Using a Fairness Monitoring Service to Improve Load-Balancing in DSR*

Hugo Miranda
U. de Lisboa
hmiranda@di.fc.ul.pt

Luís Rodrigues
U.de Lisboa
ler@di.fc.ul.pt

Abstract

Many routing protocols for MANETs do not promote a balanced use of resources among the participating nodes, since they are designed to optimize other criteria, such as the number of hops in the message path. This behavior is undesired in open MANETs, where all users cooperate to maintain connectivity and expect the system to promote a fair distribution of load. This paper presents a fairness monitoring service that rates the effort of each individual node with regard to the other nodes in its neighborhood, measured as the relative number of messages the node is required to forward. We show that this information can be captured by a service that monitors the packets exchanged in the network. We illustrate the benefits of this service by showing how this information can be used to bias DSR, such that the load distribution is improved.

1 Introduction

A technology such as Mobile Ad Hoc Networks (MANETs) promotes the emergence of open ad hoc communities of users. These communities may support distributed games, chatting, file sharing, or Internet access in regions with incomplete infra-structured networked coverage. They are likely to emerge in locations that gather large numbers of civilians such as airports, shopping malls, convention centers, and university campus.

In this paper we are interested in the problem of balancing the load among the nodes of the network. It should be noticed that the importance of this problem heavily depends on the application of the MANET. In closed MANETs, where all participants are coordinated by a single authority and share a common

*Selected sections of this report were published in the Proceedings of the First International Workshop on Services and Infrastructures for the Ubiquitous and Mobile Internet (SIUMI'05), in conjuction with the 25th International Conference on Distributed Computing Systems (ICDCS-25), Columbus, Ohio, USA, June 2005. This work was partially funded by LaSIGE and by FCT project MICAS - MIddleware para sistemas adaptáveis ao contexto, POSI/EIA/60692/2004 through POSI and FEDER.

goal, load balancing may not be the most relevant goal. In fact, it is even likely that some participants may be willing to sacrifice all their local resources to the benefit of some global goal. On the contrary, in Open MANETs, where each node is locally and independently administered by a different user, an unfair distribution of load may encourage the owners of overloaded nodes to avoid cooperation or simply disconnect from the network. Ultimately, this behavior may cause the disruption of the entire network operation.

Different proposals have been presented to promote or enforce the cooperation of nodes in MANETs [3, 4, 5]. Most of them reward the participation and/or penalize the lack of cooperation of nodes. In this paper we follow a complementary path that consists in building mechanisms to support the development of network protocols that offer a fairer usage of resources. The rationale of our approach is the following: if the network protocols are fairer, users will be more likely to adopt a cooperative behavior.

It is easy to find algorithms for MANETs that do not offer a fair distribution of load among the participants. Typically, to save the bandwidth and energy consumption required for dynamic reconfiguration, nodes elected to perform a given role in the system, are forced to perform that role until they fail or disconnect. Concrete examples of specific roles are replica caches, servers [1], Mobile IP routers [10], or cluster heads [6]. From these examples, one can observe that load balancing is a vertical concern, that must be addressed at all levels of system software.

In this paper, we are concerned with the fair use of resources promoted by the routing protocols. In particular, we are concerned with the number of messages that each node is required to forward on behalf of other nodes. To address this issue, we propose the use of adaptive routing protocols that dynamically adjust the load imposed on each node. To drive the adaptation policies, this paper describes preliminary results concerning a *fairness monitoring service* that rates the effort of each device on message forwarding. To illustrate the benefits of this service, we propose an optional extension to the Dynamic Source Routing (DSR) protocol [8]. We would like to point that the our fairness monitoring service may also be useful to promote load balancing in other system layers: distributed protocols may use the output of the service to rank candidates to perform specific roles or to trigger a new role allocation when some unfairness threshold is reached.

The paper is organized as follows. Section 2 motivates our work by presenting the causes that may lead to an unfair load distribution in DSR. Related work is discussed in Section 3. The Fairness Monitoring Service is described in Section 4. The extension to DSR and the evaluation are presented in Section 5. Finally, Section 6 concludes the paper.

2 Motivation

Routing protocol for MANETs typically attempt to discover minimum cost paths between the source and the destination, mainly using hop count as the

cost metric, while at the same time attempt to minimize the signaling overhead of the protocol. Most of these protocols do not attempt to ensure a fair use of resources for two main reasons: *i)* load balancing conflicts with the shortest path goal (as it may require the use of non optimal paths) and, *ii)* to achieve this goal one may be required to augment the signaling cost of the routing algorithm (i.e., to use additional control messages or larger control messages).

Therefore, in this paper we are interested in: *a)* finding cost effective mechanisms to extract information about the relative resource usage in the network; *b)* use this information to bias the routing protocol such that a fairer use of resources is achieved. Before we introduce our approach, and for self-containment, we provide here a brief overview of the Dynamic Source Routing (DSR) protocol. The full description of the protocol can be found in [9].

2.1 DSR Overview

DSR, as the name implies, uses source routing, i.e., the header of data messages includes the route to be followed. Each intermediary hop is required to inspect the packet header to determine the address of the next hop. Sources of packets learn new routes by flooding a RouteRequest packet in the network and waiting for the correspondent RouteReply. When a node receives a RouteRequest packet it may: *i*) send back a RouteReply packet if it is the final destination of the request; *ii*) send back a RouteReply packet if it knows a route to the destination; *iii*) otherwise, rebroadcast the RouteRequest packet, after appending its own address to the DSR header (but only if the RouteRequest is not a duplicate). A source may also learn a new route by inspecting the routes carried in the header of packets snooped from the network.

2.2 Unfairness Example in DSR

Scenarios where nodes exhibit a negligible relative movement to each other (for instance, when users are in a library or a train) are good examples to illustrate both the advantages and disadvantages of the optimizations included in DSR. On one hand, route stability makes possible to reuse previously learnt routes, and allows DSR to reduce the signaling cost associated with the discovery of new routes. On the other hand, the reuse of routes may heavily unbalance the load distribution in the network.

This behavior is illustrated by the example in Figure 1. Assume that node A initially broadcasts a route request for node D. The route reply containing $A \rightarrow E \rightarrow G \rightarrow H \rightarrow D$ will be snooped by node F. If later node B also broadcasts a route request for node D, F will use the cache to reply with the $F \rightarrow G \rightarrow H \rightarrow D$ route. Finally, by snooping the packets sent by B, C can also learn the route $B \rightarrow F \rightarrow G \rightarrow H \rightarrow D$ to reach D. Although several alternative routes with the same length exist from node C to node D, C will use the same route for any of the nodes B, F, G, H or D.

This example shows that one of the sources of unfairness in DSR is the promiscuous share of routes, which does not take in consideration the load

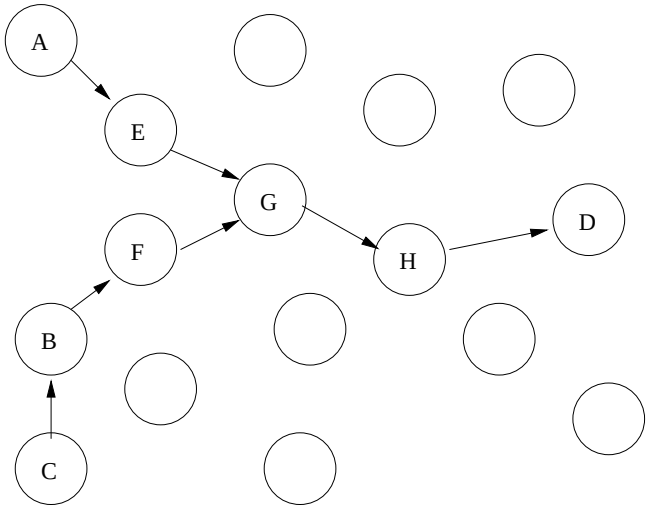

Figure 1: The unfairness problem

imposed at each intermediary hop. In these scenarios, nodes using DSR tend to converge to the same few routes, without exploiting other possible routes. In some cases, the selected routes may not be the optimal routes, even from the path length point of view.

3 Related Work

Several routing protocols for MANETs have addressed the fairness issue. Power-aware routing protocols rely on the energy information provided by each node to select message routes. Typical examples of fairness metrics used by power aware routing protocols are the "time to network partition" or "node's lifetime". It should be noted that the optimization of these metrics is far more easier in a closed system (where all nodes are forced to cooperate) than in an open system, where each user makes its own decisions. In particular, we emphasize that the *fairness* described in this paper is different from the maximization of the lifetime of each node (as described in [13]) because the fairness is independent of the characteristics or behavior of each device and user. Fairness will not attempt to extend the lifetime of nodes whose battery started with a lower energy reserve or that consume more power.

Congestion and fairness are related: a congested node is probably being unfairly overused. In load-aware routing protocols like ABR [14] and DLAR [11], intermediate hops append load information to the route discovery messages. To be useful, these protocols require the cooperation of all nodes. In the Hotspot Mitigation Protocol (HMP) [12] and in the extension to DSR proposed by Hu and Johnson [7] congested nodes temporarily suspend their normal route discov-

ery behavior by ignoring incoming route requests destined to other nodes. This all-or-nothing approach may disrupt the communication between two endpoints if no alternative route exists. To circumvent this problem, the two previous protocols include a special flag in its messages. In [7], the flag should be activated in the RouteRequest packet header if no reply to a previous RouteRequest was received. An undesirable side-effect of this approach is the duplication of the Route Requests, even in the cases where no valid (congested or not) route exists.

Congestion is evaluated by load metrics such as the number of bytes per unit of time. Unfairness may not be detected by congestion indicators if a device is more solicited than others in the long term. Metrics that capture more adequately these scenarios are those that relate the traffic at each participant. The work described in the following section weights both congestion and fairness indicators and provides numeric indications allowing a reaction that can be proportional to the severity of the situation. Its overhead is restricted to a small increase in computational power and memory, which is expected to present an acceptable trade-off for the more efficient utilization of the network interface [13, 9] and for increasing the probability of user cooperation.

4 The Fairness Monitoring Service

We now describe our fairness monitoring service. The service provides two metrics that capture different network conditions. The metrics are derived from information extracted from packets snooped from the network; therefore, their evaluation does not require the exchange of any additional dedicated control messages. We start by describing the state maintained by our monitoring service and then proceed to present and discuss the metrics it offers.

4.1 State

Each node i keeps a packet list pl_i containing the following information for each packet snooped from the network: *i)* a time-stamp of the moment at which the packet was snooped, *ii)* the address of the node that forwarded the packet and, *iii)* the packet size. Note that more than one entry may exist for the same packet, if it is successively forwarded by several neighbors of i. We define the neighborhood of node i as the set of nodes whose transmission can be listened by node i. The algorithm keeps a few other variables, that are derived from the content of pl_i.

$\boldsymbol{pkts}_i$ is the number of packets forwarded by i;

$\boldsymbol{pkts}_{\bar{i}}$ is the number of packets forwarded by other nodes;

$\boldsymbol{nnodes}_i$ is the total number of nodes that have sent at least one of the packets tracked in pl_i;

$\boldsymbol{tsize}_i$ is the sum (in bytes) of all packets tracked in pl_i;

The record of a message is kept in pl_i for a predefined period of time, denoted *historyperiod*. Entries older than *historyperiod* are discarded from pl_i, to make room for new entries.

4.2 Metrics

The service provides two metrics, denoted α and χ, derived from the information extracted from the packet list pl_i. These metrics evaluate, respectively, the fairness of the workload distribution between i and its neighbors, and the congestion at i's neighborhood.

Relative Regional Load The metric α_i evaluates the fairness of the work distribution between i and its neighbors. To achieve fairness, nodes try to keep their number of forwarded messages close to the average of forwarded messages by the remaining nodes in their region. The ratio between i's number of forwarded messages and the average number of messages forwarded by nodes in the neighborhood of i is presented in Eq. 1.

$$R(\alpha_i) = pkts_i \cdot \frac{nnodes_i}{pkts_i + pkts_{\bar{i}}}, (pkts_i + pkts_{\bar{i}}) > 0 \tag{1}$$

The ratio above is significant if based on a large number of messages exchanged, and insignificant if based on just a couple of samples. To account for this fact, we define the following weight factor, depicted in Eq. 2.

$$W(\alpha_i) = 1 - \frac{1}{pkts_i + pkts_{\bar{i}}} \tag{2}$$

Our Relative Regional Load metric, α, is simply defined by the ratio defined in Eq. 1 weighted by the factor of Eq. 2. Note that when either the total number of samples or the ratio $R(\alpha_i)$ are below some given thresholds, we simply default α_i to 0. The definition of α is given in Eq. 3.

$$\alpha_i = \begin{cases} & pkts_i + pkts_{\bar{i}} \geq min\_list\_size \\ W(\alpha_i) \cdot R(\alpha_i), & \wedge \\ & R(\alpha_i) > min\_avg \\ 0, otherwise & \end{cases} \tag{3}$$

Regional Congestion The metric χ_i evaluates the congestion at i's region. Congestion is usually evaluated by the number of messages waiting at the node's transmission queue [7]. Although this may be adequate for evaluating congestion from a node's perspective, it may fail to provide accurate information for the neighborhood, in particular, if the node is not being actively solicited for packet forwarding. *MAC layer utilization* is an alternative criteria that more suitably addresses the problem. It is defined, for each node, as the fraction of time during which the node either has one or more packets waiting in its transmission queue or if a node had attempted to transmit, the MAC rules would prevented it

from doing it [7]. In this paper, we propose a simpler congestion metric whose evaluation relies exclusively on the packet records kept locally at the packet list.

The χ_i metric estimates congestion in the neighborhood of node i by evaluating the bandwidth usage in the region. It is defined as a ratio between the bandwidth spent during the last *historyperiod* (given by $tsize_i$) and the available bandwidth during the same period (given by *historyperiod* $\cdot$ NABPS, where NABPS is the available bandwidth in bytes per second on the target network). Precisely:

$$\chi_i = \begin{cases} 0, pkts_i + pkts_{\overline{i}} < min\_list\_size \\ \left(\frac{tsize_i}{historyperiod \cdot \text{NABPS}}\right)^2, otherwise \end{cases} \quad (4)$$

The ratio is squared to make the function more steep, thus promoting more congested regions in detriment of less congested ones. If the number of messages that have been sent recently is below a given threshold (and therefore not significant), the metric χ_i simply defaults to 0.

Note that the network available bandwidth in bytes per second, NABPS, is a constant for each target network, that depends on the maximum link bandwidth and on the MAC protocol. For instance, for IEEE 802.11 like networks we may have:

$$\text{NABPS} = \frac{\frac{B \cdot 10^6}{8}}{3} = \frac{125 \cdot 10^3 \cdot B}{3} \approx 41000 \cdot B \quad (5)$$

where B denotes the network advertised bandwidth in millions of bits per second (Mb/s). In this case, the total bandwidth is first divided by eight to convert the bandwidth to bytes per second and then by three to reflect an IEEE 802.11 like network, where the useful bandwidth is typically one third of the total.

4.3 Discussion

Impact of packet snooping Network snooping consumes device resources, in particular, computational power and energy. However, research results indicate that wireless network interfaces spend comparable amounts of energy receiving packets and listening to the network [13]. Therefore, for networks running MAC protocols that do not allow network cards to enter the sleep mode (like IEEE 802.11), it is expected that the overhead imposed by the monitoring service be limited to the small computational power required to keep a list of records, each taking only a few bytes.

Relevance of congestion information The α_i metric adequately relates the effort of i with that of its neighbors. However, if a cluster of nodes is being unfairly used, the metric will return progressively lower values as the nodes get more close to the center of the cluster. This effect happens because the average for nodes at the borders will take into account nodes outside the cluster whose

contribution is below average. This creates the adverse effect of favoring traffic to enter the center of the cluster.

The χ_i metric, on the other hand, does not take into account the number or individual contributions of the nodes. Contrary to α_i, it is expected to grow from the center to the borders of the cluster.

Applications of metrics Fairness information can be obtained by combining both metrics. The selection of an adequate function to combine both α_i and χ_i is application dependent: the exact thresholds or the weights used to balance each of the metrics must be tuned depending on the behavior required by the middleware service or protocol. The next section illustrates a concrete meaningful combination of the two metrics provided by our monitoring service that allows to improve fairness in DSR.

5 Application: Biased DSR

To illustrate the usefulness of our fairness monitoring service, we now describe an extension to the Dynamic Source Routing protocol that makes use of the metrics it provides. The goal of this extension, called simply Biased DSR, is to mitigate route concentration by leveraging packet routing across different nodes. This extension does not require the exchange of additional messages and is fully compatible with nodes running implementations following the DSR Draft [9].

The metrics provided by the service grow proportionally with the node effort and congestion in the region. Each of them may return any positive value and have unrelated scales. To harmonize these functions, we define two coefficients, respectively k_α and k_χ. These factors are also used to rate the relevance attributed to fairness and congestion. The resulting combined metric, called effort index and denoted Φ is presented in Eq. 6. Its application on the DSR algorithm is presented in the following subsection.

$$\Phi_i = k_\alpha \cdot \alpha_i + k_\chi \cdot \chi_i \tag{6}$$

5.1 Delay of route requests

The key idea of the Biased DSR is to apply a different delay to the propagation of route requests according to the value of Φ_i. The effort index Φ_i is used to increase the probability of routes using less congested nodes being advertised and selected. When receiving a route request, node i, running Biased DSR, will evaluate Φ_i and multiply it by a constant *ref_delay* to determine the delay to be applied to the route request. The route request will be handled following DSR standard procedures if the outcome determines a negligible delay[1] and will be discarded if the delay exceeds some constant *max_delay*. For intermediate values, the route request will be processed according to the DSR standard after the

[1] In the current implementation, all values below 0.001s are considered negligible.

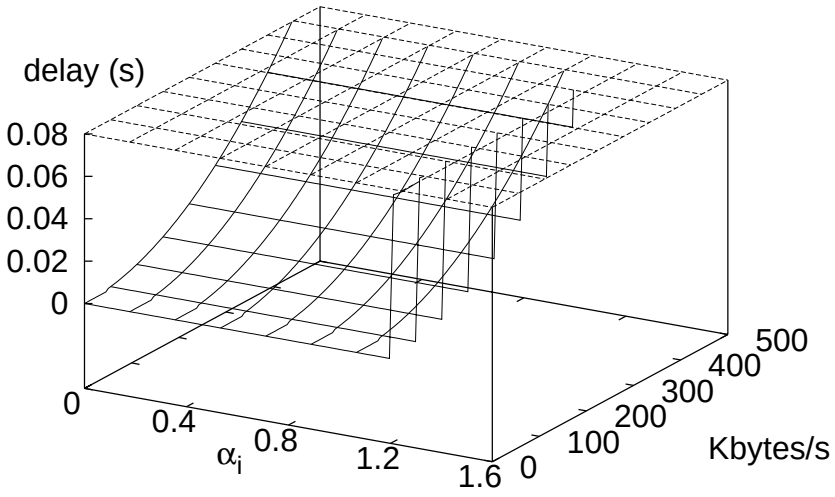

Figure 2: Delay applied to route requests

computed delay has expired. Figure 2 shows the adaptiveness of the algorithm using the constants presented in Table 1. Its sharp increase when α_i reaches 1.2 is explained by the threshold imposed by *min_avg*.

The delay of route requests is a flexible mechanism that favors the discovery of alternative routes circumventing congested regions because *i*) increases the probability that other nodes have already received and forwarded a different copy of the route request, thus removing the congested link from the path; *ii*) increases the probability of the source node to receive other cached routes with the same hop count but which were available further away from the source; *iii*) does not eliminate the possibility of having the route used. This effect is amplified because in a congested sequence of hops, the route request will be successively delayed, thus increasing the probability of not having the route selected.

5.2 Evaluation

We compare our route request delaying algorithm with the baseline DSR to confirm that it improves DSR fairness and to assert that it does not significantly degrade DSR performance. A network composed of 70 nodes running IEEE 802.11 network cards at 11Mb/s in a region of 2000x250m was simulated using the network simulator ns-2, v. 2.27. DSR was evaluated using the im-

min_avg	1.2	*max_delay*	0.08
min_list_size	10	k_α	0.8
historyperiod	3s	k_χ	1.0
ref_delay	0.08	B	11

Table 1: Values used in the simulations

plementation provided with the simulator, without flows. Our extension was implemented by changing the DSR base code and its procedures start with the beginning of the simulation. All simulations run for 300s and use Constant Bit Rate (CBR) traffic generators with 8 packets of 512 bytes per second. No error model was applied. The wireless network cards range is of 300m.

Ten independent movement files were defined for each of three movement speeds: stopped, slow (1-2m/s, 50s pause time) and fast (2-5m/s, 40s pause time). Nodes move according to the random waypoint model [2] at a speed randomly selected in the interval specified. To simulate different traffic conditions, 3 traffic scenarios were defined respectively with 10, 15 and 20 CBR connections active at each moment. Each CBR connection lasts for a randomly selected interval between 40 and 80 seconds, after which it is replaced by another between other randomly selected endpoints. The algorithm was executed with the values presented in Table 1, which provide an acceptable trade-off between reliability and unfairness mitigation. Unless noted, the results average the combination of each traffic file with the 10 independent movement files for the same speed.

Unfairness mitigation Fairness was evaluated measuring the standard deviation of the number of link layer frames sent by each node. Figure 3 shows that the average standard deviation of the number of link layer frames sent by all nodes is 9% to 30% lower which represent a significant gain in load distribution. As expected, DSR is less fair in scenarios with higher route stability and with more traffic.

To emphasize the advantages of an unfairness mechanism, we compare one particular run with 20 CBR connections and nodes moving at fast speed. Using the baseline DSR, the more active node forwarded 2.93% (5293) of the link layer frames while the less active was responsible for forwarding 0.28% (524). The difference decreases when using Biased DSR. The node more requested forwarded 2.04% (4159) of the frames while the one less active contributed with 0.74% (1508). The standard deviation in this particular run where respectively of 1008.08 and 593.86, which represents a reduction of 41%.

Reliability The comparison of the packet delivery ratio highlights the trade-offs required by unfairness mitigation. Figure 4 shows that Biased DSR does not follow the packet delivery ratio degradation of DSR. This results from refusals of nodes more unfairly charged to be included in the paths of new route requests, which in turn increases the number of route discoveries and the overall traffic in

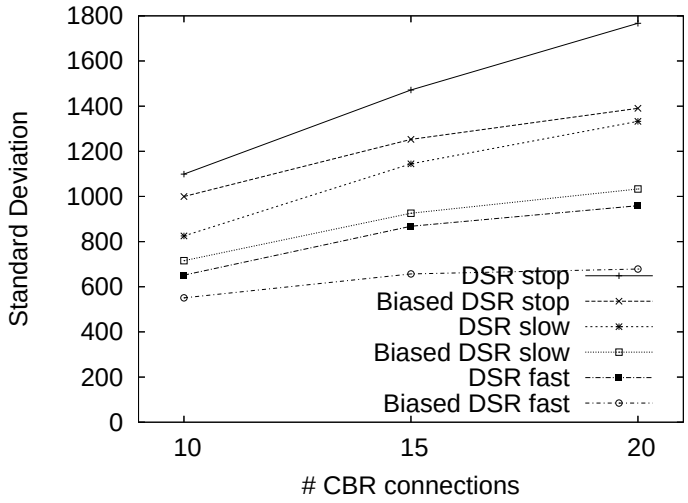

Figure 3: Standard deviation

the network. However, we believe that in some situations, a loss of 3% to 4% in reliability might be a reasonable price to pay to get the fairness gains presented above.

Efficiency The efficiency is evaluated by measuring the number of link layer frames required per data packet generated during the simulation. To better profile each protocol, the number of link layer frames is further separated in data frames and total, thus it also takes into account the number of DSR routing protocol frames forwarded by the nodes. The results are presented in Table 2. It should be noted that none of these indicators reliably estimates the route length given that the total of link layer frames includes retransmissions.

Results show that unfairness mitigation can keep the pace with DSR and even provide some marginal gains for low traffic situations, but begins to degrade as the traffic increases. The results from the column Data frames/Packet Delivered in Table 2 show a small increase that possibly indicates that some routes may have a few additional hops. The increase in the Total frames/Packet Sent column is interpreted as resulting from a larger number of route request messages that result from the fact that unfairly overloaded nodes may omit to reply to route requests.

Resource consumption Efficiency results show that Biased DSR has an overall energy consumption slightly higher. However, the standard deviation results indicate that the energy spent by each node will be closer to the average, which implicates that the burden will be placed on less used nodes. The memory consumption of the protocol is dependent of the *historyperiod* parameter. The

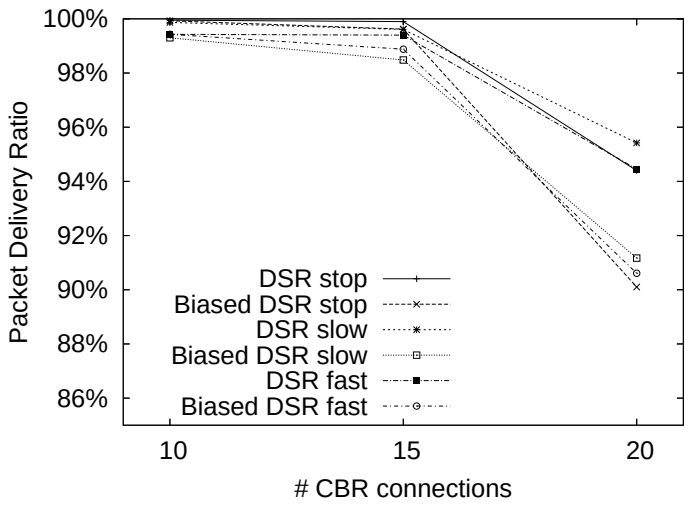

Figure 4: Packet delivery ratio

current implementation of Biased DSR in C++ requires 20 bytes per packet recorded in pl_i. During the simulations it was observed that pl_i never exceeded 600 records. These values suggest a memory consumption of approximately 12Kb per node for an *historyperiod* of 3s, which can be easily accepted for the majority of wireless enabled devices like laptops and PDAs.

6 Conclusions and Future Work

In Open MANETs, protocols need to consider the fair division of the tasks to balance energy consumption among the participants. Unfairness is a problem distinct from congestion in the sense that users may notice that their devices are being excessively used when compared with other participants even when no congestion exists. It should be noted that this is not an issue exclusive of routing: decisions on the location of services, for example, should take into account the node's past and current history.

This paper has presented early results from the use of a service that monitors the network to provide information concerning the fair division of activities among the nodes on the network. The service is completely local: it does not require the exchange of control messages with other nodes; instead it requires a small additional computational power at the devices where it is executed.

To illustrate the usefulness of this service, an extension to the Dynamic Source Routing protocol that enhances the fair distribution of load between the participating nodes was developed. This algorithm relies on the information provided by the fairness monitoring service to delay or drop route requests at

#Con	Protocol	Link Layer Frames			
		Data		Total	
		Deliv.	Sent	Deliv.	Sent
10	DSR stop	3.26	3.25	3.32	3.32
	Biased stop	3.25	3.25	3.32	3.32
	DSR slow	2.86	2.86	3.01	3.00
	Biased slow	2.87	2.85	3.03	3.01
	DSR fast	2.68	2.67	2.97	2.95
	Biased fast	2.69	2.67	2.98	2.96
15	DSR stop	3.27	3.26	3.34	3.34
	Biased stop	3.32	3.31	3.47	3.45
	DSR slow	2.94	2.92	3.10	3.09
	Biased slow	2.98	2.94	3.27	3.22
	DSR fast	2.71	2.69	3.00	2.99
	Biased fast	2.73	2.70	3.09	3.06
20	DSR stop	3.66	3.46	3.94	3.72
	Biased stop	3.90	3.52	4.64	4.18
	DSR slow	3.08	2.94	3.37	3.22
	Biased slow	3.24	2.96	3.90	3.56
	DSR fast	2.97	2.81	3.41	3.22
	Biased fast	3.16	2.87	3.92	3.55

Table 2: Link Layer frames/packet ratio

overused or congested nodes. Its use is optional on networks using the DSR protocol. Evaluation results show that the algorithm enhances fairness at the expenses of a slight increase in control traffic.

This position paper intends to open for discussion early work in a path that we find promising and that can be further improved. For instance, the protocol does not adequately address situations where routes are kept stable and active for long periods. To address this problem we will require the use of corrective measures after the route discovery procedure, for which no adequate metric has been presented here. Replies from route caches can also be improved if multiple routes exist and some information about the tentative next hops is kept locally. Future work will proceed by developing metrics to address these limitations.

References

[1] M. Avvenuti, D. Pedroni, and A. Vecchio. Core services in a middleware for mobile ad-hoc networks. In *Proc. of the 9th IEEE Work. on Future Trends of Distributed Computing Systems (FTDCS'03)*, pages 152–158, 2003.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols.

In *Proc. of the 4th Annual ACM/IEEE Int'l. Conf. on Mobile Computing and Networking*, pages 85–97, 1998.

[3] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation Of Nodes - Fairness In Distributed Ad-hoc NeTworks. Technical Report IC/2002/01, Swiss Federal Institute of Technology, Lausanne, 2002.

[4] L. Buttyán and J. P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proc. of the 1st IEEE/ACM Work. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2000.

[5] Stephan Eidenbenz, V. S. Anil Kumar, and Sibylle Zust. Equilibria in topology control games for ad hoc networks. In *Proc. of the 2003 Joint Work. on Foundations of Mobile Computing*, pages 2–11, 2003.

[6] M. Gerla, T. J. Kwon, and G. Pei. On-demand routing in large ad hoc wireless networks with passive clustering. In *Proc. of the Wireless Communication and Networking Conference*, volume 1, pages 23–28, 2000.

[7] Yih-Chun Hu and David B. Johnson. Exploiting congestion information in network and higher layer protocols in multihop wireless ad hoc networks. In *Proc. of the 24th Int'l Conf. on Distributed Computing Systems (ICDCS'04)*, pages 301–310, 2004.

[8] D. Johnson, D. Maltz, and J. Broch. *Ad Hoc Networking*, chapter DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pages 139–172. Addison-Wesley, 2001.

[9] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet draft, IETF MANET Working Group, July 19 2004.

[10] Boris A. Kock and J. R. Schmidt. Dynamic mobile IP routers in ad hoc networks. In *Proc. of the 2004 Int'l Work. on Wireless Ad-hoc Networks (IWWAN'04)*, 2004.

[11] S.-J. Lee and M. Gerla. Dynamic load-aware routing in ad hoc networks. In *Proc. of the IEEE Int'l Conf. on Communications (ICC 2001)*, volume 10, pages 3206–3210, 2001.

[12] Seoung-Bum Lee, Jiyoung Cho, and Andrew T. Campbell. A hotspot mitigation protocol for ad hoc networks. *Ad Hoc Networks*, 1(1):87–106, 2003.

[13] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proc. of the 4th Annual ACM/IEEE Int'l Conf. on Mobile Computing and Networking*, pages 181–190, 1998.

[14] Chai-Keong Toh. Associativity-based routing for ad hoc mobile networks. *Wireless Personal Communications*, 4(2):103–139, 1997.