A Self-Organized Approach for Stimulating Cooperation in Mobile Ad Hoc Networks

Borzoo Shadpour, Shahrokh Valaee, and Baochun Li

Department of Electrical and Computer Engineering, University of Toronto

Abstract— **In civilian applications of ad hoc networks, a "virtual currency" is used to provide an incentive for cooperation by having a node to send its traffic at a given cost, while allowing it to profit every time it forwards others' traffic. However, there are situations when a node could run out of virtual currency, and enter a *broke* state. In this paper, we propose a self-organized mechanism called the *Broke Service*, which allows for a broke node to use the network to transmit its traffic, in addition to providing an incentive to stimulate non-broke nodes to cooperate with broke ones. We show that the proposed scheme delivers a reasonable quality of service under a wide range of network traffic.**

I. Introduction

Ad Hoc networks represent a challenging minimalist approach to network design, one that requires no infrastructure. As a result, providing network connectivity in civilian ad hoc networks requires cooperation among a large number of 'strangers'. In such cases, there are strong urges for individual users not wanting to cooperate with others and to remain 'selfish' [1]. In other words since routing the traffic of other nodes consumes considerable amount of bandwidth and processing power, each node could choose to send its own traffic at all times, and this has been shown to seriously affect the network throughput [2].

To enforce cooperation, two distinct approaches have been suggested. The first, which is the basis of this paper, is that of providing an incentive for cooperation. In [2], Buttyan and Hubaux, suggest the use of an incentive referred to as "nuglets", which materializes the universal cooperation concept of allowing for gaining a profit, versus having to pay a cost, associated respectively with routing for others, and with sending ones own traffic. In other words, a node is allowed to accumulate nuglets, which can later be used to pay for the cost of routing of its traffic. To police this process, nuglets will reside within a counter protected by security modules (*SM*), which are tamper-proof. They assume that each node uses a secure nuglet counter, which is increased by one every time the node routes a packet for other nodes, and is decreased by an estimate of the number of the intermediate hops (between the node and its destination), whenever the node sends its own traffic. Specifically, authorization for nuglet addition to counters is made without any possibility for user intervention and strictly through the communication between the *SM*'s.

The second approach to ensuring cooperation is suggested in [3] and [4], which are based on detecting malicious (non-cooperative) nodes, and in the latter, also on isolating them from the network. The difference between [2] and these approaches is that, whereas the former thrives at reaching an optimal point of functionality via discouraging non-cooperative behavior (since no cooperation means no profit), the latter ensures the survivability of the cooperative nodes, by threatening to leave out or to downgrade the service level of the non-cooperative ones. Though this approach promises a less complex security component, the first method is a more logical starting point, since it truly provides a simple tangible incentive for cooperation (encouraging nodes to maximize their profit), and if ever required, the second approach should only be used as a complement to the first.

A crucial problem addressed in this paper is that of how to provide network connectivity (the ability to transmit over an ad hoc network), when a particular node runs out of nuglets. A node could run out of nuglets due to poor network conditions such as the requirement for retransmission of packets due to a faulty wireless channel, which can quickly drain a node out of nuglets. In fact, this problem, also pointed out in [5], occurs with any system that uses an incentive for cooperation, and is particularly crucial for 'urgent traffic', where the users require immediate network connectivity and cannot afford the wait associated with accumulation of nuglets. We refer to such node as a *broke node*, and the required network service as the Broke Service.

II. Broke Service

The original contribution of this work is to improve the connectivity of broke nodes in a pure ad-hoc network. We consider and propose our solution within the nuglet-based framework of [2]. The proposed solution is that of *loaning*, which is particularly interesting since it can be performed 'on-the-fly' by the nodes' systems, and is hence suitable for the conditions of ad-hoc networks since it allows for nodes to remain self-organized. The scheme stimulates nodes to actively participate in the network, while allowing the broke nodes to experience less delay when urgent transmission is desired. The Broke Service is made up of two stages: loaning and payback.

The loaning stage itself involves two events: negotiation and authorization. In the beginning, a broke node sends a signal to all its neighbors indicating that it needs to use the Broke Service to send data, piggybacking *L,* the estimated number of nuglets that it will require (i.e. the loan value), λ, the arriving traffic rate to the Broke node's counter per unit time, and *Expiry Count*, *EC*, which is a measure of number of times a node has failed to payback its loans in the past. As explained later in the paper, the amount to be borrowed *L* is checked to ensure that it does not encourage begging in the network.

Upon receiving the broadcast, the neighbors (assuming that they have the Broke Service enabled), check the number of nuglets that the broke node is asking against a lender specific parameter, the *Min_Threshold*, and determine whether or not they can afford to lend to the broke node the full required amount. If the amount is feasible then the lenders send the expected rate of return α, which is a per transaction system parameter with an upper bound chosen depending on the current amount of risk associated with loaning, as explained later α is determined from the value of probability of success of the entire investment. Finally, when the broke node receives α, it checks to see if the rate of return is feasible and if it is, it sends a final acknowledgment, and the process of 'negotiation' is thereby completed. At this point, via signaling the broke nodes security module, the lender authorizes the addition of nuglets to the broke nodes counter giving it permission to send its traffic.

Fig. 1 summarizes the loaning process in a chronological order. The prefixes 'Broke' and 'Lender', at each stage refer to the party which takes the action, and *c* is the current value of the counter in lender's *SM*. Prior to broadcasting, the Broke node's loan value needs to be checked to ensure that following the receipt of the loan the node is not encouraged to stop cooperating with the network. This is referred to as the 'Beggar-Proof' mechanism, which basically, performs the function of checking the loan value against a maximum allowable limit.

```
1 Broke: Choose $L$ (the number of required nuglets)
2 Broke: if ($L$ is Beggar-Proof & no outstanding Break-
         Even) then Broadcast $L$, $\lambda$ & $EC$
3 Broke: else Decrease $L$ first (if possible)
4 Lender: if ($c - L > Min\_Threshold$)
5 Lender:    then Lookup the Risk Management table and
             send the corresponding value of $\alpha$
6 Broke: if $\alpha$ is feasible then send acknowledgement
7 Lender: Upon receiving ACK, authorize $L$ nuglet additions
          to the broke node's counter
8 Broke: Immediately use the $L$ nuglets to send data
```

Figure 1: Loaning process in Broke Service - Negotiation and Authorization

As it can be seen above, a key feature of the loaning process of the Broke Service is 'Risk Management', which is the function of assesing whether a lender should participate in the Broke Service. To enable lenders make such decision, provided a Broke node has no outstanding loans, the broke node sends three parameters to the lender. Having the three parameters *L*, λ, and *EC*, the lender's system will be able to predict how likely it is for it to receive its entire investment on the initial *L* value lent. A probability of success is assigned to the investment to be made, and this probability is later used to come up with a rate of return α, proportional to which a broke node commits to payback 'interest' on its loan. The mapping of the probability of success to the rate of return can be accomplished through a universally consistent risk management table provisioned onto every node's system by the manufacturers.

The second stage of the Broke Service is nuglet payback. Once the broke node has accumulated enough nuglets it will be required to payback its loan plus 'interest' in multiple portions to guarantee a higher probability of success for payback of the principal. Also as shown in the next section, payment values of *L* each are used since they obtain reasonable performance. The broke node is required not to send its own traffic until it has made the first payback (ie., the principal) during the time referred to as the *Break-Even* period. Thereafter, it will be able to send its traffic at a degraded level of service compared to the normal mode until it has paid its debt in full, this time period is referred to as the *return on investment*, *ROI*, period.

In addition, the Broke Service incorporates a *Broke Avoidance* mechanism ensuring that the broke node is protected against becoming broke, even after it is finished using the Broke Service. The key to this mechanism as shown in the next section is enforcing the counter of a broke node to remain above a *Broke Avoidance Margin* (*BAM*), for the duration of time when the Broke Service is in use. Fig. 2 summarizes the payback process in a chronological order.

```
1  payback ← 0
2  for [$payback < (\alpha+1)L$] do
3  if ($c > L + BAM$), then signal the lender then
4     if [$(\alpha+1)L - payback$] >=$L$ (after receiving lender's
      acknowledgment)
5           Authorize $L$ nuglet additions to lender's counter
6           $payback \leftarrow payback + L$
7     else
8           authorize [$(\alpha+1)L - payback$] nuglet additions to
            lender's counter
9           $payback \leftarrow (\alpha+1)L$
```

Figure 2: Payback process within Broke Node's Security Module

Finally, the security of the Broke Service is ensured using security modules, and the Broke Service is hence tamper-proof, and also integrates well with the cooperation system suggested in [2].

III. Performance Analysis

In this section we analyze the practicality of the Broke Service and show that in its proposed form it delivers a superior quality of service. To see this, we again break the function of Broke Service payback into two parts: Break-Even and *ROI*, and show that following our 'send-wait' mechanism, the Broke Service could be used with a reasonably low additional delay, when compared to the 'wait-send' approach implied in [2].

First, when using the Broke Service in an Ad-Hoc network, Break-Even can be achieved in a reasonable time within a flexible range of network traffic flows. For example, Fig. 3 shows the time for which a Broke node has to wait in order to accumulate enough nuglets to transmit 50 equal-sized packets to an average destination of 5 hops away from it. The robustness of the scheme can be seen by observing that if an appropriate *BAM* value is chosen, and when the incoming traffic is on average in the broad range of 1.8 to 5.3 packets/seconds, then a broke node that should have waited for at least 15 seconds in the previous schemes, can send its traffic 'immediately'. Afterwards, the node has to wait for only a maximum of up to 60 seconds to accumulate enough nuglets to payback its loan, with the additional peace of mind for knowing that the node will not become broke again after sending its traffic.

Once Break-Even is achieved, the *ROI* on loans made through the Broke Service can also be achieved in a reasonable time provided that the borrower does not send its own traffic excessively. For example, Fig. 4 depicts a similar scenario to Fig. 3, where various values of α, (ie., returns on the initial 50 packet investments) is desired. It can be seen that up to 300% net profit can be accommodated in less than 2 minutes, provided that the lower limit of the traffic range is increased slightly to 2.2 packets/seconds. On the other hand, if up to a maximum of 200% net profit (ie., $\alpha = 2$) is desired, then the lender can expect the profit to be attainable even within the same range of traffic as in Fig. 3.

Note that the *ROI* cases considered above are probably not going to be the most frequent scenarios encountered, and in a more realistic example, where a lender aims for a reasonable 100% or less return on its investment, it only has to wait up to a maximum of 90 seconds before it receives its entire investment. That being said, it is natural to expect that the more the Broke Service is used within the network, the less ROI is required to justify its usage by lenders, and the delay will also be further improved.

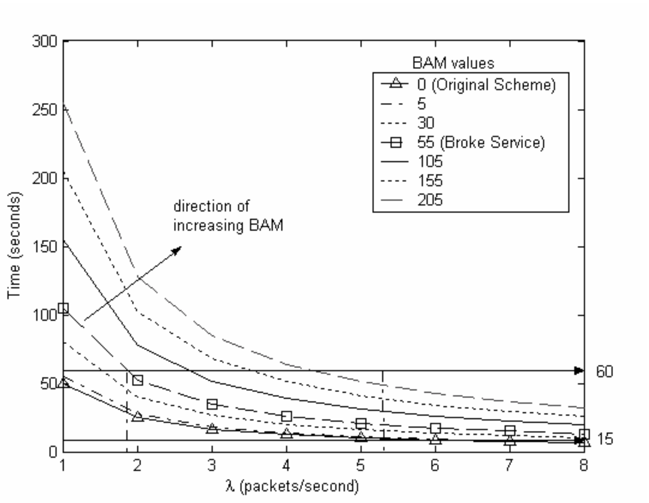

Figure 3: Break-Even Time vs. Traffic Levels for various *BAM* values

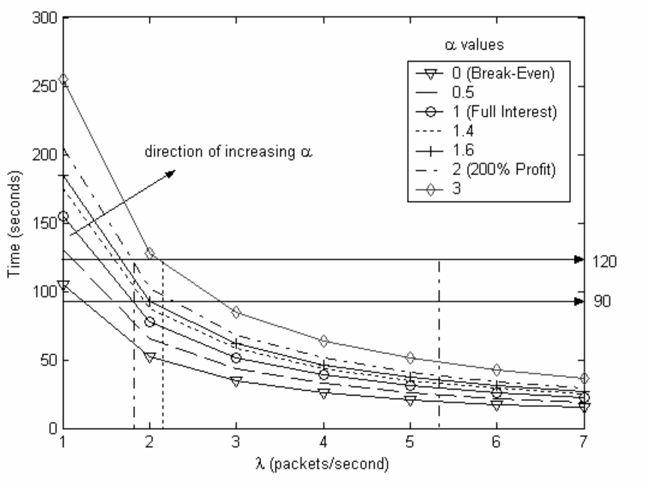

Figure 4: *ROI* Time vs. Traffic Levels for various α values

References

[1] L. Blazevic, C. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y Le Boudec, "Self organization in mobile ad hoc networks: the approach of Terminodes," *IEEE Communications Magazine*, vol. 39, no. 6, June 2001.

[2] L. Buttyan and J.-P Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks", *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

[3] S. Marti, T. Giuli, K. Lai, and M. Baker. "Mitigating routing misbehavior in mobile ad hoc networks". In *Proc. Of the 6th Ann. Int'l Conf. on Mobile Computing and Networking (Mobicom '00)*, August 2000.

[4] S. Buchegger and J.-Y Le Boudec. "Performance Analysis of the CONFIDANT Protocol (Co-operation of Nodes – Fairness in Distributed Ad-Hoc Networks)". In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, Switzerland, June 2002.

[5] S. Zhong, J. Chen, Y. R. Yang, "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks", *Proceedings of IEEE INFOCOM 2003*.