A Two-Side Perspective on Cooperation in Mobile Ad Hoc Networks*

Hugo Miranda and Luís Rodrigues

Departamento de Informática - Faculdade de Ciências
Universidade de Lisboa
Portugal
{hmiranda,ler}@di.fc.ul.pt

Abstract. Technological advances are leveraging the widespread deployment of mobile ad hoc networks. An interesting characteristic of ad hoc networks is their self-organisation and their dependence of the behaviour of individual nodes. Until recently, most research on ad hoc networks has assumed that all nodes were cooperative. This assumption is no longer valid in open (spontaneous) networks formed by individuals with diverse goals and interests. In such environment, the presence of selfish nodes may degrade significantly the performance of the ad hoc network. In open mobile networks, users may become selfish to defend their devices from the unfair load distribution presented by many protocols, designed to optimise other criteria, such as the number of hops in the message path.
This paper summarises research results addressing the problem of unfairness in open mobile ad hoc networks from a two-side perspective: the detection and punishment of selfish users and the definition of a more fair network environment that attempts to leverage the load among the participants to prevent selfish behaviour.

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 centres, and university campus. In such environment, different users, with different goals, share the resources of their devices, ensuring global connectivity. This sort of communities can already be found in wired networks, namely on peer-to-peer networks.

Until recently, the research effort on mobile networks (infra-structured or ad hoc) has focused mainly on routing and usually assumes that all nodes are

* This work was partially funded by LaSIGE and by FCT project MICAS - Middleware para sistemas adaptáveis ao contexto, POSC/EIA/60692/2004 through POSC and FEDER.

cooperative. These assumptions hold on application such as military or search-and-rescue operations, where all nodes belong to the same authority and their users share the same goals. There is a significant difference between the fixed and the mobile environment. Some resources, like battery power, are scarce in a mobile environment and can be depleted at fast pace with the device utilisation. In this scenario, open MANETs will likely resemble social environments: a group of persons can mutually benefit from cooperation as long as every participant contributes with approximately the same share. Selfish behaviour threatens the entire community. Optimal paths may not be available and cooperative nodes may become overloaded and be forced to abandon the community. In the worst case scenario, the network may become partitioned. This kind of problems as been already observed on peer-to-peer file distribution systems. For instance, it was observed in Gnutella that the number of sites providing valuable content to the network is a small part of the number of users retrieving information [1].

In order to effectively support open and spontaneous communities, MANETs should provide mechanisms and algorithms that discourage selfish behaviour. However, selfish behaviour may emerge as a consequence of the unfair division of tasks by middleware or routing protocols. Therefore, protocols must also consider the leverage of the resources by the participants to prevent users from justifiably adopting selfish behaviour. 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. We argue that load balancing is a vertical concern, that must be addressed at all levels of system software.

This paper surveys research results of the authors [2, 3] on mechanisms to provide a fair division of tasks in a MANET. Fairness is improved under two complementary perspectives: to identify and punish selfish nodes and to enhance existing protocols with respect to the fair division of tasks. For the later, this paper describes preliminary results concerning a *fairness monitoring service* that rates the effort of each device on message forwarding. We would like to point that our fairness monitoring service may be useful to promote load balancing in different 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, routing protocols can use it to enhance load balancing between equivalent routes.

The paper is organised as follows. Related work is discussed in Section 2. The Fairness Monitoring Service is described in Section 3. The algorithm for monitoring selfishness behaviour is presented in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

A possible approach to mitigate selfishness is to reward cooperative nodes. In modern societies, services are usually provided in exchange of an amount of

money, previously agreed between both parts. The Terminodes [4] project proposes the use of a virtual currency named *beans*. This currency is used by nodes to pay for the forwarding of their own messages. The payment is distributed by the intermediary nodes that forwarded the message. Unfortunately, current implementations of reliable digital cash systems require several different participants and the exchange of a significant number of messages. To reduce this overhead, Terminodes assume that nodes are equipped with a tamper resistant security module, responsible for all the operations over the *beans* counter. The security module refuses to forward messages whenever the number of beans available are not sufficient to pay for the service. However, this approach is inadequate for MANETs since that, to ensure the authentication of the tamper resistant modules, a Public Key Infrastructure (PKI) is used.

Reputation Systems Recently, the literature has been emphasising the advantages of using reputation systems for the establishment of trust relationships in large scale networks, for example to improve the reliability of packet delivery in ad hoc networks [5–7]. Reputation systems provide the mechanisms to capture and spread information about which users are reliable and which users are unreliable. In most of these systems, the initial trust level for a node with which other has never interacted is based on the reputation information disseminated by other nodes.

Reputation systems are an adequate tool for detecting misbehaviour. However, they fail to adequately recompensate nodes that for some reason are overused, for example because of their location. Nodes may become overloaded with requests because they are positioned in a strategic point of the topology. A well-behaved node that temporarily supports a huge amount of requests should be rewarded by this service. To the extent of our knowledge, no existing reputation system adequately addresses this problem.

Fairness in Routing Protocols Several routing protocols for MANETs have addressed the fairness problem as a side effect of the principal goal of improving packet delivery ratio. However, in the majority of the cases, these routing protocols rely on the information voluntarily provided by the nodes and therefore, are not adequate for an environment where it is assumed that nodes can lie. This is the case of the power-aware class of routing protocols, which 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".

Another example of protocols assuming the cooperation of the user are those in the load-aware category. Examples are ABR [8] and DLAR [9]. In these protocols, intermediate hops append load information to the route discovery messages. In the Hotspot Mitigation Protocol (HMP) [10] and in the extension to DSR proposed by Hu and Johnson [11] congested nodes temporarily suspend their normal route discovery behaviour by ignoring incoming route requests destined to other nodes. The decision to ignore route requests is local to the nodes.

3 The Fairness Monitoring Service

We now describe our fairness monitoring service. The goal of this service is to provide to middleware protocols running on a mobile device a metric allowing to compare the effort of the node with that of its neighbours. Middleware protocols can for example volunteer for coordinating the next round of some protocol requiring a large number of messages if they feel that its effort is below average or refrain from propagating route requests if they have been forwarding a significant amount of traffic.

The fairness monitoring 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.

3.1 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 neighbours of i. We define the neighbourhood 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. The memory used by the fairness monitoring service is limited by imposing a maximum size to the number of entries in pl_i.

3.2 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 neighbours, and the congestion at i's neighbourhood.

Relative Regional Load The metric α_i evaluates the fairness of the work distribution between i and its neighbours. The metric is defined by the ratio between node i's number of forwarded messages and the average number of messages forwarded by nodes in the neighbourhood of i weighted by an attenuation factor to decrease the relevance of the metric when the total number of messages is low. 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. 1.

$$\alpha_i = \begin{cases} \left(1 - \frac{1}{pkts_i + pkts_{\overline{i}}}\right) \cdot pkts_i \cdot \frac{nnodes_i}{pkts_i + pkts_{\overline{i}}}, & pkts_i + pkts_{\overline{i}} \geq min\_list\_size \\ & \wedge \\ & pkts_i \cdot \frac{nnodes_i}{pkts_i + pkts_{\overline{i}}} > min\_avg \\ 0, & otherwise \end{cases} \tag{1}$$

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 [11]. In this paper, we propose a congestion metric whose evaluation relies on the packet records kept locally at the packet list.

The χ_i metric estimates congestion in the neighbourhood 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:

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

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.

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 behaviour 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.

3.3 Fair Routing in MANETs

To illustrate the usefulness of our fairness monitoring service, we now briefly describe an extension to the Dynamic Source Routing (DSR) [12] 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 IETF Draft [13].

DSR Overview For completeness, we now provide a brief overview of the DSR protocol. The interested reader is referred to [12] for an in-depth description.

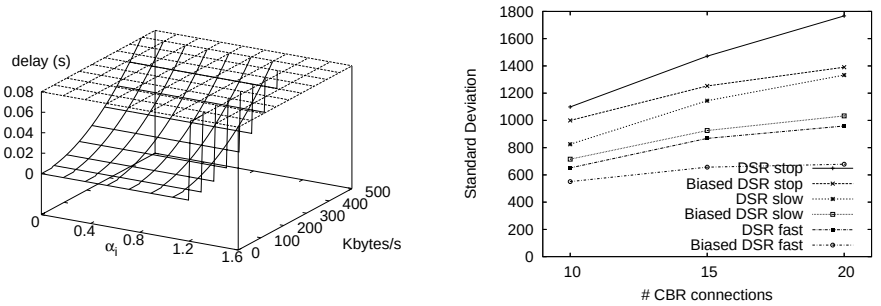
DSR, as the name implies, uses source routing, i.e., the header of data messages includes the route to be followed. Each intermediary hop is required to inspect the packet header to determine the address of the next hop. Sources of packets learn new routes by flooding a RouteRequest packet in the network 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.

Delay of Route Requests The metrics provided by the fairness monitoring 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 harmonise 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. 3.

$$\Phi_i = k_\alpha \cdot \alpha_i + k_\chi \cdot \chi_i \tag{3}$$

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 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 computed delay has expired. Figure 1(a) shows the adaptiveness of the function.

The delay of route requests is a flexible mechanism that favours the discovery of alternative routes circumventing congested regions and therefore of leveraging the load of more congested nodes.



(a) Delay applied to route requests

(b) Standard deviation

Fig. 1. Evaluation of Biased DSR

Evaluation The improvements of Biased DSR in the distribution of the messages forwarded by the nodes in a MANET was evaluated measuring the standard deviation of the number of link layer frames sent by each node. Figure 1(b) shows the standard deviation of the number of link layer frames sent by 70 nodes randomly deployed in a region of 2000mx250m using the *ns–2* network simulator. The x axis shows the number of simultaneous connections, between random endpoints. In each connection, a node sends 8 packets of 512 bytes per second using UDP. Connections last for a random interval between 40s and 80s, after which they are replaced by another between randomly selected endpoints. Nodes move at speeds between 1 and 2m/s (slow), 2 and 5m/s (fast) or do not move for the entire simulation (stop). It can be seen that the standard deviation is 9% to 30% lower when using Biased DSR, what represents a significant gain in load distribution. As expected, DSR is less fair in scenarios with higher route stability and with more traffic.

4 Selfishness Detection

The previous algorithm provided the mechanisms for nodes to rate their effort in relation with its neighbours. In this section, we describe an algorithm providing the mechanisms to detect the misbehaviour of other nodes.

People tend to cooperate as long as they notice that there is a fair division of tasks in a group. This observation can also be applied in MANETs where the unfairness of the service division may prevent users from using their own devices. Therefore, any selfishness prevention algorithm must also include some mechanisms to foster a fair distribution of resource consumption. This can be achieved if nodes are allowed to present some "justified selfishness", by refusing some of the requests received, forcing the clients to search for alternatives.

This section presents a new selfishness prevention algorithm presenting a long lived memory that allows nodes to be rewarded by services provided in the past. Additionally, the algorithm introduces new features by tolerating justified

selfishness. The algorithm uses only one type of message, that each node periodically broadcasts. Furthermore, nodes that do not participate in the algorithm may be still allowed to benefit from the MANET.

The algorithm supports the management of fairness by allowing nodes to publicly declare that they refuse to forward messages to some nodes. Nodes are uniquely identified by an address which we assume that can not be changed. Each node i maintains three variables:

friends$_i$ The set of neighbours to which i is willing to provide services.
foes$_i$ The set of neighbours to which i is not willing to provide services.
selfish$_i$ The set of neighbours which are known to act as if i is a foe.

We define a neighbour node of i as a node that has recently sent a message received or snooped by i. Each node i periodically advertises the content of these three variables in a control message named SelfState broadcasted to i's neighbours.

In order to evaluate the degree of selfishness of other nodes, and to detect inaccuracies in the information they advertise, each node keeps a record of received SelfState messages. For each other neighbour node j, node i also keeps a number of status variables allowing it to monitor j's behaviour. For example, i keeps track of the messages recently forwarded by j and of those that j ignored, of the list of j's friends and foes, the balance between the services provided and requested to j (credits), etc. We denote, for example, $friends_i^j$ as the list of friends advertised by j and stored by i.

Note that, in our protocol, it is acceptable for participants to have some foes. It is up to the community to evaluate if the number of foes a given node declares reaches an unacceptable level of selfishness. The information stored in these variables allows each node to keep track of recent behaviour of its neighbours, detecting misbehaviour and improving the trust on those that are more cooperative. Each node widely advertises its opinion on each other node to accelerate the detection and punishment of selfish nodes.

4.1 Execution Scenarios

We now illustrate the operation of the algorithm in some concrete scenarios. Due to the lack of space, only the most representative cases are presented.

All Cooperative Nodes, Lightly Loaded Network This is the most favourable scenario. Nodes will broadcast periodically one SelfState message, where the fields $foes_i$ and $selfish_i$ will be empty. The overhead introduced by the protocol is minimal: the periodic broadcast of one message.

All Cooperative Nodes, Unbalanced Load Network The nodes that have been more loaded by requests from others will start to move some of the nodes from the $friends_i$ to the $foes_i$ set. One can envision several criteria for the selection, using for example statistical information about the number of hops toward the usual destination of the messages of each node, or the number of credits. To preserve network connectivity, it is important that two or more

loaded nodes attempt to define disjoint $foes_i$ sets. Note that well-behaved nodes should keep providing services to peers, even if these peers consider them as foes.

One Selfish Node Selfish nodes are always penalised, whether they run the protocol or not, because the decision is taken locally at each node based on the information provided by its peers. When a node j refuses to forward messages, its address will be included in the $selfish_i$ set of the SELFSTATE messages of the other nodes. Nodes listening to these messages will start to remove the entries from $friends_i^j$ and placing them in $foes_i^j$. Whenever the rate $\frac{|friends_i^j|}{|friends_i^j|+|foes_i^j|}$ goes bellow some threshold, node i will begin to mark j as selfish and reciprocally become its foe. Eventually, j will have all other nodes to refuse its messages.

Malicious Selfish Node Malicious nodes will attempt to hide its selfishness from the remaining participants. Classifying a node as selfish is an operation local to each protocol based on the following conditions: *i*) it exhausted the credits limit or *ii*) it refused to forward messages to some nodes.

The protocol provides enough information to let each node to detect the envisioned attacks. A malicious node could try to defeat the system using one of the following methods: *i)* by declaring to be friend to a large number of fake nodes or; *ii)* by omitting to be foe of some nodes. Fake nodes are ignored when each node accounts for the number of nodes that each peer is serving, so the node gain no advantage from advertising dummy friends. Note that the limited transmission range of the network devices will make some nodes mark correct addresses as invalid, so their existence cannot be considered an attempt to subvert the protocol. Additionally, nodes can not hide their selfishness because information about the nodes known by each other node is crossed: if some node i mentions some other node j in its SELFSTATE message, and j omits i in its own message, then it is assumed that j is selfish for i.

Malicious nodes may also attempt to corrupt the MANET by falsely declaring other nodes as foes. However, in order to consider j as selfish, correct node i will use the information provided by all of i's neighbours. If the malicious node acts alone, its information will not be sufficient to achieve its goal.

The definition of a proper rate threshold must be subject to further study and may vary depending on the density of the nodes in the MANET area.

5 Conclusions and Future Work

The generalisation of wireless devices will soon turn MANETs into one of the most important connection methods to the Internet. However, the lack of a common goal in MANETs without a centralised human authority will make them difficult to maintain: each user will attempt to retrieve the most of the network while expecting to pay as less as possible. In human communities, this kind of behaviour is called selfishness. While prohibiting selfishness shows to be impossible over a decentralised network, applying punishments to those that present this behaviour may be beneficial.

On the other hand, protocols need to consider the fair division of the tasks to balance energy consumption among the participants. In Open MANETs un-

fairness is likely to promote selfish behaviour if users notice that their devices are being excessively used when compared with other participants. 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 history.

This paper surveyed recent results on two complementary mechanisms providing more adequate conditions for the wide generalisation of Open MANETs. A fairness monitoring service provides the means for nodes to leverage its effort with those of the remaining participants. A selfishness detection mechanism monitors the network to detect misbehaving nodes.

As future work, we expect to combine both mechanisms and apply them in protocols at different levels of the networking stack. Good candidates for the applications of both services are routing and group communication protocols.

References

1. Adar, E., Huberman, B.A.: Free riding on gnutella. First Monday **5**(10) (2000)
2. Miranda, H., Rodrigues, L.: Friends and foes: Preventing selfishness in open mobile ad hoc networks. In: Proc. of the Int'l Workshop on Mobile Distributed Computing (MDC'03), in conjunction with ICDCS'2003, IEEE (2003) 440–445
3. Miranda, H., Rodrigues, L.: Using a fairness monitoring service to improve load-balancing in DSR. In: Proc. of the 1st Int'l Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'05), in conjunction with ICDCS'2005. (2005) 314–320
4. Buttyán, L., Hubaux, J.P.: 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. Boukerche, A., El-Khatib, K., Xu, L., Korba, L.: Anonymity enabling scheme for wireless ad hoc networks. In: Workshops of the Global Telecommunications Conference, GlobeCom 2004, IEEE (2004) 136–140
6. Buchegger, S., Le Boudec, J.Y.: Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks. In: Proc. of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing. (2002) 403–410
7. Marti, S., Giuli, T., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: Proc. of Mobicom 2000. (2000)
8. Toh, C.K.: Associativity-based routing for ad hoc mobile networks. Wireless Personal Communications **4**(2) (1997) 103–139
9. Lee, S.J., Gerla, M.: Dynamic load-aware routing in ad hoc networks. In: Proc. of the IEEE Int'l Conf. on Communications (ICC 2001). Volume 10. (2001) 3206–3210
10. Lee, S.B., Cho, J., Campbell, A.T.: A hotspot mitigation protocol for ad hoc networks. Ad Hoc Networks **1**(1) (2003) 87–106
11. Hu, Y.C., Johnson, D.B.: 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). (2004) 301–310
12. Johnson, D., Maltz, D., Broch, J.: DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. In: Ad Hoc Networking. Addison-Wesley (2001) 139–172
13. Johnson, D.B., Maltz, D.A., Hu, Y.C.: The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet draft, IETF MANET WG (2004)