M-Backs: Mobile Backbones for Multi-hop Wireless Networks

Maurizio A. Nanni
ECE Dept.
Northeastern University
Boston, MA, U.S.A.
Email: mnanni@ece.neu.edu

Stefano Basagni
ECE Dept.
Northeastern University
Boston, MA, U.S.A.
Email: basagni@ece.neu.edu

***Abstract*—We present a new distributed protocol for setting up and maintaining a backbone for multi-hop mobile networks. Our solution, termed M-Backs, is effective in producing backbones with limited size, while being robust and reliable in the face of node mobility. We compare our protocol with previous solutions for mobile backbone set up and maintenance. M-Backs outperforms previous protocols, providing backbones of reasonable size, with superior connectivity, shorter route lengths, higher robustness and lower maintenance-related overhead. When running AODV over the backbones M-Backs obtains higher packet delivery ratio and lower latency than previous backbone protocols and than AODV run over the flat network topology.**

I. Introduction

In the past two decades of investigation of multi-hop mobile wireless networks (also called ad hoc networks) we have witnessed several advances, especially in the field of routing and MAC protocol design, interdisciplinary applications (e.g., vehicular networks and robotics), and public safety networks. Ad hoc technology is now considered mature and many of its incarnations, such as wireless sensor networks and wireless mesh networks, are reaching the stage of actual deployment. However, despite the host of quality research on multi-hop mobile networks, the mobility of the nodes still represents a challenge that we have been coping with, but that we have not completely tamed when designing ad hoc protocols.

This paper aims at revisiting the problems of mobility and in particular on how to design protocols for constructing and maintaining hierarchical structures (*backbones*) over a flat network topology of mobile nodes. Our goal is that of leverage the experience matured in this field to determine simple procedures that imposing little overhead succeed in producing backbones that are robust and reliable in face of nodal mobility.

The problem of building backbones over multi-hop mobile wireless networks has been considered since the very beginning of the research on these networks [1], and successively revisited for routing [2] and multimedia [3] purposes. Backbone maintenance in the face of the mobility of nodes has remained quite a problem, however, and in order to maintain a connected backbone many protocols resort to re-execute the formation protocol either periodically or on-demand. Only recently, thanks to the new interest on mobile networking sparked by vehicular networks, robotics and mesh networking, mobility has returned center stage in the discussion on scalable protocols for mobile clustering and backbones. In particular, three protocols have been presented that take mobility into consideration so that backbone maintenance and formation are seamlessly obtained throughout common network operations. Ju and Rubin propose ETSA, an Extended Topology Synthesis Algorithm for mobile backbones in [4]. ETSA operations are based on the periodic exchange of HELLO packets, which are needed anyway for neighbor discovery, a key operation in presence of mobility. ETSA packets contain the node ID, a parameter (called the node *weight*) indicating how good the node is for serving as backbone node, the list of its neighbors in the backbone, and some other information serving both ETSA and neighbor discovery needs. ETSA builds rather sparse backbones, i.e., formed by a small fraction of the network nodes, in a distributed fashion. A clustering-based approach to mobile backbone formation is presented by Nanni and Basagni [5]. The backbone is built by connecting the clusterheads of the clusters formed by the GDMAC protocol [6], of which the backbone solution maintains the name. HELLO packets are used by GDMAC for conveying backbone control information, and problems typical of the GDMAC-like approach to clustering (such as long convergence time) are mitigated by introducing threshold-based backbone reorganization. GDMAC backbone are fairly large, which ensure robustness and impose less frequent reorganizations. In [7] Lee et al. propose *TRUNC-K* (simply TRUNC in the following), a backbone formation and maintenance protocol also based on periodical dissemination of control packets. Nodes perform a leader election scheme to elect a node as the root of a backbone fragment and then apply rules to join the various fragments into a connected backbone. Being leader election an intrinsically sequential process, and because HELLO packets carry the entire neighborhood list of a node, TRUNC encounters scalability problems. Despite the differences of these protocols, their design—and the design of other solutions found in the literature [8], [9], [10]—suggests common rules that should apply to mobile backbones.

- The protocol should not require node synchronization.
- Control information should be carried by HELLO packets.

- The HELLO packet size should not depend on the network size/density.
- The protocol should converge quickly to a connected backbone (if the flat network topology is connected).
- The protocol should choose backbone nodes based on the suitability of a node to be part of the backbone.
- The protocol should not require extra hardware/information such as geodetic coordinates.
- The protocol should not influence or require specific mobility patterns.

Not all solutions proposed so far for mobile backbones present these desirable characteristics. In this paper we propose *M-Backs*, a new protocol for mobile backbone formation and maintenance achieving all the desiderata listed above in a single solution. We have compared the performance of M-Backs with that of ETSA, GDMAC and TRUNC through ns-2 based simulations to evaluate the effect of different design choices on the protocols in scenarios with different node densities and speeds. We observed that M-Backs produces backbones with size intermediate between that of the large GDMAC backbones and that of the slimmer backbones of ETSA and TRUNC. In other words, M-Backs achieves backbones with a number of nodes that on one side is reasonably small, so that the backbone itself can be easily and quickly repaired when needed, and on the other it is large enough to obtain a robust structure that delivers packets reliably. As a consequence, and because of the simple rules implemented by M-Backs to maintain the backbone connected, our protocol achieves high backbone connectivity time and routes that are reasonably short, comparable to the routes of GDMAC backbones. When we use AODV [11] to route packets over the backbones, M-Backs outperforms all other protocols and AODV over the flat network topology, in terms of packet delivery ratio, latency and routing overhead.

The rest of the paper is organized as follows. In Section II we describe M-Backs in details. Section III presents a comparative performance evaluation of the solutions for mobile backbones considered. Section IV sums up the paper.

II. M-Backs: A Protocol for Mobile Backbones

The idea of M-Backs is that of selecting among the network nodes a fraction of them to serve as backbone nodes. Each node listens for, and collects data from, the HELLO packets broadcast by its current neighbors, and periodically decides whether to affiliate with a backbone node or to be one. Then the node broadcast its decisions to its neighbors, and so on. In M-Backs each node can assume one of the following roles: *Head Node* (HN), *Gateway Node* (GN), *Associated Node* (AN) and *Regular Node* (RN). HNs and GNs form the backbone; ANs are nodes that are associated to a neighboring HN (they could enter the backbone, if needed, in the sense that they have backbone capabilities); RNs are nodes with no backbone capabilities and always stay RN.[1] The choice of becoming a HN depends on how suitable a node u is to be on the backbone, is expressed by a real number $wt(u) \geq 0$ that node u can dynamically compute based on application requirements.

The format of the HELLO packet broadcast by node u is shown in Fig. 1. The HELLO packet carries the source

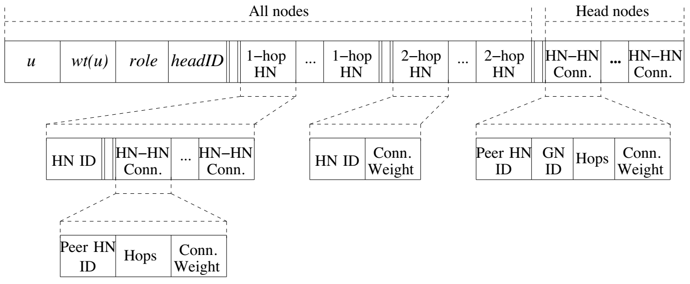

Fig. 1. HELLO packet format.

node ID u, its weight $wt(u)$, its role and the ID of the HN it is affiliated with (itself, if u is a HN). Each node u also includes the list of HNs that are one and two hops away along with the list of the 2-hop HN-HN connections established by each of these HNs. Finally, if u is an HN it includes the list of the HN-HN connections that should be established by some of the neighboring ANs, thus allowing selected neighbors to become GN, if needed. Every node u maintains a timer *NeighborTimer*$_v$ for each neighbor v. When a node u receives a HELLO packet from a node v, u starts/resets v's timer and stores/updates v's information. If v is a HN, u also checks whether its own ID appears in the gateway field of the HN-HN connection list of some HN neighbor. If this is the case, node u becomes (stays) GN, otherwise it stays (becomes) AN. If no HELLO packet is received from v for *NeighborTimeout*$_v$ seconds, node v and its related information are removed from u tables. Every *Hello-Period*$_u$ seconds node u executes the *updateRole* procedure (Algorithm 1), through which it makes decisions based on the information collected so far. The procedure terminates with the broadcast of a HELLO packet carrying its possibly new role and connections. The *updateRole* procedure is made up

Algorithm 1 *UpdateRole*()

```
1: HeadConversion()
2: if (role(u) = HN) then
3:     HeadConnection()
4:     PruneConnections()
5: sendHELLO()
```

of two parts: A *HeadConversion* procedure during which each node decides whether to become a HN or not, and a second part in which a node that has become or is a HN executes the procedures *HeadConnection* and *PruneConnections* in order to establish HN-HN connections with the HNs that are 2 and 3 hops away. This is the sufficient condition for obtaining a connected backbone.

The *HeadConversion* procedure (Algorithm 2) is designed

[1] M-Backs is deployable in heterogeneous networks where not all nodes may have the resources for becoming part of the backbone.

so that at the end of its execution each node is either a HN or is associated to a neighboring HN, as RN, AN or GN (i.e., the set of HNs forms a dominating set). The procedure is executed by each node u and works as follows: If u is a RN (i.e., it has no backbone capability) it simply picks the highest weight node in its neighborhood as its own HN. If node u has backbone capability (i.e., it is a HN, an AN or a GN), it checks whether there exists a neighbor v that has picked u as its HN. If this is the case, u becomes (or stays) HN. If no neighbor has chosen u as HN, u picks as its own HN the node $v \in N(u) \cup \{u\}$ with the highest weight, where $N(u)$ is the set of the 1-hop neighbors of node u. If the highest weight node in u neighborhood is u itself, it becomes (or stays) HN, otherwise it maintains its current role (AN or GN). The ID of the selected node v will be included in the *headID* field of the HELLO packet that u will broadcast next.

Algorithm 2 *HeadConversion*()

```
1:  if ($role(u)$ = RN) then
2:      $headID(u) \leftarrow \max_{wt(v)}\{v \in N(u)\}$
3:  else
4:      if ($\exists\ v \in N(u)$ s.t. $headID(v) = u$) then
5:          $headID(u) \leftarrow u$
6:      else
7:          $headID(u) \leftarrow \max_{wt(v)}\{v \in N^{+}(u)\}$
8:      if ($headID(u) = u$) then
9:          $role(u) \leftarrow$ HN
10:     else if ($role(u)$ = HN) then
11:         $role(u) \leftarrow$ AN
```

The *HeadConnection* procedure (Algorithm 3) is executed by each HN u. The procedure sets up connections between u and the HNs that are 2 and 3 hops away through existing intermediate GNs or promoting to GN some of its AN neighbors. Through the information exchanged via HELLO packets, node u knows all the HNs that are up to 3 hops away. It is also aware of the HN-HN connections established by HNs that are 1 and 2 hops afar. The *HeadConnection* procedure uses this information for reducing the number of HN-HN connections that are required to obtain a connected backbone. In particular, node u builds its *Closure*(u) set, which is defined as the set of all HNs up to 3 hops away reachable through paths made up of HNs. If a 2 or 3 hops away HN v belongs to *Closure*(u) no connection needs to be established. If a 2-hop HN v does not belong to the set *Closure*(u), then node u checks whether there exists a higher weight 2 hops connection to v from a neighboring HN x. If this is the case, no connection is established to the 2 hops HN v, since v is reachable already through x. Otherwise, the connection is added to u connection list. If a HN v that is 3-hop neighbor of u such that $v \notin$ *Closure*(u), node u has to check three conditions. First, if there exists a 2-hop connection to HN v from a neighboring HN x. Node u needs to check also whether there exists a non-HN neighbor x such that x is connected to v through a path made up of only HNs. Discovering the existence of these paths is performed by checking that the value $CW(x, v)$ included in the HELLO packet of some non-HN neighbor x, is equal to ∞ (more below). If this is the case, there must be a path from node x to the 3-hop HN v such that their intermediate node y is also a HN. It follows that HN y belongs to $H_2(u)$[2], thus a 2-hop connection to y implicitly establishes a 3-hop connection to v. Finally, node u has to check whether there exists a 2-hop HN x such that x established a 2-hop connection to HN v. If so, the 3-hop connection can be avoided because there is a path made of two 2-hop connections to HN v. If none of these three conditions apply, u adds the 3-hop connection to its connection list. The function $CW(u, v)$ provides the weight of the "best" connection from node u to HN v. In our case, "best" is defined so that a shorter (i.e., 2-hop) connection is always preferred to a longer one (3-hop). If two connections have the same number of hops, the one with the higher weight

Algorithm 3 *HeadConnection*()

```
1:  for all $v \in H_2(u)$ s.t. $v \notin$ Closure$(u)$ do
2:      let $C \leftarrow (v, g, w, 2)$ be the highest weight 2-hop
        connection to $v$
3:      let $C' \leftarrow (v, g', w', 2)$ be the highest weight 2-hop
        connection to $v$ from HN $x \in H_1(u)$
4:      if ($w > w'$ or $C'$ = NULL) then
5:          addConnection($C$)
6:  for all $v \in H_3(u)$ s.t. $v \notin$ Closure$(u)$ do
7:      if (($\nexists$ a 2-hop connection to $v$ from $x \in H_1(u)$) and ($\nexists$
        $x \in N(u)$ s.t. $x \notin H_1(u)$ and CW$(x, v) = \infty$) and ($\nexists$
        $x \in H_2(u)$ s.t. $x$ has established a 2-hop connection to
        $v$)) then
8:          let $C \leftarrow (v, g, w, 3)$ be the highest weight 3-hop
            connection to $v$
9:          addConnection($C$)
```

Algorithm 4 *PruneConnections*(u)

```
1:  for all $C_1 \leftarrow (v_1, g_1, w_1, h_1)$ and $C_2 \leftarrow (v_2, g_2, w_2, h_2)$
    do
2:      if ($h_1 = 2 \wedge h_2 = 2 \wedge g_1 \neq g_2$) then
3:          if ($v_1 \in$ Closure$(g_2) \wedge v_2 \in$ Closure$(g_1)$) then
4:              if ($w_1 < w_2$) then
5:                  Delete$(C_1)$
6:              else
7:                  Delete$(C_2)$
8:          else
9:              if ($\exists$ a 2-hop connection $C^* \leftarrow (v^*, g^*, w^*, 2)$
                between $v_1$ and $v_2$ then
10:                 if ($w_1 = w^* \vee (w_2 < w_1 \wedge w_2 < w^*)$) then
11:                     Delete$(C_2)$
12:                 if ($w_2 = w^* \vee (w_1 < w_2 \wedge w_1 < w^*)$) then
13:                     Delete$(C_1)$
```

[2] $H_i(u)$ are the sets of HNs v that are i hops away from u, i.e., whose hop distance $d(u, v) = i$. Note that y cannot be in $H_1(u)$ otherwise node v would be in the set *Closure*(u).

is selected (possible ties are broken through unique IDs). If $v \in \mathit{Closure}(u)$ then $C(u,w) = \infty$.

After the connection list has been formed, a HN u runs the *PruneConnections* procedure (Algorithm 4) in order to remove some redundant connections from it. The *PruneConnections* procedure examines each pair of connections (C_1, C_2) in the connection list. If $C_1 = (v_1, g_1, w_1, h_1)$ and $C_2 = (v_2, g_2, w_2, h_2)$ are two 2-hop connections created by node u through two different gateway nodes $g_1 \neq g_2$, then node u checks whether HN v_1 belongs to the set $\mathit{Closure}(g_2)$ and, symmetrically, whether HN v_2 belongs to the set $\mathit{Closure}(g_1)$. If this is the case, one of the two connections is redundant, therefore node u prunes the connection with the lowest weight between C_1 and C_2. Node u also checks whether there exists a further 2-hop connection $C^* = (v^*, g^*, w^*, 2)$ between HNs v_1 and v_2. If such connection exists, then either C_1 or C_2 could be removed from the connection list, based on the weights of the three connections C_1, C_2 and C^*. (M-Backs correctness is not shown here for lack of space.)

III. Performance Evaluation

In order to evaluate the performance of *M-Backs*, we implemented it in the VINT Project ns-2 simulator [12] and its extension ns-MIRACLE [13]. We also implemented three other protocols for mobile backbones, namely ETSA [4], GDMAC [5] and TRUNC [7] for comparison purposes. In all the experiments nodes are distributed uniformly and randomly on a square area of side $L = 1000$m. In order to investigate the behavior of the selected protocols with different network densities we tested networks of different size. More precisely, we varied the number of nodes between 100 and 500 while maintaining the same simulation area. The corresponding average node degrees range between 6.06 and 29.81, i.e., from sparse to quite dense networks. For the sake of investigating the backbone protocol performance in terms of building and maintaining a backbone, in our experiments we consider all nodes to be backbone capable, i.e., all of them can be part of the backbone when needed. Each node is equipped with a 802.11g radio interface that is set to transmit at 6Mb/s. The transmission radius is 150m. The HELLO packets period has been set to 2 seconds while the neighbor timeout has been set to 6 seconds in all protocols implemented. Nodes move according to the Gauss-Markov mobility model, designed to adapt to different levels of randomness of motion via one tuning parameter α. Random (or Brownian) motion is obtained by setting $\alpha = 0$, while linear motion is obtained by setting $\alpha = 1$. Intermediate mobility patterns are obtained by varying the value of α between 0 and 1. In our simulations, we set $\alpha = 0.5$. The average nodal speed is up to 10m/s. (For lack of space here we only show results for when the average speed is 2.5m/s.) We performed two sets of experiments. The first one concerns the type of backbones produced by the tested protocols and the corresponding overhead (no data traffic was injected in the network). The second battery of tests concerns instead the investigation of the different backbones for routing network traffic, as opposed to routing it over the flat network.

All results shown in this section are the average over 100 different runs for each network size, which allows us to achieve 95% statistical confidence within 5% precision.

A. Building and maintaining connected backbones

We are interested in investigating the kind of backbones produced by the four considered protocols in terms of the resulting topology structure. To this aim, our experiments concern metrics such as *backbone size* (i.e., the fraction of network nodes ending up in the backbone), *backbone route length*, *backbone connectivity* (fraction of the simulation time during which the backbone is connected), *resilience to link breakage* and *backbone stability* (the number of times a node enter and leaves the backbone), and *backbone construction overhead*, measured by the average size of the control packets.

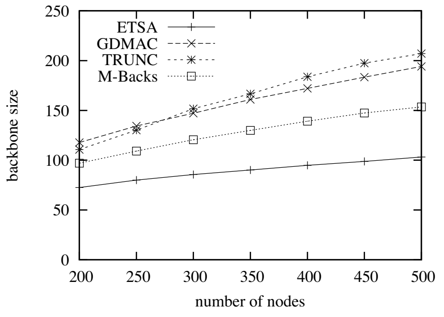

Fig. 2. Backbone size.

In Fig. 2 we show the average backbone size as the number of nodes increases. ETSA produces the smallest backbones (the backbone size ranges from 75 to 105 nodes, increasing with the network size). TRUNC and GDMAC build the largest backbones: Their size ranges between 120 and 195 nodes. M-Backs backbones lies in the middle, being made up of a number of nodes ranging between 95 and about 150. We observed that, as we increase the nodal speed, ETSA is no longer able to build small backbones. On the other hand, M-Backs is much less affected by nodal speed, being able to handle topology changes more effectively and more quickly. Therefore, M-Backs resulted to be the best option in a mobile scenario.

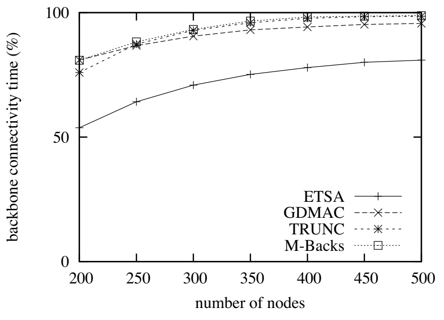

Fig. 3. Backbone connectivity time.

Fig. 3 shows the fraction of simulation time in which the backbone is connected. M-Backs maintains the backbone connected for most of the simulation time, namely, from 83% of the time (100 nodes) to 98% of the time (500 nodes) and outperforms all the other algorithms. TRUNC and GDMAC obtain results comparable to those of M-Backs because of their larger backbones, which are harder to disconnect. ETSA backbones, the smallest ones, stay connected only for 53% of the time in networks with 200 nodes and up to 80% of the time in networks with 500 nodes.

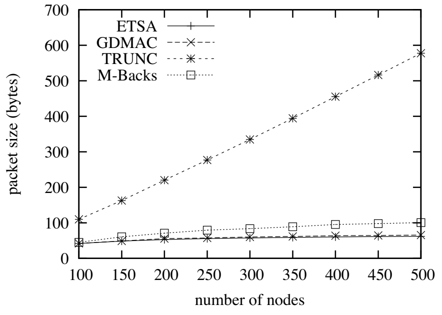

Fig. 4. Control packet size.

Fig. 4 depicts the average packet size for each protocol. We observed that TRUNC does not scale with network size. This is because the size of its control packet depends on a node *whole* neighborhood, which is potentially large and possibly proportional to the size of the network (e.g., $O(n)$, n being the number of nodes). All other protocols piggyback to their HELLO packets the list of their 1 and 2 hops *backbone neighbors* thus effectively controlling the HELLO packet size. Our experiments show an exponential gap between M-Backs and TRUNC packet size, the former therefore scaling gracefully with increasing network sizes. The average packet size for M-Backs is slightly higher compared to those of ETSA and GDMAC, because of the connection lists carried by the HELLO packets for the pruning procedure (Fig. 1). Despite this, M-Backs imposes low overhead (an average of 0.35Kbps vs. 0.25Kbps for ETSA and GDMAC, if the *HelloPeriod* is set to 2s).

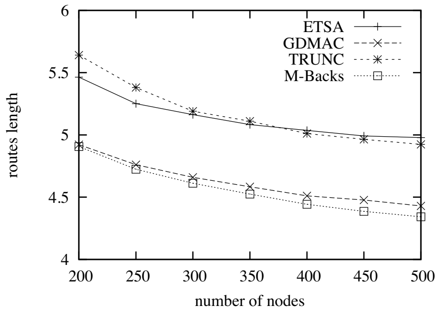

Fig. 5. Backbone route length.

Average route lengths between any pair of nodes are shown in Fig. 5. M-Backs is successful in building and maintaining the backbones with the shortest routes. GDMAC, because of its large backbones, obtains routes comparable to those of M-Backs, although at the higher price mentioned above. TRUNC does not succeed in taking advantage of it large backbones because its tree-like structure. ETSA, as expected, generates longer routes because of its smaller backbones. (Other key metrics such as backbone diameter, backbone robustness and backbone stability show trends that are strictly connected to the backbone size.)

B. Sending data over backbones

This set of experiments concerns routing data over the backbone. The selected routing protocol is the well-known AODV [11] whose flooding of route request (RREQ) packet is limited to the sole backbone nodes. We also implemented a multiple priority queue at the link layer so to impose three levels of priority among the packets: Backbone formation packets go first, then AODV control packets and finally data packets. Traffic packets are UDP packets, each 500B long (header included). Node queues can contain at most 100 packets. Data packets are generated according to a Poisson arrival process with parameter λ indicating the number of arrival per second, i.e., the number of packets injected in the network per second. Every time a new packet is generated a source and a destination are randomly and uniformly selected among all network nodes. We considered four different arrival rates, namely, we set $\lambda = 5$, 15, 25, and 35. The considered protocols are also compared to AODV running over the flat network topology (called *FLAT* in the following). In our experiments we investigate the *packet delivery rate*, i.e., the fraction of packets correctly delivered to their intended destination, and the *number of RREQ packets*, per second, which gives a measure of the effectiveness of the considered protocols in reducing the AODV control overhead. We have also investigated the *end-to-end latency* (only for delivered data packets), computed as the time it takes from when the packet appears in the queue of the source to when it arrives to the queue of the destination. (We do not picture this last metric, since the values for the tested protocols are very similar, with those for M-Backs being slightly better than all the others). For this set of experiments we show results for networks with 500 nodes moving at an average speed of 2.5m/s. (For higher nodal speeds, the relative differences among the tested protocols show similar trends.)

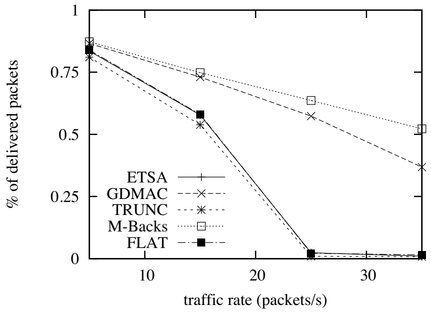

Fig. 6. Percentage of delivered packets.

Fig. 6 shows the percentage of data packets delivered by the tested protocols in networks with 500 nodes. M-Backs delivers the highest fraction of packets. In particular, M-Backs delivers between 87% and 52% of packets (this latter measure refers to when $\lambda = 35$, which is a congested scenario), while FLAT, as the traffic increases, increasingly fails to deliver packets (a nose dive from 84% down to 1%). GDMAC also improves with respect to AODV, despite its backbone size, which however still heavily reduce the route discovery overhead. ETSA and TRUNC do not show any improvement compared to FLAT. The reasons are different for the two algorithms. We found that ETSA exhibits an oscillatory behavior that keeps changing the backbone structure even in static networks. We assessed this problem by running ETSA on static topologies and showing the average number of nodal role changes per second. This oscillatory behavior invalidates some of the routes stored in the AODV tables, which keeps triggering the route discovery process. On the other hand, we noticed that TRUNC backbone formation operations are heavily affected by data traffic. Even in a static scenario, in presence of traffic TRUNC backbones are connected only for about 95% of the simulation time. Their size increases dramatically, ranging now from 55 to 150 nodes in networks of size 100 and 500, respectively (without traffic TRUNC backbone size never exceeds 100 nodes). These remarkable changes depend on the same protocols rules. TRUNC backbones are built using a complex message exchange scheme. Despite backbone formation packets take precedence over data packets, the presence of traffic (together with the large size of TRUNC formation packets) renders the building process weak to the point that it fails fairly often, even at no mobility or low nodal speed. As a consequence, the protocol is no longer able to properly connect backbone fragments and traffic packets are delivered to their destination only occasionally. In other words the TRUNC packets exchange process, which is the core of TRUNC backbone formation, is not robust. This confirms that building backbones through control information exchanged by HELLO packet is the quite a crucial optimization for backbone protocols.

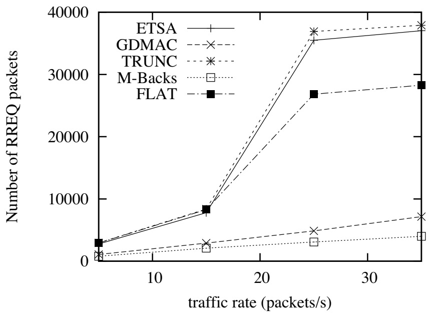

Fig. 7. RREQ packets/s.

Fig. 7 shows the average number of RREQ packets per second flooded in networks with 500 nodes. Once more we see that M-Backs outperforms all other backbone protocols and FLAT. In particular, compared to FLAT, M-Backs reduces the number of RREQ packets by 86%. As before, GDMAC is the only other backbone protocol able to improve over FLAT.

IV. Conclusions

We presented M-Backs, a new protocol for backbone formation and maintenance capable of efficiently cope with node mobility, quickly recovering from link breakage and that, whenever possible, maintains a connected backbone. The M-Backs backbone size is a small enough fraction of the backbone capable nodes to enable quick updates, but large enough to provide good connectivity and robustness in face of mobility. The performance of M-Backs has been compared to that of other backbone protocols for mobile multi-hop networks, namely, ETSA, GDMAC and TRUNC. We have observed that the key design choices of M-Backs are successful in making it low-overhead and in allowing it to deliver higher fractions of packets to their destinations with respect to all other backbone solutions and to AODV run over the flat network topology.

References

[1] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communications*, vol. COM-29, no. 11, pp. 1694–1701, November 1981.

[2] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A core-extraction distributed ad hoc routing algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1454–1465, August 1999.

[3] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, September 1997.

[4] H.-J. 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.

[5] M. A. Nanni and S. Basagni, "Mobile ad hoc backbones: Formation and maintainance," in *Proceedings of the International Radio Wireless Symposium, RWS 2010*, New Orleans, LA, 2010, pp. 613–616.

[6] 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.

[7] S. Lee, B. Bhattacharjee, A. Srinivasan, and S. Khuller, "Efficient and resilient backbones for multihop wireless networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 11, pp. 1349–1362, 2008.

[8] A. Srinivas, G. Zussman, and E. Modiano, "Mobile backbone networks: Construction and maintenance," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2006*, Florence, Italy, 2006, pp. 166–177.

[9] H.-J. Ju, I. Rubin, K. Ni, and C. Wu, "A distributed mobile backbone formation algorithm for wireless ad hoc networks," in *Proceedings of the 1st International Conference on Broadband Networks, BROADNETS 2004*, San Jose, CA, 2004, pp. 661–670.

[10] Y. Wang, W. Wang, and X.-Y. Li, "Efficient distributed low-cost backbone formation for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 681–693, 2006.

[11] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) routing," Request for Comments: 3561—Mobile Ad hoc NETworking (MANET) Working Group of the Internet Engineering Task Force (IETF), July 2003.

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

[13] 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 *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools, ValueTools 2007*, Nantes, France, 2007, pp. 1–8.