SEARCHING FOR RESOURCES IN MANETS:
Improving flooding efficiency in 802.11

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

Abstract: Searching algorithm's performance for mobile ad hoc networks depends strongly on the routing and MAC protocol characteristics. 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. A clustering algorithm is required for densely populated networks in order to avoid the broadcast storm problem. This paper proposes two new clustering and searching algorithms using 1-hop and 2.5-hop neighborhood information. It presents a set of simulation results on the clustering efficiency and on searching efficiency for low movement and for high movement, showing that the algorithm performance is acceptable for both scenarios.

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 indoor. On these conditions, standard proactive, table-driven, routing protocols (e.g. DSDV (Bhagwat, 94)) have worst performance than on-demand routing protocols (e.g. AODV (Perkins, 99), DSR (Johnson, 96), ABR (Toh, 97), etc.), which flood the network looking for an address only when it is needed. The problem is that routing information becomes outdated too fast, especially for lengthy paths. Due to bandwidth restrictions, it is not feasible to maintain the tables always updated - it is preferable to updated only when needed, possibly flooding the entire network with route lookup packets at that time.

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 directory (e.g. LDAP (Howes, 98)) and peer-to-peer (P2P) services (e.g. Pastry (Rowstron, 01), Tapestry (Hildrun, 02), Gnutella (Clip2, 02), FastTrack (FastTrack, 01)) 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 in not similar to the network physical topology, due to the routing protocol overhead. Crossing several virtual links may imply flooding several times the physical network after topology changes. An obvious alternative is to short-circuit the MANET routing protocol and to flood the searching protocol's query message directly using multicast or broadcast (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). An additional feature required for replicated resources is that searching must return the nearest replica. It is extremely important to reduce the number of hops on a communication, and when possible, to preload the MANET routing tables during the resource search phase.

This paper presents a new searching algorithm for MANETs, which has an interesting trade-off between reliability and efficiency, and that locates the nearest replica for replicated resources. It is optimized for very dynamic continuous MANETs. It does not handle the extreme case of delay tolerant applications on MANETs scattered on several islands, where a pure packet store-and-forward approach is required (Zhao, 04). Section two presents an overview of existing searching services and related work. Section three analyses the main trade-offs when designing a searching service. Sections four and five present the proposed clustering and searching protocols. Section six present the ns-2 simulation setup, and an evaluation of the proposed protocols using several simulation results. Finally, section seven draws some conclusions and presents future work directions.

2 SEARCHING SERVICES

Searching services may be classified in two big classes, regarding on how searching is performed: it either uses guided search to a specific node, or a flooding approach. The first approach is used for structured P2P services (e.g. Pastry, Tapestry, etc.)

and for directory services. Index information is stored on a subset of the nodes, where each node is responsible for a subset of the index database. The protocol also creates routing information that can be use to route search queries to the responsible node. This approach has low performance for ad hoc networks because: a) the maintenance of the routing tables is costly on unstable networks (Bernardo, 04); b) it is inefficient to concentrate index information on a potentially unreliable node. Although, some of the structured P2P network organization features were used by other authors on MANETs. DPSR (Charlie, 03) trades off memory usage by increased probability of route path failure: nodes do not store in cache the complete path to all destinations, but only a subset of the paths, accordingly to the Pastry information structure. KELOP (Bashir, 03) uses the routing layer caching information and loose coherence caching mechanisms to improve the structured P2P performance on MANETs. However, these two mechanisms fail for very dynamic networks.

A flooding approach is more adapted to MANETs due to the null registration costs. All efforts are concentrated during the search phase. Flooding can be done over an a priori defined overlay network (e.g. Gnutella), or directly over the network layer, without using the address routing protocol (e.g. JXTA rendezvous protocol). On this later case, the overlay network is created on demand: the query message includes the path traveled; when the resource is found, the hit message follows the path stored in the query message. This approach assumes that networks do not change during a search. However, this hypothesis may fail for high mobility scenarios. Additionally, this simple approach may also fail for dense MANETs, because of a high number of concurrent accesses to the shared medium, similar to a storm. In order to handle the broadcast storm (Tseng, 02) only a subset of the nodes must retransmit the search query message. This set of nodes is usually called a connected dominant set (CDS), and their broadcast region must cover the entire set of nodes. The remaining nodes should have a passive role, generating hit packets only. A new algorithm is thus introduced – a clustering algorithm that group nodes, and using a set of heuristics elects broadcast group leaders (BGLs), responsible for forwarding packets for their broadcast group (BG). In order to collect information about the neighborhood, most approaches require that each node periodically broadcasts a beacon packet. (Wu, 03) presents a taxonomy for clustering based flooding algorithms, where it classifies the algorithms accordingly to the state information used. Clearly, due to the inherent network instability, it is not possible to use global state information, nor any algorithm that requires a significant subset of global state information to create a CDS. The information state should be local, or quasi-local, using information about the node's neighbors (1-hop), or at most, its neighbor's neighbors (2-hop or 2.5-hop) to reduce the number of active nodes. These approaches differ on what information is included in the beacon: 1-hop sends only local information; 2-hop includes information about its neighbors; and 2.5-hop includes information about its neighbors BGL, which may be 3 hops distant. (Dai, 04) presents a thorough study for algorithms in low diameter extremely dense networks (it analyses 160 nodes on a network diameter of three hops). It concludes that algorithms using more neighbor information perform better. However, it fails to analyze what happens when the network diameter grows, and the routing redundancy importance grows. (Kozat, 03) proposes a flooding search approach where a 2.5-hop algorithm is used to define the CDS. However, it tries to create a minimum spanning tree using PRUNE packets, possibly introducing high signaling overhead for unstable network. Every modification on the network may change the spanning tree configuration.

3 TRADE-OFFS

Designing algorithms for potentially unstable mobile networks is completely different from designing them for static networks with fail silent nodes (e.g. wired networks). It is not advisable to base search on a minimum CDS (MCDS), which connects a minimum spanning tree, because it can be too costly to maintain updated the MCDS. Beaconing adds to the searching cost a constant cost (K) in bytes per second per node, proportional to the beacon frequency. It is only worthwhile to use beaconing when the result of the sum is below the cost without using clustering (L). If α is the fraction of nodes which are BGL, and N is the expected number of active queries in parallel, then the beaconing cost must always satisfy equation 1.

$$K < N.L.(1 - \alpha) \qquad (1)$$

Therefore, equation 1 limits the maximum beacon frequency. This implies that: a) it is possible that some old neighbor moved after sending their last beacon; b) it is possible to have black nodes on the neighborhood, which have not sent any beacon yet. Therefore, the CDS must have moderate redundancy, allowing it to cope with inaccurate neighborhood information and with search packet loss. A 1-hop strategy is more redundant (has a higher α) because it can only decide to forward packet based on the visited nodes and on the neighbors. A 2.5-hop strategy can select which neighbors resend the packet on the 2-hop range. However, it is more sensible to outdated information. The relative performance of 1-hop and 2.5-hop algorithms depends on the network instability level.

A second trade-off relates to the use or not of unicast packets. Clearly, multicast packets have several advantages over unicast packets: they send packets to unknown neighbors; they lower the network load when sending packets to several nodes within radio range. They also have a disadvantage: they are more unreliable since no confirmation is used for multicast packets on DCF (distributed coordination function) mode of 802.11 MAC. Unicast uses MACAW protocol, which uses a RTS/CTS exchange to reserve a time slot for sending packets (avoiding the hidden node interference problem that exists on multicast), It retransmits packets in case of failure. Unicast can be used while sending hit packets, or while flooding packets over the network defined by the clustering algorithm. However, beacons are always sent using multicast, and since their transmission is unreliable, then the clustering information is also potentially unreliable. This problem may affect clustering efficiency on a network near broadcast storm situation, where a significant percentage of beacons may be lost. This is a second reason not to increase the beacon frequency indefinitely, and to use preferentially multicast packets to flood the network.

A final trade-off relates to how clustering is made. Classical clustering approaches (Dai, 04) group nodes using the node identification or node degree (the number of neighbors) to define groups and select the BGL. On this paper we propose another approach inspired on ABR (Toh, 97): to base clustering decisions on beacon stability. The rational for this approach is that: there is a strong probability that a link remains stable during the next interval if the link nodes stay within radio range during a minimum interval of time. This clustering approach detects these groups of nodes moving coherently, and form stable clusters with them. Unstable nodes on the neighborhood are treated as second class members, which may not lead the cluster. For instance, a group of executives on a van form a MANET with only stable nodes, because they are all stable in relation to each other. However, they are moving in relation to nodes external to the van.

4 CLUSTERING ALGORITHM

The proposed clustering algorithm creates groups with a maximum diameter of 2 hops based on beacon stability and on the node identification, as a second criterion. It can be classified as 1-hop clustering, since it is uniquely based in a beaconing scheme. Each node periodically sends a beacon message. All nodes that receive a beacon from a determined node are defined as neighbor nodes. In figure 1, dashed lines represent links with neighbor nodes.

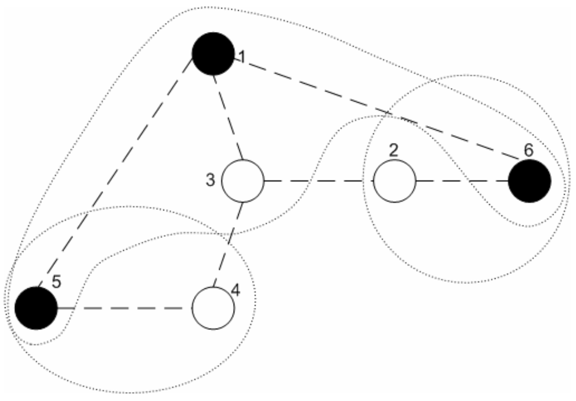

Fig. 1. Illustration of a MANET with 3 BGs. Nodes 1, 5 and 6 are BGLs.

Nodes are grouped according to their link stability η. Link stability is defined as the sum of consecutive beacons received from a determined neighbor. If more than one beacon is lost, then link stability is set to null. High stability values represent low nodes mobility and vice-versa. All nodes have a beacon table that is used for BGL election. For a 1-hop searching algorithm, in each beacon message a node sends its node identification, its BGL node, and the higher link stability value contained in its beacon table, which is represented by μ.

Beacon messages are broadcasted and when a new beacon is received, the receiving node identifies its source neighbor. If the receiving node beacon table contains an entry associated with that source neighbor, then it increments η value and copies BGL and μ contained in the beacon message to the table entry. If not it creates a new entry with BGL and μ contained in the beacon message and η value equals to 1. 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 can elect itself as its BGL, as is observed with neighbor 1. The algorithm converges frequently to a set of connected BGLs, as represented in figure 1, or where BGL are separated by one non-BGL when networks have higher connectivity. Node's 5 BGL is node 1, but node 5 is also a BGL elected by node 4. The subset of BGL plus the non-BGL nodes connecting BGLs define a backbone, which can be used to broadcast packets on a stable set of nodes.

Neighb.	**Stability - η**	**BGL**	**Neighb. Stability - μ**
1	43	1	43
2	8	6	64
4	2	5	33

Table 1. Beacon table of node 3 on figure 1

A node is stable if there is at least one η value contained in its beacon table that is higher than a defined `stability_threshold`. BGL election algorithm is performed on each node after a new beacon reception. The election algorithm is summarized in figure 2. In line 1, it computes the maximum values of η and μ contained in the beacon table. An unstable node cannot be elected as BGL but it can elect its BGL, since it has one or more stable nodes in its neighborhood. In this case it is elected the neighbor with highest μ value (algorithm lines 25 to 30). If it does not have stable neighbors, the BGL field is left empty. For a stable node, first it is computed 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. Having some elected BGL in the neighborhood, the key idea is choose a stable neighbor node that has the lowest address and is also a BGL (lines 11 to 15), or choose the node itself if there are no lower BGL node' address in its neighborhood (lines 16 to 18). If there are no BGLs elected in its neighborhood, a node simply elects its neighbor with the highest η value as its BGL (lines 20 to 24). If there is more than one neighbor pursuing the maximum η value then it is elected the node with lowest address.

Cluster overlapping can occur in result of sticking together several 1-hop radius clusters into a wider cluster. Usually the BGL is also a member of another group when BG has a single neighbor BG, as presented in figure 1. During system startup, a transitory overlap may also appear, because the initial criteria for selecting BGL is link stability (lines 21 to 23), which possibly is different from node to node. However, on this last case, when several BGLs exist for a set of stable nodes within radio range, 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 and converge to the same BGL. A node that receives *n* new beacons must receive all of them during the beaconing period, but they can be delivered with different time drifts (jitter). This instability does not affect the initial BGL election, because `transient_threshold` was set to one. Therefore, BGL is elected from a set of nodes that contains not only the neighbors with the higher stability value, but also all neighbors that could get that stability value during the present beaconing period.

```
1.  ($\eta_{max}$, $\mu_{max}$)=find_maximum_$\eta$_and_$\mu$_values_in_
                                   neighborhood_table()
2.  last_addr = MAX_INT
3.  pre_elected = -1
4.  if is_stable($n_a$)    // stable node
5.    //insert all known BGL's neighbor nodes
      //in n 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_elected = $n_x$
15.     if (pre_elected $\neq$-1) break;
16.     if ($n_a$=$bgl_x$) // auto-election
17.       pre_elected = $n_a$
18.       break
19.   // elect new BGL
20.   if (pre_elected =-1)//BGL is not selected
21.     for each neighborhood_node $n_x$
22.       if ($\eta_{max}$-$\eta$($n_x$)-transient_threshold $\leq$ 0)
                          $\wedge$ (addr($n_x$)<last_addr)
23.         last_addr = addr($n_x$)
24.         pre_elected = $n_x$
25. else  // unstable node
26.   // elect BGL if it is available
27.   for each neighborhood_node $n_x$
28.     if ($\mu_{max}$-$\mu$($n_x$)-transient_threshold $\leq$ 0) $\wedge$
                            (addr($n_x$)<last_addr)
29.       last_addr = addr($n_x$)
30.       pre_elected = $n_x$
31. BGL_ELECTED = pre_elected
```

Fig. 2. Outline of BGL node election algorithm applied in node n_a.

If a node stops the beaconing transmission process, because of mobility or operational reasons, each one of its old neighbors will take two beaconing periods to detect its absence. During this time period incorrect BGL elections may occur. However, beacon loss can also result from message loss due to congestion, producing false alerts.

A second version of clustering algorithm was implemented to support 2.5-hop searching algorithms. The only modification was the addition of a neighbor's BGL list to the beacon message. Notice that conventional 2.5-hop clustering algorithms (Wu, 03) usually have must more overhead because they send the entire list of neighbors on the beacon.

This clustering algorithms performance depends on the network stability. Its beacon period should be selected accordingly to the nodes velocity, limited by the maximum supported rate. If a large percentage of the nodes are "stable", the algorithm is able to detect them, and reduce their load by creating clusters. If all nodes are unstable, beaconing only introduces extra overhead. Notice however that non-clustered searching algorithms are more immune to errors. If conventional criteria were used, the clustering algorithm would create highly unstable clusters, which would include passing-by moving nodes, and would root search packets based on this error prone information.

5 SEARCHING ALGORITHM

This section proposes a new searching algorithm based on the 1-hop clustering algorithm presented above, tuned for very unstable networks. It also presents an alternative 2.5-hop based searching algorithm (with tuning optimizations similar to the 1-hop version). Both algorithms will be compared on the next section.

The searching algorithm was developed as an evolution of the basic source routing flooding algorithm (SR): The lookup operation is started with a query message originated by a node, which carries a unique identification (Q_{id}), its source node (n_{source}) and the resource identification to locate (R_{id}). This message is successively resent by each node, as long as it has not been received before. 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. Bandwidth usage could be reduced if hit routing information is store in the nodes (Clip2, 02). However, this second approach is less reliable because all up flow path is lost in case of node movement or failure, whereas on the first, it can jump a few nodes on the path.

The clustering algorithm classifies nodes has BGL, or non-BGL. Eventually, the node can be isolated, if is outside any BG, and thus it did not elect a BGL. A node knows that it is a BGL when it receives a beacon selecting him. The proposed searching algorithm modifies SR in three ways:

a) BGL and isolated nodes always broadcast queries one time, while non-BGL nodes may inhibit its transmission, if they have the certainty that all its neighbors had already received the query;
b) The query path list size is reduced by containing only BGLs, except for the tail, which may have some non-BGLs;
c) Hit messages use flooding when the reserve path fails.

In order to reduce the bandwidth usage, non-BGL nodes use the neighborhood information collected with the clustering algorithm, and the query path list, in the query message (a). Two approaches were proposed: using 1-hop or 2.5-hop information.

Two modifications are common to the two approaches above. The query path list shortening (b) results from the addition of node storage of parts of the total query path. The non-BGL node list at the tail of the list is stored and pruned, each time the message passes on a BGL. BGLs maintain in its internal lists, the partial path with non-BGL from the previous BGL, which may include several unstable nodes. Since this information can be volatile, it can be stored on a less secure place. In case of node failure, the node can always use the BGL list (inside the query message) to recover the route to the query source. When hit messages follow the query reserve path unicast is used and their sending is confirmed. They retrieve the unstable list at each BGL, and follow the reverse path. 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 (c). 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 algorithms are able to survive to extreme mobility, and route hit messages over failed or moving nodes.

A. 1-hop searching algorithm

The 1-hop searching algorithm proposed results from an improvement of SBA (Peng, 00), which was initially designed without clustering. Using the 1-hop clustering algorithm, a node has information about its neighbors' BGL. When a non-BGL node receives a new query, it first queues the message in a local buffer, creates a local variable with the list of visited BGL, and starts a timer, for a fixed delay plus a jitter interval. When the timer triggers, the node checks to see if all neighbors' BGL and local BGL are already listed in the visited BGL list. If they are not, then it resends the message, to cover the missing BGLs. Otherwise, it drops the message.

While the timer is active, it continues to receive replicas of search messages resent by neighbors, just for extracting the query path list, and to update the visited BGL list. Since BGLs do not delay the message and isolated nodes do, search path goes preferentially over the BGL backbone. Due to the timer's jitter, this approach limits the number of retransmissions that occur on dense networks. The first non-BGL transmission triggers the immediate transmission of the missing BGL, shutting off all the remaining transmissions.

This first algorithm does not guarantee total coverage on unstable networks, because it does not guarantee the coverage of black nodes (nodes in the neighborhood that did not yet transmit a beacon). It handles transmission errors similarly to SR, nodes keep sending a query message as long as a BGL does not appear on the path. This behavior may originate load peaks after a node failure or movement. Finally, it delays the search, compared to the other algorithms analyzed on this paper.

B. 2.5-hop searching algorithm

For the 2.5-hop clustering 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. All BGL nodes always resend the message. The message is sent by the query starting node; by each BGL 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.

This second algorithm is more sensible to errors in the clustering information, since it uses topology information received one beacon period ago to select on-demand the next hop for the query message flooding. Since it reduces the flooding to a MCDS, it also has less redundancy to tolerate transmission and topology errors, compared to the other algorithms.

6 SIMULATIONS

To study the clustering algorithm performance for different node densities, seven different simulation scenarios were defined. In each scenario nodes are moving during 10 simulation hours in a 1000m x 1000m area, using an improved Random Waypoint mobility model (Yoon, 03). Mobility model parameters are 0.1 m/s and 3 m/s for minimum and maximum velocity, respectively and 3600 seconds for pause time. Each node has approximately 100 meters of communication range, and its beaconing frequency is 1 Hz. Using beacon reception information, it was computed the average number of nodes in the neighborhood, to compare with the average number of nodes that elect the same BGL. First simulation uses 75 nodes, and it was the lowest node density value simulated. The average number of neighbors is around 2 nodes, which states that the optimum value for average BG dimension should be around 3 nodes, counting 2 neighbors and the node that detects the neighborhood. Simulation results presented in figure 3 show that average dimension of BG is approximately 3 nodes, which is closed to the optimum value. In the following simulations, nodes' densities were increased. Results show that average number of neighbors will increase, as expected, but the average of BG dimension will increase more slowly because the standard deviation will be higher. This is mainly because of BGs merging operation. The average of nodes without election of was also computed and is presented in table 2. Simulations with lower node densities present higher average of nodes without election of BGL, as expected. In high-density scenarios, less than 1% of all nodes don't belong to any BG. Other important feature is that average BGL election period will decrease for higher density values, because the BGs dimension increases which also increases the probability a node leave the BG resulting in a new BGL election.

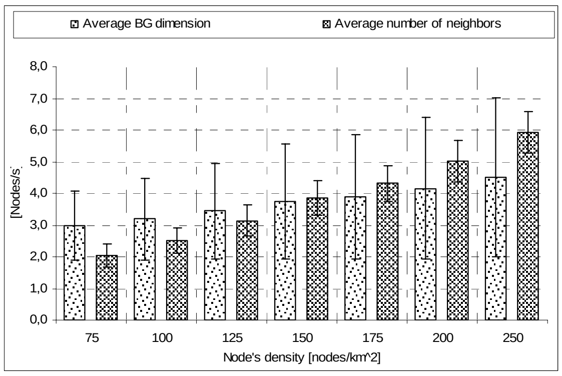

Fig. 1. Average broadcast group dimension *versus* average number of neighbors for 7 simulations. Lines presented over the bars represent standard deviation.

The clustering algorithm performance for different node mobility scenarios was also tested using five different average node speeds for the network above with 250 nodes, modeled using the

Random Waypoint mobility model. Figure 4 shows that the clustering algorithm performance degrades with the speed increase. It affects mainly the BGL average election time that is reduced to about 5 seconds for an average speed of 40 m/s. The average BG dimension stabilizes on an average size of three members, for speeds above 20 m/s, proving that some aggregation is still possible due to the node's pause times.

Density [nodes/km^2]	75	100	125	150	175	200	250
Without BGL [%nodes/sec]	12.1	7.8	4.0	2.2	1.2	0.9	0.2

Table 2. Average number of nodes without BGL, and average BGL election period statistics.

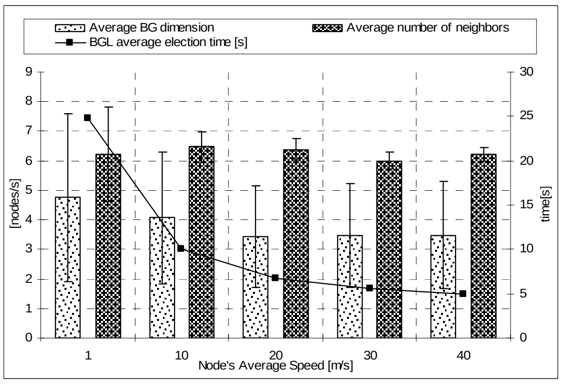

Fig. 4. Average broadcast group dimension and duration *versus* node average speed for 5 simulations. Lines presented over the bars represent standard deviation.

The three proposed clustering algorithms were analysed for three extreme scenarios: low speed MANETs (average speed of 1 m/s) and high speed MANETs (average speed of 40 m/s), on the conditions presented above. Figure 5 shows that the two improvements proposed for the two clustering approaches perform better than source routing for both high and low mobility. The results are disastrous for source routing for high mobility scenarios because the answer message is lost when a node on the path moves. Results for 2.5-hop clustering information are also worst than 1-hop for high mobility because clustering information is less precise, and 2-5-hop approach uses this information to control flooding. On the other hand, 1-hop introduces a higher packet delay. For low mobility scenarios, 2.5-hop clustering approach provides the best performance, because it uses the lower number of packets to flood the network, based of reliable clustering information. Answer messages loss relate to collisions, which are compensated by the flooding approach for the clustering approaches.

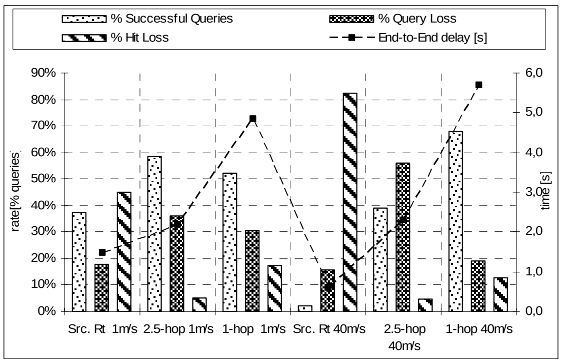

Fig. 5. Success rate and end-to-end delay for source routing; 1-hop and 2.5-hop clustering algorithms for 2 node average speeds.

Figure 6 presents the total bandwidth load for the conditions above. It shows that flooding based on the clustering algorithm is capable of reducing the total bandwidth usage on the network, for low node mobility. When node mobility is high, clustering approaches use flooding to compensate return path loss, increasing the total load, and a higher number of collisions.

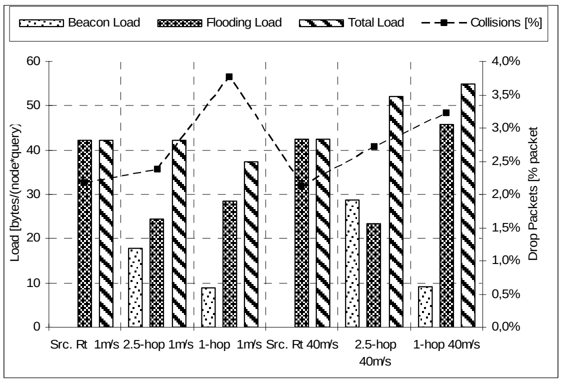

Fig. 6. Bandwidth load and collisions for source routing; 1-hop and 2.5-hop clustering algorithms for 2 node average speeds.

7 CONCLUSIONS

This report proposes two new approaches for searching on unstable MANET, supported on positioning information provided by a clustering algorithm. It shows that for high mobility scenarios, performance improves for the algorithms the use the least possible network information (1-hop). It also shows 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 report presents on-going work. Further study is being made on beacon overhead reduction and beacon self-stabilization algorithms, which reduce beacon collision effects. Further work is also being done for searching algorithms for less unstable networks, where a DHT approach may be used on top of a stable virtual overlay network.

REFERENCES

Bashir, S., Li, B., 2003. KELOP: Distributed Key-Value Lookup in Wireless Ad Hoc Networks. In 12th IEEE International Conference on Computer Communications and Networks.

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

Bhagwat, P., 1994. Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers. In ACM SIGCOMM'94, pp. 234-244, ACM Press.

Charlie, Y., Das., S., and Pucha, H., 2003. Exploiting the synergy between Peer-to-Peer and Mobile Ad Hoc Networks. In Usenix HotOS-IX. Retrieved from http://www.usenix.org/events/hotos03/tech/hu.html

Clip2. The Gnutella Protocol Specification v0.4 Rev. 2.3. Retrieved October 11, 2002 from http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

Dai, F., Wu, J., 2004. Performance Analysis of Broadcast Protocol in Ad Hoc Networks Based on Self-Pruning. In. IEEE WCNC'2004, pp. 802-807, IEEE Press.

FastTrack,. Peer-to-peer technology company. Retrieved 2001 from http://www.fasttrack.nu.

Hildrun, K., Kubiatowicz, J. D., Rao, S., Zhao, B. Y., 2002. Distributed Object Location in a Dynamic Network. In SPAA*'02, 14th annual ACM symposium on Parallel algorithms and architectures*, ACM Press.

Howes, T., Smith, M. C., Good, G. S., Howes, T. A., Smith, M., 1998. Understanding and Deploying LDAP Directory Services. Macmillan Technical Pub.

Johnson, D., Maltz, D., 1996. Mobile Computing. Kluwer academic publishers.

Kozat, U. C., Tassiulas,L., 2003. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In IEEE INFOCOM 2003. IEEE Press.

The 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.

Perkins, C. Royer, E., 1999. Ad-Hoc On-Demand Distance Vector Routing. In 2nd IEEE Workshop on Mobile Computing Systems and Applications, IEEE Press.

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

Rowstron, A. I. T., Druschel, P., 2001. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Middleware'01, *18th IFIP/ACM Int. Conf. on Distributed Systems Platforms*, LNCS Vol. 2218, Springer 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.

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.

Zhao, W., Ammar, M., Zegura, E., 2004. A Message Ferrying Approach for Data Delivery in Sparce Mobile Ad Hoc Networks. In MobiHoc'04, ACM Press.

Yoon, J., Liu, M., Noble, B., 2003. Random Waypoint Considered Harmful. In: Infocom'03. IEEE Press.