Mobile Ad Hoc Backbones: Formation and Maintenance

Maurizio A. Nanni and Stefano Basagni

ECE Dept., Northeastern University, Boston, MA 02115, {mnanni, basagni}@ece.neu.edu

Abstract— **This paper presents a new protocol for the formation and maintenance of a backbone from the nodes of an ad hoc network where all the nodes are mobile. The new protocol is defined and its performance evaluated with respect to metrics such as its size, that favors scalability, its connectivity in time, which allows protocols to run efficiently, robustness and route length. Via ns2-based simulation experiments we demonstrate the effectiveness of proposed protocol for mobile ad hoc backbones in achieving a manageable size of the produced backbone, a connectivity in time which is maintained for the greatest majority of the simulation time, considerable robustness even at lower densities and route lengths that are negligibly bigger that the length of the routes in the flat network topology.**

Index Terms— **Ad hoc networks, hierarchical organization, wireless mobile backbones.**

I. Introduction

Among the problems that have been investigated in the wide realm of mobile ad hoc networks, one that is still largely unresolved is that of providing the mobile, infrastructureless network with a suitable organization that would make communication protocol scalable. Solutions that attempted to solve the problem have explored the direction of providing the network with a hierarchical organization, such as partitioning the network into clusters and considering the clusters as "super nodes" interconnected to form a *backbone*, i.e., a virtual network with a smaller number of nodes and links, and therefore more manageable. While several protocols have been proposed for backbone formation over a flat ad hoc network topology, only a few of them, and quite recently, have started to show that backbone maintenance in the presence of mobility can be achieved so that actual communication protocols such as routing can be run over the backbone thus achieving the needed scalability. Among the most representative solutions of mobile backbone formation and maintenance we cite the works by Ju et al. [1] and by Srinivas et al. [2] (where the reader is referred to for reference on further previous work).

In this paper we contribute to the problem of providing a mobility-resilient protocol for backbone construction and maintenance that achieves the requirements of size, robustness and connectivity one would expect for running routing and other communication protocols on it reliably. Starting, as customary, from a core clustering protocol that quickly recover from mobility-caused link failures, we define rules for the cluster coordinator nodes (*clusterheads*) to react to backbone link failures quickly and effectively, without leaving connectivity interrupted but for a very short time. Our aim in this paper is that of providing the definition of the protocol, highlighting the simplicity of its rules that result in ease of implementation. More important, as demonstrated by our ns2-based simulation experiments, the backbone produced by our protocol is considerably smaller than the flat networks, is connected for over 94% of the time independently of the network size/density, is robust to link failures, and achieves routes that, especially in dense networks are negligibly longer that those in the flat network topology.

The rest of the paper is organized as follows. Section II describes our protocol in detail. Section III shows the results we obtained in our simulation-based investigation. Section IV concludes the paper.

II. Mobile Backbone Protocol

In this section we first briefly describe the core clustering protocol upon which we build and maintain a mobile connected backbone. We then describe in details the *Clusterhead-to-Clusterhead* (CH-to-CH) connection protocol that allows the set of CHs to select those nodes, the *gateways*, that provide backbone connectivity in the face of mobility.

A. Core clustering

A protocol to select clusterheads (CHs) and their cluster members (CMs) and maintain the obtained clustering while the nodes move was presented in [3] and called GDMAC, a generalized distributed and mobility adaptive clustering protocol. GDMAC is packet-driven, i.e., specific protocol operations are executed upon receiving a packet. More specifically, a *CH* packet is broadcast to let the neighboring nodes know that the sending node is going to be a clusterhead; a *JOIN* packet is employed to make neighboring nodes aware that the sending node has joined a particular cluster, and a *RESIGN* packet is broadcast when the number of clusterheads in the neighborhood is bigger than a certain parameter K, thus keeping the CHs well scattered through the network. (For more details the reader is referred to [3].) The original GDMAC protocol assumes that periodic HELLO packets are employed for making nodes aware of broken links (because of nodes that move away, nodal failures or environmental conditions) or newly formed ones (nodes that recently arrived in the neighborhood or become operational). We perfect that mechanism by "merging" the HELLO packets mechanism with the clustering packets, thus optimizing the clustering formation in terms of clustering time, of the time for recovering from mobility and other failures, and of overhead (number of bytes exchanged for clustering formation and maintenance). Thus, an HELLO packet sent by node v contains the following information:

- The ID of the node (say, v),
- its weight,
- v's clusterhead ID,
- v's clusterhead weight,
- the GDMAC node status (CH, CM or gateway), and
- a resign field as required by GDMAC operations.

These field are interpreted by the receiver nodes according to the semantics of GDMAC which is proven to form and maintain a clustering organization of the network nodes so that no more than K clusterheads are neighbors and where a CM always affiliates with the neighboring clusterhead with the bigger weight [3].

B. Backbone formation: The CH-to-CH protocol

Since GDMAC is a clustering protocol, all it does is selecting CHs and CMs and organizing them into disjoint clusters. In order to form a connected backbone we must ensure that all the clusterheads that are at most three hops away are connected [4]. Connection between clusterheads u and v is provided by a direct link (when the two clusterheads are neighbors), by a CM that is common neighbor of both u and v (although belonging only to the cluster of one of them) or by two neighboring CMs, one belonging to u's cluster and the other to v's. The CMs providing inter-CHs connections are called *gateways*. CHs and gateways are also called backbone nodes. Gateway selection is CH driven, i.e., the CHs decide which CMs should become gateways. To this purpose, CH, CM and gateway nodes exchange HELLO packets, and specifically:

- Each CM or gateway node includes in the HELLO packets the list of the CHs that are one and two hops away. In particular, for each CH that is two hops away each CM or gateway node includes, along with the CH ID, the weight of the node providing the best connection, i.e., the intermediate node with the highest weight to that CH.
- Each CH includes in the HELLO packets the list of its neighboring CM nodes that should become gateways.

By receiving the HELLO packets each CH gathers information on all CHs that are two and three hops away and is able to choose the CM node or the pair of CM nodes that will act as gateways. Each CM node will be a gateway if and only if it receives an HELLO packet from its CH that includes its own ID in the carried gateway list. The gateway selection is based on the intermediate node IDs. When two CHs are two hops away the chosen CM that becomes a gateway is the one with the highest weight. When two CHs are three hops away the chosen CMs that become gateways are the two in the path that contains the nodes with the the highest weight. For maintaining the backbone connected, when a CH u receives an HELLO packet from one of its neighbors v, it checks whether the CHs in the v's CH neighbor list are known, i.e., there is already a path to them. A CH-to-CH path that is already established can be changed if a new neighbor provides a better path between the two. For instance, 2-hop paths are always preferred to 3-hop ones. If instead there is a new CH w to find a path to, then u will pick the 2-hop or 3-hop CH-to-CH path through v. Fig. 1 shows the

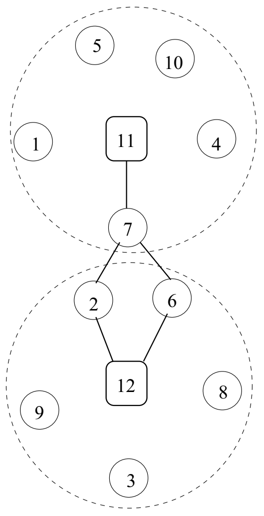

Fig. 1. CH-to-CH connection.

snapshot of a network region with 12 nodes that GDMAC has organized into two clusters, with 11 and 12 being the two clusterheads (here the node weight coincides with its ID). The two clusterheads are three hops away and they could be connected via two different paths, namely, $11 \rightarrow 7 \rightarrow 2 \rightarrow 12$ and $11 \rightarrow 7 \rightarrow 6 \rightarrow 12$. The CM nodes include in the HELLO packets the list of the CH that are one and two hops away. At first, they are only aware of the CHs that are one hop away. For instance, CM nodes 2 and 6 include the 12 in their CH list, while the CM node 7 has only CH 11 in its list. Once 7 has received HELLO packets from 2 and 6 it becomes aware that CH 12 is two hops away. Therefore, in the next HELLO packet it will include 12 as a clusterhead that can be reached through it in two hops. Since it has currently two ways to reach 12, via CM nodes 2 and 6, it will choose the latter for connection to 12. In the next HELLO packet to its clusterhead 11, node 7 will list CH 12 as a 2-hop away CH, together with the weight of the best connection to it (the weight of CM node 6). Similarly, when 2 and 6 receive the HELLO packet from 7, they are made aware of CH 11 as a two hops away CH, and will include this information in the list of their next HELLO packet. Let us assume now that 12 receives the HELLO packet from CM 2 first. Since 12 was not aware of CH 11, it choose the path through CM 2 to connect to it. When 12 receives the HELLO packet from the member 6, it already knows about CH 11. However, since the connection through 6 has a higher weight, it switches to the CH-to-CH path through 6. Thus, at the next HELLO packet, CH 12 includes CM 6 in the gateway list. When 6 recognizes its own ID in the gateway list of CH 12 HELLO packet, it becomes a gateway. When CH 11 receives an HELLO packet from 7, it establishes a 3-hop connection to CH 12 through CM 7 and includes the ID 7 in its next HELLO packet. Upon receiving it, the CM 7 becomes a gateway and the CH-to-CH link is set. CM node 2 ceases to be a gateway when, receiving an HELLO packet from CH 12, it is made aware that CM node 6 has been designated to be the new gateway to CH 11.

III. Simulation Results

In order to investigate the performance of our mobile ad hoc backbone protocol, we implemented it in the VINT Project simulator ns-2 [5] and its extension ns-MIRACLE [6]. Nodes are distributed uniformly and randomly on a deployment area that is a square of side $L =$ 1000m. In order to investigate the protocol performance with different nodal densities our experiment concerns networks with 100 to 500 nodes. Each node is equipped with a 802.11g card. In this set of experiments the weight of the nodes coincide with their unique ID. The GDMAC parameter K has been set to 3 (i.e., at most 3 clusterheads can be neighbors). The data rate has been set to 6Mb/s and the transmission radius is 250m. The HELLO packets period p has been set to 2 seconds. The mobility model we employed in our simulations is the Gauss-Markov model, with tuning parameter α, available with ns-MIRACLE. Totally random movement (or Brownian motion) is obtained by setting $\alpha = 0$ while linear motion is obtained by setting $\alpha = 1$. Intermediate levels of randomness are obtained by varying the value α between 0 and 1. In our simulations, we set $\alpha = 0.5$. The average nodal speed has been set to 10m/s.

Each simulation run lasts 1000 seconds. All nodes are started at random during the first 60s of simulation. After that, we start measuring the metrics of interest by taking a network screenshot every second of the remaining simulation time. The results shown in this section are the average over 100 different runs for each nodal density, which allows us to achieve 95% statistical confidence and 5% precision.

We have investigated the following four metrics: *A. Backbone size*, which counts the number of nodes in the backbone, i.e., CH and gateway nodes; *B. Backbone connectivity*, which indicates the fraction of time that the backbone was connected, i.e., able to provide a route between any two nodes in the network; *C. Backbone robustness*, indicating how many link failures it would take to disconnect the backbone, and the *D. Backbone route length*, i.e., the average number of hops of the shortest paths between any two nodes in the network passing through the backbone.

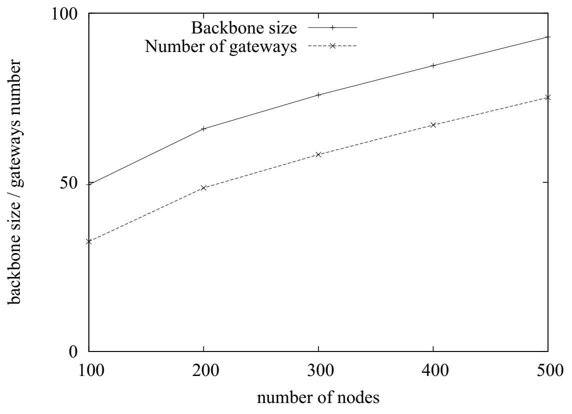

Fig. 2. Backbone size vs. number of gateways.

A. Backbone size

In Fig. 2 we show the average backbone size for different nodal densities, together with the average number of gateways. The backbone size ranges between 49 for networks with 100 nodes and to 92 for denser networks (500 nodes). We observe that the greater part of the backbone nodes is made up of gateway nodes (about 66% for 100 node networks up to about 80% for 500 node networks). Fig. 2 also shows that the number of CHs remains constant as the number of nodes increases (about 17 CHs on average). This suggests that the number of CHs might only depends on the size of the deployment area. On the other hand, the number of gateways increases as the number of nodes increases. This behavior can be explained by observing that in a sparse network, a member node has higher probability to be chosen as gateway by some CHs. Thus, in sparse networks, a member node serves as gateway for more CH-to-CH links. For denser networks, each gateway serves a lower number of CH-to-CH connections, which increases the number of gateways. Furthermore, in a denser network, there is an higher chance that a pair of CH neighbors has an intermediate node or a pair of intermediate nodes that can connect it.

B. Backbone connectivity

Fig. 3 shows the average time in which, given that the flat network topology is connected, the backbone is connected. As expected, the sparser the network, the higher the chance to disconnect the topology and the backbone. However, our protocol shows fast recover from

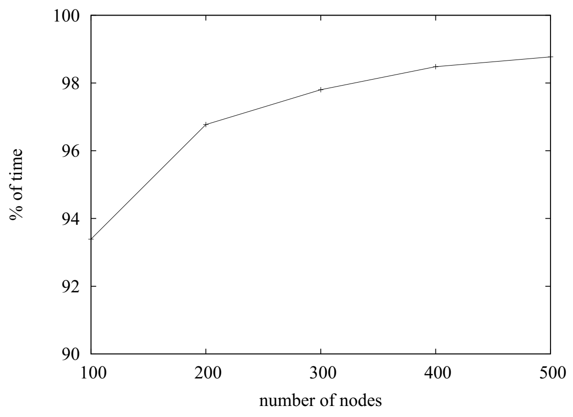

Fig. 3. Backbone connectivity.

disconnection. In particular, for network with 100 nodes the backbone has been connected, on average, for almost 94% of the time. This percentage increases to 99% for denser networks, i.e., the backbone is basically always able to provide a route between any two nodes in the network.

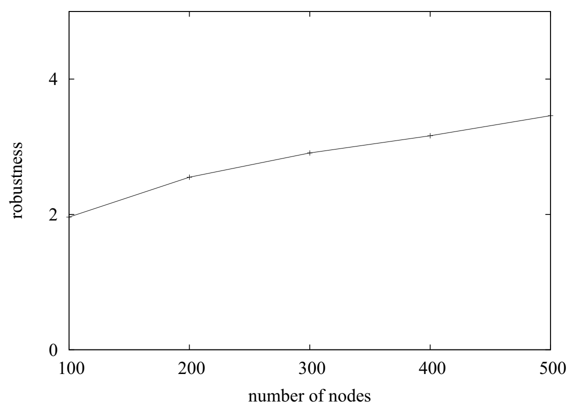

Fig. 4. Backbone robustness.

C. Backbone robustness

In Fig. 4 we show the average robustness of the backbone, i.e., the minimum number of links that needs to fail for disconnecting the backbone. We observe that our CH-to-CH protocol is effective in producing robust backbones. On average, in networks with 100 nodes the average number of links that should fail is 2, which means that each node in the network has at least two different paths through the backbone to any other node in the network. The average robustness is about 3.5 for denser networks (500 nodes).

D. Backbone route length

Building a clustering-based backbone equates giving the network a hierarchical organization. This is turn leads to an increase of the average route length (in number of hops) between any two nodes in the networks. We have observed that the average route length on the backbone goes from an average of 2.9 hops in sparser networks down to 2.7 hops for networks with 200 nodes and up (Fig. 5). This compares favorably with the average length of the routes in the flat network topology which, in denser network, is 2.5.

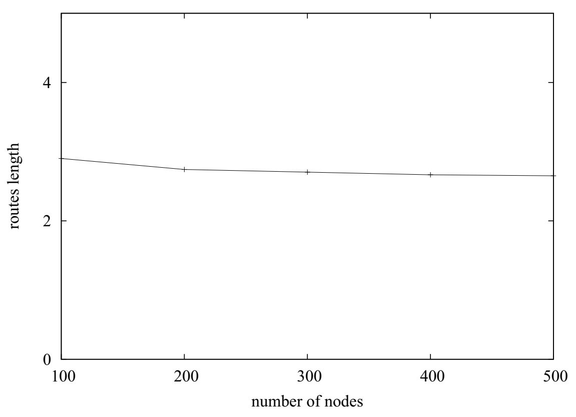

Fig. 5. Backbone routes length.

IV. Conclusions and Future Work

We presented a new distributed protocol for backbone formation and maintenance for wireless ad hoc networks designed for a mobile scenario. The protocol shows good performance on key metrics such as backbone size, backbone connectivity, robustness and routes length. In future works we will explore further metrics and will compare our protocol with previous solutions for mobile backbones.

Acknowledgment

This work has been supported in part by NSF Grants CCF 0634847 and 0634848.

References

[1] H. Ju and I. Rubin, "Mobile backbone synthesis for ad hoc wireless networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 12, pp. 4285–4298, December 2007.

[2] A. Srinivas, G. Zussman, and E. Modiano, "Construction and maintenance of wireless mobile backbone networks," *IEEE Transactions on Networking*, vol. 17, no. 1, pp. 239–252, February 2009.

[3] R. Ghosh and S. Basagni, "Mitigating the impact of node mobility on ad hoc clustering," *Wiley InterScience's Wireless Communications & Mobile Computing, WCMC, Special Issue on Resources and Mobility Management in Wireless Networks, L. Bononi and S. Nikoletseas, eds.*, vol. 8, no. 3, pp. 295–308, March 2008.

[4] I. Chlamtac and A. Faragó, "A new approach to the design and analysis of peer-to-peer mobile networks," *Wireless Networks*, vol. 5, no. 3, pp. 149–156, May 1999.

[5] The VINT Project, *The ns Manual*. http://www.isi.edu/nsnam/ns/, 2002.

[6] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, and M. Zorzi, "ns2-MIRACLE: A modular framework for multi-technology and cross-layer support in network simulator 2," in *ValueTools '07: Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, 2007, pp. 1–8.