revoking explicit state configurations created using signaling (e.g. OLSR BGL election).

Another approach is to adopt 1-hop strategies (e.g. SBA (Peng, 00)), just for maintaining the list of neighbors, and to use an external searching protocol for restraining the number of active nodes. In SBA, nodes delay the sending of query messages for a random time waiting for its possible transmission by other neighbors. Nodes include their neighbor list (Nq) on the query message before sending it. Receivers store the union of Nq lists received (Nu) and compare it with their list of neighbors (Nr), canceling the transmission when Nu is equal to Nr. ABC-QS (Choi, 02) modifies the forwarding rule proposed by SBA reducing searching delay: nodes do not delay the query message sending if the number of neighbors in Nr and not in Nq is above the number of neighbors common to Nr and Nq. Otherwise, they delay an average time proportional to the number of nodes in Nr and not in Nq. ABC-QS may fail for dense MANETs where overlapped nodes may send packets without waiting.

MANETs are not homogeneous. Some nodes stay together during a large period of time (e.g. students on a bus tour) while others move independently. Beacons can also be used to detect relative stability relationships. Toh introduced the concept in ABR (Toh, 97), measuring the number of beacons received. ABC-QS extended the metric to cope with asynchronous piggybacked beacons. Other authors introduced link stability measurements based on packet probability failure (McDonald, 99). Nevertheless, most clustering approaches do not take link stability into consideration (OLSR, etc.), producing unstable clusters for unstable MANETs.

ABC-QS and (McDonald, 99) create proactive routing information within islands of stable connected nodes, to speed up searches. However, they ignore the stability information for thorough flooding network searches that cover several stable islands. This paper proposes a new solution, which improves flooding using stability information.

3 CLUSTERING ALGORITHM

The proposed clustering algorithm groups "stable" nodes into 1-hop radius clusters. Each node selects a BGL periodically using a local soft-state protocol. The resulting network simplified view is used to reduce the flooding search overhead.

Each node periodically broadcasts a beacon message. All nodes that receive a beacon from a node n_y are defined as n_y neighbor nodes. Nodes keep a table of neighbors' link stability η (called the beacon table). Following ABR, link stability for n_y is defined as the sum of consecutive beacons received from n_y. If more than one beacon is lost, then link stability is set to null. The stability measurement trades-off a faster link failure detection (compared to packet loss rate measurements) for a higher probability of false link loss detection due to two successive beacon collisions. High stability values represent low nodes relative mobility and vice-versa.

In each beacon message, a node sends its node identification, its BGL node address, and the higher link stability value contained in its beacon table, which is represented by μ. The beacon table includes the neighbor's address, their link stability (η); their BGL address; and the μ value received in the last beacon. Every beacon table entry is automatically destroyed if a beacon is not received during two beaconing time periods. Table 1 is a hypothetical beacon table of node 3 illustrated in figure 1. Node 3 received 43 beacons from neighbor node 1.

A node is stable if there is at least one η value in its beacon table that is higher than a defined `stability_threshold`. BGL selection algorithm is run on each node before sending a beacon. The selection algorithm for node n_a is summarized in figure 2.

Table 1: Beacon table of node 3 on figure 1

Neighb.	Stability (η)	BGL	Neighb. Stability (μ)
1	43	1	43
2	8	6	64
4	2	5	33

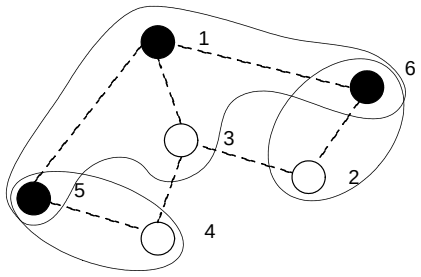

Figure 1: Illustration of a MANET with 3 BGs. Nodes 1, 5 and 6 are BGLs.

```
1.  (η_max)=find_maximum_η_value_in_table()
2.  last_addr = MAX_INT
3.  pre_selected = -1
4.  if is_stable(n_a)    // stable node
5.    //insert all known BGL's stable neighbor
      //nodes in BGL_list
6.    for each neighborhood_node n_x
7.      insert_in_sort_list(BGL(n_x),BGL_list)
8.    if is_BGL(n_a) // if this node is BGL
9.      insert_in_sort_list(n_a,BGL_list)
10.   // Choose BGL based on stability and
      // lowest address criteria
11.   for each bgl_x contained in BGL_list
12.     for each neighborhood_node n_x
13.       if ((n_x=bgl_x)and(is_stable(n_x)))
14.         pre_selected = n_x
15.     if (pre_selected ≠-1) break;
16.     if (n_a=bgl_x) // self-selection
17.       pre_selected = n_a
18.       break
19.   //select new BGL
20.   if (pre_selected=-1)//BGL is not selected
21.     for each neighborhood_node n_x
22.       if (η_max-η(n_x)-transient_threshold ≤ 0)
                        ∧ (addr(n_x)<last_addr)
23.         last_addr = addr(n_x)
24.         pre_selected = n_x
25. BGL_SELECTED = pre_selected
```

Figure 2: BGL node selection algorithm applied in node n_a.

A stable node first computes a sort list of all available neighbor's BGL (lines 5 to 7), that includes the node in case of being BGL (lines 8 to 9). This list is sorted from the smallest to the largest BGL address. If there are BGLs selected in the neighborhood, the node chooses the BGL that has the lowest address (lines 11 to 15), which can be the node itself if it was chosen as BGL by a neighbor (lines 16 to 18). If there are no BGLs selected in its neighborhood, a node simply selects as BGL its neighbor with the highest η value (lines 20 to 24). If there is more than one neighbor owning the maximum η value then it is selected the node with lowest address.

During system startup, transitory cluster overlap may appear, because the initial criteria for selecting BGL is a local measurement for link stability (lines 21 to 23), which may differ from node to node. The `transient_threshold` was set to one, to compensate different beacon delivery time drifts (jitter). The initial BGL is the neighbor with the lowest address that could get the maximum stability value during the present beaconing period. Yet, when several BGLs exist within radio range connected by stable links, they are merged into a single cluster (lines 10 to 18) after one beacon period. Two nodes from overlapped clusters sort neighbor's BGLs independently into the same order (only node address is considered) and converge to the same BGL.

Cluster overlapping also occurs for continuous groups of stable nodes wider than one hop. The algorithm leads to the construction of multiple tree structures of BGLs, called cluster-trees, centered on BGLs with local minimum addresses. Each branch has a sequence of BGLs (n_i) with increasing addresses, whose BGL is the branch predecessor ($BGL(n_i) = n_{i-1}$). The exception is the periphery of the cluster-tree, where nodes with lower addresses can exist. Due to line 16, a node can only self-select as BGL if another node previously selects him. This avoids the existence of single node clusters on the periphery of a cluster-tree.

Within a connected stable group (a group of cluster-trees connected by stable links), the border between cluster-trees' BGLs is composed by one or two non-BGL nodes. It cannot be zero because lines 12-15 would merge the BGLs. Also, it cannot be more than two because that would mean that a node would not have a BGL in the neighborhood, and lines 20-24 would select a new BGL.

Figure 1 presents a cluster-tree with a root BGL (node 1) and two branch BGLs (nodes 5 and 6). Node's 6 BGL is node 1, but node 6 is also a BGL selected by node 2. Node 2 will only form an independent cluster-tree if a new node creates a stable link and selects him as BGL.

The clustering algorithms' performance depends on the network stability. If a large percentage of the nodes are stable, the algorithm is able to detect them, and reduce their load by grouping them in clusters. If all nodes are unstable, beaconing only introduces overhead. A lower beacon period value tolerates higher nodes velocity. However, it increases the bandwidth overhead and the network collisions. It is better to reduce the clustering overhead and increase

the flooding algorithm redundancy, to tolerate clustering inaccuracies. If conventional criteria were used, the clustering algorithm would create highly unstable clusters, which would include passing-by moving nodes, and would route query packets based on this error prone information.

4 SEARCHING ALGORITHMS

The searching algorithms were developed as an evolution of the basic source routing flooding algorithm (SR). In SR the lookup operation is started with a query message originated by a source node, which carries a unique identification (Q_{id}), the source node address (n_{source}), a resource identification pattern to locate (R_{id}), and the path (P). This message is successively resent by each node, as long as it has not been received before and the hop limit is not reached. Each sender appends its identification to P. Nodes maintain a local table indexed by source node id, with last query' ids received. A hit message is sent to the source node when any local information satisfies the query. Hits are routed to the query's node source using the path included in the query message.

This paper proposes 1.5-hop and 2.5-hop algorithms that enhance SR flooding phase, reducing its overhead, and the hit message routing, improving its resilience to node movement and failure. SR is modified in three ways:

(a) The number of active nodes is reduced using the clustering node information. A node can be: a BGL if it receives a beacon selecting it; a non-BGL if it selects a BGL but is not selected a BGL; or isolated if it does not select a BGL. An unstable node with one or more stable nodes in its neighborhood selects for BGL the node with the highest μ value, strictly for flooding purposes. Two approaches are presented above;

(b) Query message size is reduced by removing all non-BGLs and isolated nodes' ids before the last BGL from the path field (P). The partial path is stored and pruned, each time the message passes on a BGL. In case of node failure, the node can always use the BGL list (stored in the query message) to recover the route to the source node;

(c) When hit messages follow the query reverse path, unicast is used and their sending is confirmed. When a link fails, the node looks at its neighbor list, and neighbor's BGL list, looking for any node on the reverse path. As a last resort, when no information is available, the node that detects the failure starts a hit message flooding. The hit message is treated as a special query packet, looking for a node id within the remaining query path list, which does not receive any reply. Hit flooding stops when the message reaches a node whose neighbor's (or the node itself) are part of the remaining path. Therefore, contrary to SR, the proposed algorithm is able to survive to extreme mobility, and is able to route hit messages over failed or moving nodes.

A. 1.5-hop searching algorithm

BGL and isolated nodes always broadcast queries one time (though isolated delay message transmission). A non-BGL delays the query sending for a fixed delay plus a jitter interval, and lists the visited BGL on a local variable. While the timer is active, the node continues to receive replicas of the query message resent by neighbors. It just extracts the query path list (P), and updates the visited BGL list with the node's address and the nodes's BGL address. When the timer goes off, the node checks to see if all its neighbors' BGLs and his own BGL are already listed. If they are not, then it resends the message to cover the missing BGLs. Otherwise, it drops the message.

Since BGLs do not delay the message and isolated nodes do, search path goes preferentially over BGL nodes. For cluster-tree borders defined by non-BGLs, the timer's jitter limits the number of retransmissions that occur on dense networks. The faster non-BGL on an area transmits the query to the destination BGL (or non-BGL for BGLs separated by two non-BGLs), which retransmits it. The BGL is added to the visited BGL list of other non-BGLs on the same area suspending their transmissions.

The algorithm improves SBA (Peng, 00) and ABC-QS (Choi, 02): It reduces the searching delay while crossing a connected set of stable nodes because BGLs never delay a query message; it reduces the message size (the number of BGL is lower than the number of nodes); it bases search paths preferentially over stable nodes, less likely to disappear; and it degrades more gracefully in the presence of transmission errors. It handles transmission errors similarly to SBA and ABC-QS: nodes keep sending a query message as long as a BGL does not appear on the path. Therefore, it only fails to reduce the load if none of the neighbor members of a BGL cluster retransmit the query message. This behavior improves the algorithm effectiveness for high network loads (due to the higher collision rate) and for high mobility conditions.

The algorithm does not guarantee total coverage on unstable networks, because it does not take into account unstable nodes in the neighborhood that did not yet transmit a beacon.

B. 2.5-hop searching algorithm

A second searching algorithm was developed as an extension of the 2.5-hop algorithm proposed in (Wu, 03).

A clustering algorithm modification is needed to support 2.5-hop searching algorithms: the neighbor's BGL list is added to the beacon message. The original 2.5-hop clustering algorithms (Wu, 03) sent the entire list of neighbors on the beacon producing more overhead.

On this algorithm, a node has information about all BGLs and isolated nodes within 2-hop distance. In order to reduce bandwidth usage, each sending node puts in the query message the list of non-BGL nodes at 1-hop distance (v) that must resend the message. The message is sent by the query starting node; by each BGL visited and isolated nodes; and by the non-BGL nodes that are in list v. List v is constructed from the set of 1 hop neighbors, and includes the non-BGLs required to cover all 2-hop distance BGLs. The algorithm: 1) first adds the neighbor nodes with unique paths to a BGL; 2) then, adds the neighbors that cover the maximum number of BGLs not yet in the list. A minimum node identification criterion was used to select from nodes with similar number of BGLs accessible.

The algorithm is more sensible to errors in the clustering information than the 1.5-hop version, since it uses topology information received one beacon period ago to select on-demand the next hop for the query message flooding. It also has less redundancy to tolerate transmission and topology errors, because it floods queries on a minimum CDS.

5 SIMULATIONS

The proposed algorithms and the source routing algorithm were implemented on version 2.27 of ns-2 platform (ns-2). The presented simulations compare the algorithms performance, using the same query generation and node movement patterns. In each simulation scenario 200 nodes are moving during 1000 seconds on a 1000m x 1000m area according to the Generalized Random Waypoint mobility model. Five different mobility scenarios were defined to study the mobility behavior of each flooding technique. Node’s average speeds of 0m/s, 1 m/s, 10m/s, 30 m/s and 40m/s were obtained using constant pause times of 1000, 150, 10, 9 and 5 seconds, respectively. Each node has approximately 100 meters of communication range using IEEE 802.11b over the two-ray ground propagation model. The beaconing frequency of each node is 1 Hz. The clustering algorithm parameters `transient_threshold` and `stability_threshold` are one and five seconds, respectively.

Ten thousands of different resources are randomly distributed on the network nodes. Three different behavior patterns were defined using the model presented in (Ge, 03). High, medium and low network load correspond, to 10927, 1125 and 267 generated queries, respectively. Finally, all broadcasts are sent with a jitter value of 100 ms, and the 1.5-hop algorithm uses a delay of 700 ms for non-BGLs and isolated nodes.

Figure 3 presents experimental results for the average BGL selection time for the fifteen combinations of speed and load, and for 1.5-hops and 2.5-hops neighborhood information. The selection time values not shown on the graph, for low and medium load and mobility zero were respectively 195 and 78 seconds for 1.5-hop and 118 and 61 seconds for 2.5-hop algorithm. These results show that the BGL selection time is negatively influenced by the beacon size and the load, but the average speed is the dominant parameter.

For zero mobility, all BGL changes resulted from having two successive beacon losses, producing a significant churn on the BG composition for heavy loads. Beacons are sent using multicast, and these results show how sensible multicast traffic is to collisions. The clustering algorithm stability could be improved for low mobility scenarios by tolerating more beacons losses. However, the algorithm's performance would degrade significantly for high average speed values. Notice that the algorithm also degrades for the 2.5-hops algorithm, due to the largest beacon length.

Node movement introduces extra BGL re-selections due to topology changes, which become the dominant factor for the two highest speeds. For node average speeds of 30 and 40 m/s the BGL persistent time converges for the minimum possible value (5 minus the selection tolerance of one). The percentage of nodes without a BGL also increased significantly, which means that on these scenarios the clustering is almost turned off.

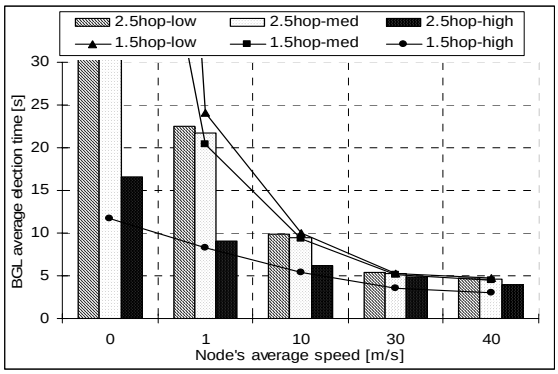

Figure 3: Average BGL selection time *versus* node average speed for 1.5-hop and 2.5-hop algorithm.

Figure 4 presents the percentage of successful queries for two extreme mobility scenarios of 1 m/s and 40 m/s, where 1.5-hop, 2.5-hop and source routing algorithms are compared using the medium load. It shows that the 1.5-hop searching algorithm outperforms the other two algorithms on both scenarios. It also shows that the pure source routing algorithm performance is poor for both scenarios. The main factor that penalizes source routing algorithm is the dependence on a single reverse path to route the hit packet. Source routing performance for 1 m/s is conditioned by the higher number of nodes disseminating query messages and the longest query message (it carries the complete path), which lead to more packet collisions, destroying query messages and hit messages. For the highest speed, the success probability drops to 2% is result of a high probability of return path failure.

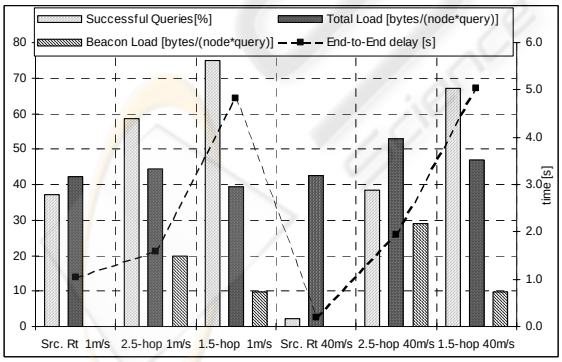

Figure 4: % Successful queries, load and end-to-end delay *versus* algorithm and node speed for medium load.

2.5-algorithm has a higher beacon overhead, which penalizes the bandwidth load. It is also sensible to packet loss during the query message dissemination due to using a minimum CDS. A node movement or a packet loss may produce a query coverage shedding, reducing the successes rate for higher speeds.

The 1.5-hop algorithm has the lowest load levels and is more tolerant to network changes, presenting a low degradation on the successful query rate. On the other hand, it increases the end-to-end search delay. Notice that due to the clustering reduced efficiency for high mobility, the 1.5-hop algorithm load increases, tending for SBA model for extreme mobility scenarios. This characteristic limits the network scale and to the network load supported by the algorithm for very high node average speeds.

6 CONCLUSIONS

The results presented in this paper show that the proposed 1.5-hop searching protocol has a strong resilience to network load and node movement, constituting a good choice for extreme mobility scenarios with low load levels. Its adaptability results from an adaptive clustering protocol, based on link stability, which adapts the controls the clustering granularity based on the network conditions. It reduces the cluster size and duration for extreme mobility scenarios increasing searching redundancy; it reduces redundancy for low mobility nodes, reducing the searching overhead.

The obtained results show that for high mobility scenarios, performance improves for the algorithms that use the least possible network information (1.5-hop). It is concluded that source routing approach fails for high mobility scenarios. Since most MANET routing protocols are based on source routing, this can present an important problem for common applications, not prepared to handle this kind of instability.

This paper presents on-going work. Further study is being made on beacon overhead reduction and beacon self-stabilization algorithms, which reduce beacon collision effects.

REFERENCES

Bernardo, L., Pinto, P., 2004. A Scalable Location Service with Fast Update Responses. In Proc. ICETE'04, Vol.1, pp.39-47.

Choi, Y., and Park, D., 2002. Associativity based clustering and query stride for on-demand routing protocol. Journal of Communications and Networks 4(1), pp. 4-13, Korean Institute of Communications Sciences (KICS).

JXTA Project, 2004. JXTA v2.0 Protocols Specification. Retrieved September 2004 from http://spec.jxta.org/nonav/ v1.0/docbook/JXTAProtocols.html

Ge, Z., Figueiredo, D., Jaiswal, S., Kurose, J., Towsley, B., 2003. Modeling Peer-Peer File Systems. In IEEE INFOCOM'2003.

Jacquet, P., Mülethaler, P., Clausen, T., Laouitit, A., Qayyum, A., Viennot, L., 2001. Optimized Link State Routing Protocol for Ad Hoc Networks. In IEEE INMIC'01, pp. 63-68. IEEE Press.

Klingber, T., Manfredi, R., 2002. RFC Draft of Gnutella v0.6. Retreived from http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

McDonald. A. B., Znati, T.F., 1999. A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks. IEEE Journal on Selected Areas in Communications 17(8) pp. 1466-1487

Network simulator - ns-2. Retrieved from http://www.isi.edu/nsnam/ns/

Peng, W., Lu, X., 2000. On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks. In MobiHoc'00, pp. 129-130, ACM Press.

Toh, C.-K., 1997. Associativity-Based Routing for Ad-hoc Mobile Networks. Journal of Wireless Personal Communications, vol. 4, pp. 103-139, Kluwer Academic Publishers.

Tseng, Y.-C., Ni, S.-Y. , Chen, Y.-S., Sheu, J.-P., 2002. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In Wireless Networks, vol. 8, nos. 2/3, pp. 153-167, Mar.-May 2002. Kluwer Academic Publishers B.V.

Tsumochi, J., Masayama, K., Uehara, H., Yokoyama, M., 2003. Impact of Mobility Metric on Routing Protocols for Mobile Ad Hoc Networks, In IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing (PACRIM03), pp.322-325. IEEE Press.

Wu, J., and Lou, W., 2003. Forward-node-set-based broadcast in clustered mobile ad hoc networks. Wireless Communications and Mobile Computing, N.3, pp.155-173, John Wiley & Sons, Ltd.