RTC: A Reputation-based Technique for Cooperation Enforcement in Ad Hoc Networks based on TCP Congestion Control Mechanism

HALA ASSAL
Arab Academy for Science, Technology & Maritime Transport
Dept. of Computer Engineering
Abo Kir, Alexandria
EGYPT
assal.h@gmail.com

MOHAMAD ABOU EL-NASR
Arab Academy for Science, Technology & Maritime Transport
Dept. of Computer Engineering
Abo Kir, Alexandria
EGYPT
mnasr@aast.edu

AHMED ABOU EL FARAG
Arab Academy for Science, Technology & Maritime Transport
Dept. of Computer Engineering
Abo Kir, Alexandria
EGYPT
abouelfarag@aast.edu

Abstract: Multihop ad hoc networks were designed assuming trustworthiness and cooperation between all participating nodes; nodes in an ad-hoc network rely on one another in packet forwarding, detecting routes, etc. Thus, the performance of an ad hoc network degrades considerably with the presence of misbehaving nodes. Nodes' misbehaviour includes, but is not limited to, refusing to relay data packets, control packets or even sending false information. In this paper we focus on selfish nodes, which are more concerned about saving their resources than they are about the network's good. We are proposing a reputation-based technique that punishes selfish nodes, by probabilistic dropping of packets destined to such nodes based on their reputation. Punishment is done to encourage selfish nodes to cooperate in relaying data packets, and thus our technique decreases the overall percentage of packet loss in the network by saving packets from being selfishly dropped.

Key–Words: Cooperation, multihop ad hoc networks, packet forwarding, punishment, reputation, selfish nodes

1 Introduction

The field of wireless ad hoc networks has become more and more popular over years due to the noticeable improvements in wireless technologies [1]. Multihop ad hoc networks are infrastructureless and rely on their nodes to maintain the network and perform network operations [2]. A source communicates with a destination, out of its wireless range, through the forwarding process done by relay nodes, which are not directly benefiting from this process [3]. Thus, there are motivations for the existence of misbehaving nodes. We categorize nodes' according to their incentives for misbehaving to *selfish* and *malicious* nodes.

We define a *selfish* node as a node that does not fully cooperate in forwarding data packets aiming to save its resources. On the other hand, a *malicious* node is one that aims to disturb the network's regular behaviour causing partial or local damage, without concern about its resources. In this paper we are focusing on encouraging selfish nodes to participate in the forwarding process and decreasing the percentage of packet loss in the network. We will name our technique RTC.

The rest of this paper is organized as follows: In section 2, we discuss briefly the two main cooperation enforcement technique categories and present different techniques in each category. In section 3, we specify the assumptions we have made, and explain our strategy in section 4. In section 5, we present our simulation scenarios and results and we compare RTC to DARWIN [4]. Finally, in section 6, we give a brief conclusion that summarizes our technique and results, as well as, a brief discussion of possible future work.

2 Literature Survey

Generally, cooperation enforcement techniques fall under two categories: reputation-based and credit-based techniques[5, 4, 6].

2.1 Reputation-based techniques

The behaviour of a node is monitored and quantified through a *reputation value*. A node's reputation increases, when it participates cooperatively in relaying packets. Some techniques base their routing decisions [7, 8] and/or punishing an uncooperative node on the node's reputation value. Decisions can be made either with *first-hand reputation* information, or *first- and* *second-hand reputation* information. Nodes that base their decisions on first-hand information, rely only on their own observations. While nodes that use first- and second-hand reputation rely on routing information messages, that they received from their neighbours, as well as their own observations.

J. J. Jaramillo et al., in [4], propose a reputation-based cooperation enforcement technique based on game theory concepts. They used the Contrite Tit for Tat [9] to model the reputation game. In order to avoid retributive actions a node may take against other nodes if it was falsely identified as selfish, DARWIN does not rely on perfect estimation of the probability that a node forwarded a packet or not. DAWRIN was designed to be collusion resistant, whereas SORI [10] was not. SORI has three basic components:

- *Neighbour monitor* Each node monitors its neighbours' packet forwarding behaviour, and uses a metric, named, *confidence* metric to show how confident it is about its judgment on each one of its neighbours. In the implementation of SORI, the *confidence* metric was the number of packets the monitored node was asked to relay by the monitoring node, i.e. the more the node asks its neighbour to forward packets, the more confident it is of its judgment of its neighbour.
- *Reputation propagation* Nodes forward reputation information to their neighbours, so that a misbehaving node is punished by all its neighbours, not only the ones who got hurt from its selfish behaviour.
- *Punishment* Probabilistic dropping of packets destined to a selfish node is done when it's reputation is below a pre-defined threshold.

CONFIDENT [8] is deployed on the network layer, as opposed to [4] and [10] on the link layer. It is an extension to the Dynamic Source Routing (DSR) [11] protocol that aims to making cooperation more attractive by detecting and isolating misbehaving nodes. It is considered reputation-based because a node takes routing decisions based on the routing and forwarding behaviour of other nodes. CONFIDENT has the following components (on each node):

- *Monitor* A node employs neighbourhood watch to detect anomalies.
- *Reputation system* Each node maintains a list of local node ratings for each one of its neighbours, which could be exchanged with its altruistic neighbours.
- *Path manager* It is responsible for deleting paths that contain misbehaving nodes, re-ranking paths according to the reputation of nodes in the path, and deciding what should be done when a misbehaving node requests a path, or an altruistic requests a path containing misbehaving nodes.
- *Trust manager* It is responsible for sending warning of misbehaving nodes.

2.2 Credit-based techniques

Packet forwarding is treated as a commercial deal, with a form of virtual money. Every time a nodes forwards a packet it receives payment, which it could later spend when sending its own packets. These techniques require a form of tamper-resistant hardware or a virtual bank (trusted third party).

SMART [12] was developed for delay-tolerant networks, where intermediate nodes should perform opportunistic data forwarding for bundles of in-transit messages. It does not need a tamper-proof hardware, as it relies on a virtual bank. It uses a multilayer coin, where the sender indicates, for example, the credit value on the base layer. Intermediate nodes append their digital signatures on other layers. When the virtual bank receives collected layer coins from nodes, it calculates the credit for each node and charges the bundle senders, thus SMART does not require the sender to take part in the transferring or distribution of credit. SMART's security aspect is to secure the coin.

Luzi Anderegg et al. proposed a game theoretic approach for routing that works on top of DSR that tries to find the most efficient route based on energy efficiency in [13]. They calculate the total energy of a path by summing up the emission levels used at the source and the intermediate nodes in that path. [13] uses the *cost-of-energy* as the cost parameter. A node receives payments for every watt it consumes in the forwarding process. If a node cannot get paid enough, it is free to refuse forwarding the packet. Ad hoc VCG technique may fail if the network contains more than one misbehaving node and in the presence of colluding nodes.

To sum up, reputation-based techniques perform punishment, or routing decisions in case it was a network layer technique, based on the node's reputation. The more a node participates in the network, the better reputation it has. Thus, in order to avoid being punished, nodes are motivated to participate in network operations. On the other hand, credit-based techniques, consider communication in the network as a business deal. Whenever a node wants to send a message to a destination outside its wireless range, it pays relay nodes some kind of virtual currency or points. A relay node is motivated to forward packets in order to get points that it could later use when it wants to send a packet to a certain destination. Accordingly, since reputation-based techniques perform punishment, they enforce cooperation, while on the other hand, credit-based techniques are motivational techniques. In credit-based systems, a node could simply declare that it would not participate in the forwarding process if it considers the payment less that it needs.

3 Assumptions

The following assumptions were made in developing RTC:

- Selfish nodes are uncooperative when it comes to relaying data packets, as long as they do not directly benefit from such packets.
- A node may be selfish in terms of saving its resources, but it is not malicious.
- There are no colluding nodes.
- All nodes in the network have the desire to communicate, thus they care about *not* being isolated from the network.
- Nodes operate in promiscuous mode in order to perform behavioural watch.
- Nodes are able to send information about selfish nodes to their neighbours.

We also assume that selfish nodes are detected via a detection algorithm [14], and we use this as an input to RTC for correcting the behaviour of these nodes.

4 Proposed Strategy

4.1 TCP congestion control

Since, RTC is based on TCP congestion control algorithm, we will present a brief background on it and shed the light on how our technique's concept map to those of TCP congestion control algorithm. TCP employs congestion control to avoid congested links and to provide fairness in distributing the bandwidth, of a shared link, among different TCP connections [15]. TCP congestion control is referred to as an

Table 1: Mapping RTC's concepts to TCP congestion control

Point of comparison	**TCP congestion control**	**RTC**
On occurrence of loss/ dropping	Decrease sending rate multiplicatively	Decrease selfish node's reputation multiplicatively
On possible improvement	Increase sending rate multiplicatively	Increase selfish node's reputation additively
Indication of improvement	Congestion window reaches a $threshold$	% of dropped packets reaches a $Threshold$ $\beta_{i,e} \geq \Omega * \beta_i C$
On reaching the indication of improvement	Increase sending rate additively	Fast recovery by increasing node's reputation multiplicatively

additive-increase, multiplicative-decrease (AIMD) algorithm. The sender increases its sending rate linearly when the end-to-end path is congestion free. When a loss event occurs, it implies that the path is congested, the sender, then, decreases its rate multiplicatively. This behaviour develops a sawtooth pattern. After a loss event perceived through timeout, when the sender perceives that the network has recovered and has the capability of sending at least some segments, TCP Reno [16] increases the sending rate multiplicatively. Whereas, if the loss event was perceived through three duplicate acknowledgments, it increases the sending rate additively. Table 1 shows the mapping of RTC's concepts to TCP congestion control, especially TCP Reno [16]. In RTC, possible improvement is when an a selfish node cooperatively relays a packet. While, possible improvement in TCP congestion control is indicated by receiving an acknowledgment for a packet that was already sent. (Variables used in this table are explained in details later in this section.)

4.2 RTC

Because the cooperation between nodes in an ad hoc network highly affects the network's performance, it is important to encourage selfish nodes to participate in the forwarding process. We advocate the punishment approach to achieve this goal. First, a node is identified as selfish, then it is punished for behaving selfishly. Our contribution is in the punishment of selfish nodes. However, punishment should be confined by two conditions:

- Selfish nodes react positively when punished and become active participants in the forwarding process.
- Network performance is not negatively affected by punishment-based packet dropping done by altruistic nodes.

We propose a reputation-based technique that punishes selfish node i by probabilistic dropping of packets destined to node i, based on its $ReputationFactor$ (α_i). On the occurrence of an event e, where a selfish node is asked to relay data packets, we monitor the behaviour of the selfish nodes. Inspired by the Additive Increase Multiplicative Decrease (AIMD) behaviour of TCP congestion control algorithms [17, 16], on the occurrence of e, an altruistic node updates the $ReputationFactor$ of the selfish node based on the following:

- **Multiplicative Decrease** When selfish node i selfishly drops a packet, it's $ReputationFactor$ is decreased multiplicatively by a rate ωdm, where $\omega dm < 1$ (1).
- **Additive Increase** When selfish node i is starting to behave properly and participate in the forwarding process, it's $ReputationFactor$ is increased additively by a rate ωil, where $\omega il < 1$ (2).
- **Multiplicative Increase** When a selfish node i continues to behave altruistically, it's $ReputationFactor$ is increased multiplicatively by a rate ωim, where $\omega im < 1$ (3).

$$\alpha_{i,e} = \omega dm * \alpha_{i,e-1} \tag{1}$$

$$\alpha_{i,e} = \omega il + \alpha_{i,e-1} \tag{2}$$

$$\alpha_{i,e} = \omega im * \alpha_{i,e-1} \tag{3}$$

At each event e involving selfish node i, altruistic nodes calculate the % of packets dropped by node i ($\beta_{i,e}$) (4). When i starts to behave properly and participates in forwarding data packets, altruistic nodes save the value of ($\beta_{i,e}$) at the current event e, where selfish node i started to cooperate in relaying data packets. Node i is continuing to behave properly when its $\beta_{i,e}$ reaches the previously stored $\beta_i C$ multiplied by a $correctingFactor$ (Ω). This the $Threshold$ (5) for beginning the *fast recovery* phase, where altruistic nodes reward node i by increasing its $ReputationFactor$ multiplicatively (3) for faster recovery of reputation.

$$\beta_{i,e} = \frac{\text{\# of packets dropped by } i\ (d_i)}{\text{total \# of packets } i \text{ was supposed to relay } (N_i)} \tag{4}$$

$$Threshold = \Omega * \beta_i C \text{ , where } \Omega < 1 \tag{5}$$

The flowchart for RTC, shown in Figure 1, mentions a $correctingFlag$, this is Set when node i starts to forward data packets, and is $Reset$ if it behaves selfishly again. A node that performs relatively small number of droppings will almost *not* be punished by our technique. Such infrequent dropping will be deemed as noise. One advantage of such feature is taking into consideration droppings due to any other factors that are irrelevant to deliberate dropping due to selfishness, such as network layer dropping. However, if network layer droppings were extremely large, it could negatively affect the decisions of our technique, increasing the number of false positives.

RTC, relies on first- and second-hand reputation information, and assumes that nodes exchange reputation information about selfish nodes with their neighbours. The aim of this paper is to introduce a reputation-based punishing technique that efficiently encourages selfish nodes to cooperate in forwarding data packets. Accordingly, we consider the exchange of reputation information between nodes a separate problem that does not directly affect the main aim of RTC. The frequency of exchanging reputation information messages and the overhead this incurs on the network will be studied in future extension to RTC.

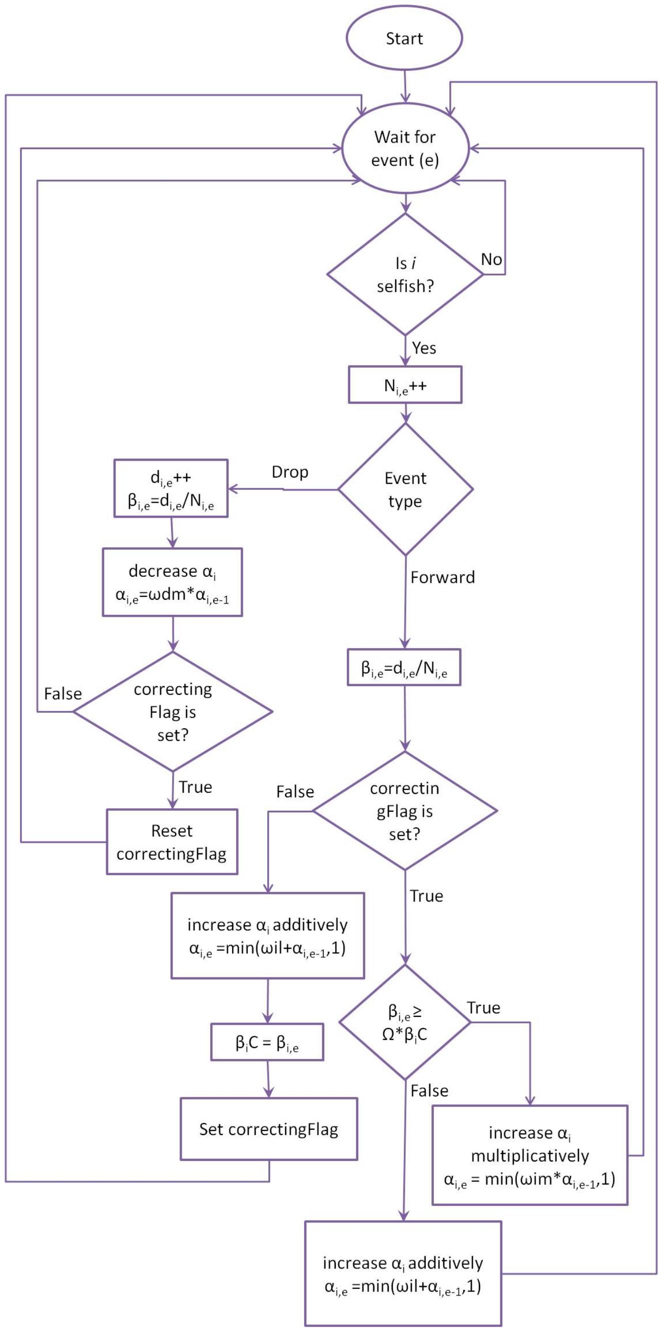

Figure 1: RTC flowchart

5 Simulation

In this section, we present the results obtained from employing RTC on a multihop wireless ad hoc network, where selfish nodes exist. Our aim is to encourage selfish nodes to cooperate in the packet forwarding process, and hence to decrease the percentage of packet loss, due to selfishness, in the network. We also compare RTC to DARWIN [4].

5.1 Settings

We implemented our technique on OMNeT++ v2.4 [18]. DSR [11] was used on the network layer, while at the link layer IEEE 802.11 Distributed Coordination Function (DCF) was used. We simulated a network of 100 nodes and performed simulations according to 9 scenarios, each scenario had a different percentage of selfish nodes in the network. We started the first scenario with 5% selfish nodes, and increased the percentage of selfish nodes by 5% in each scenario, to reach 45% selfish nodes. Each scenario was run 10 times for 5 simulation seconds each, with a different set of selfish nodes in each run.

In RTC, we assume, supported by simulation results, that network layer packet loss is much less than packet loss due to selfishness. Recall that relatively small number of droppings will not disturb the smooth operation of RTC.

5.2 Results

Figure 2, shows the similarity of the behaviour of the $ReputationFactor$, of a randomly picked example selfish node 6, to the TCP congestion control sawtooth pattern [17, 16]. Recall that with RTC, when a node selfishly drops a packet, its $ReputationFactor$ is decreased multiplicatively. Whereas, it is increased cautiously with an additive rate, when it participates in relaying packets. Upon reaching a certain threshold ($\Omega\beta_i C$), a *fast recovery* phase begins. During this phase, as long as the selfish node is continuing to relay packets, its $ReputationFactor$ is increased multiplicatively. Eventually, the selfish node's $ReputationFactor$ would reach 1, and would remain as such, as long as packet relaying continues. If the node returns back to its selfish behaviour and drops a packet, it will be punished by decreasing its $ReputationFactor$ multiplicatively.

Figure 3, is a plot of the percentage of packets dropped by an example selfish node 6, with and without employing RTC. From this figure it is clear that selfish nodes react positively to our punishment technique and are encouraged to participate in the forwarding process. The percentage of packets dropped by selfish node 6 is decreased by, approximately, 33%. Accordingly, RTC showed to be successful in encouraging selfish nodes to be more cooperative while complying to the rules of punishment mentioned in section 4.

Figure 4, shows the relation between the node's $ReputationFactor$ and the percentage of packets dropped by this node. Notice that the

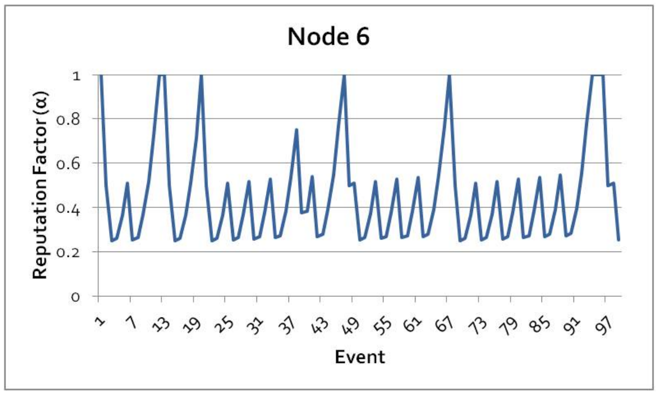

Figure 2: Behaviour of $ReputationFactor$

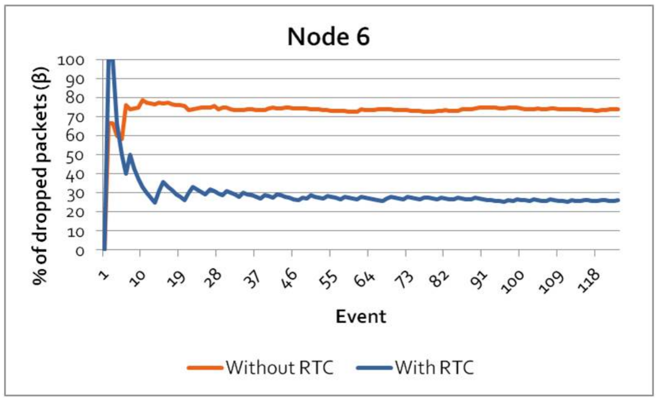

Figure 3: % of dropped packets, with and without RTC

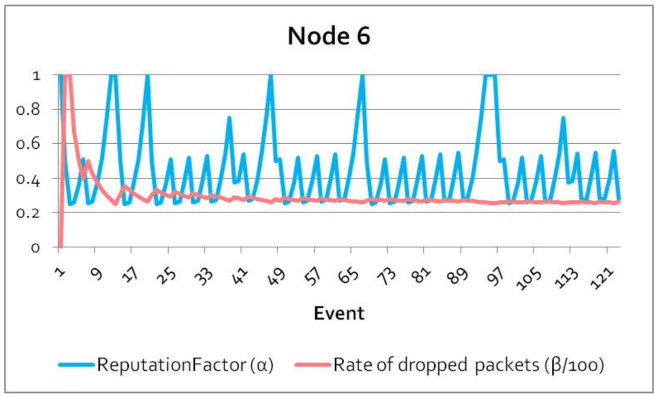

Figure 4: Relation between a node's $ReputationFactor$ and the % of packets it drops

$ReputationFactor$ of selfish node 6 remains fluctuating as long as it is not 100% cooperative. A node's $ReputationFactor$ does not reach 1 except when it is fully cooperative, i.e. it does not perform selfish dropping of packets.

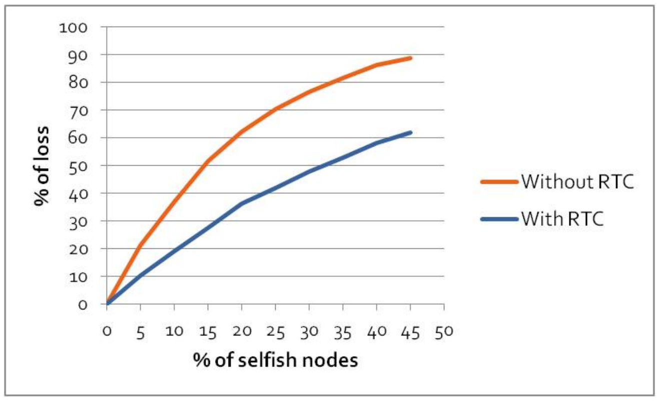

Figure 5: Relation between the % of selfish nodes and the % of packet loss in the network

Figure 5, shows the relation between the percentage of selfish nodes in the network and the percentage of packets lost due to selfishness. As the percentage of selfish nodes increases, the percentage of packet loss increases. The overall percentage of packet loss due to selfishness is calculated before and after employing RTC. Figure 5, shows that although altruistic nodes perform packet dropping for the sake of punishing selfish nodes, the overall percentage of packet loss is improved when employing RTC. Again, this complies with the rules of punishment we set in section 4.

To have an insight on the improvement in packet forwarding we plotted Figure 6 to show the total number of packets relayed in the presence of selfish nodes, with and without employing RTC. An interesting observation is that as the percentage of selfish nodes increases, the difference between the number of packets relayed without employing RTC and with employing it increases. Our explanation is that as the percentage of selfish nodes increases, their negative effect on the network increases, and the number of packets lost due to selfishness increases. Employing our strategy encourages selfish nodes to behave more cooperatively, thus a number of packets are forwarded (that would have been selfishly dropped without RTC) relatively proportional to the percentage of selfish nodes in the network. Thus, as the percentage of selfish nodes increases, the amount of packets salvaged from selfishness increases. R gives acceptable results in the presence of both a small and a relatively large number of selfish nodes in the network.

We measure the effect of increasing the percentage of selfish nodes in a network on the percentage of packets forwarded. Figure 7 shows the percentage of

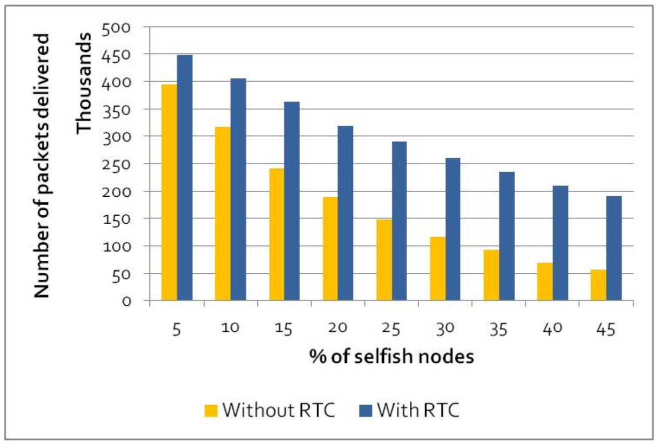

Figure 6: Relation between the number of packets delivered and the % of selfish nodes

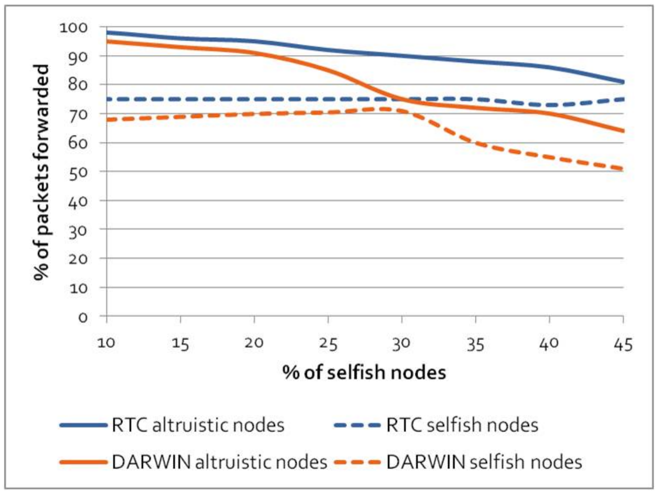

Figure 7: Comparing RTC to DARWIN [4]

packets forwarded by selfish and cooperative nodes, and compares them to those obtained by DARWIN. RTC showed better improvement for both cooperative and selfish nodes than DARWIN's. With RTC, the percentage of packets forwarded by altruistic nodes remains above 90% in a network with up to 30% selfish nodes. In a network with more than 30% of its nodes acting selfishly, the percentage of packets forwarded by altruistic and selfish nodes converges with the increase in the percentage of selfish nodes. This implies that altruistic nodes are not overused and that our technique provides fairness in the usage of resources between nodes.

6 Conclusion and Future Work

In this paper we introduce a reputation-based cooperation enforcement technique for wireless multihop ad hoc networks. RTC punishes selfish nodes to encourage them to cooperate in the forwarding process. Punishment is done, based on the selfish node's reputation, by probabilistic dropping of packets destined to the selfish node. Although altruistic nodes perform packet dropping to punish selfish nodes, the overall percentage of packet loss in the network is decreased. This implies that our technique successfully motivates selfish nodes to act more cooperatively in forwarding data packets. When comparing RTC to DARWIN [4], it is shown that RTC is more effective in punishing selfish nodes, while at the same time it does not degrade the percentage of packets forwarded by altruistic nodes as much as DARWIN does.

References:

[1] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: challenges and directions," *Communications Magazine, IEEE*, vol. 40, no. 5, pp. 20 –22, may 2002.

[2] R. Hekmat, *Ad-hoc Networks: Fundamental Properties and Network Topologies*. Springer, 2006.

[3] A. AbdelRahman, M. El-Nasr, and O. Ismail, "A novel forwarding/dropping decision engine for wireless multi-hop ad-hoc networks," in *Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on*, nov. 2009, pp. 1 –5.

[4] J. J. Jaramillo and R. Srikant, "Darwin: distributed and adaptive reputation mechanism for wireless ad-hoc networks," in *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, ser. MobiCom '07. New York, NY, USA: ACM, 2007, pp. 87–98. [Online]. Available: http://doi.acm.org/10.1145/1287853.1287865

[5] K. Balakrishnan, J. Deng, and V. Varshney, "Twoack: preventing selfishness in mobile ad hoc networks," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 4, march 2005, pp. 2137 – 2142 Vol. 4.

[6] G. F. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas, “Cooperation enforcement schemes for manets: a survey,” *Wireless Communications and Mobile Computing*, vol. 6, no. 3, pp. 319–332, 2006. [Online]. Available: http://dx.doi.org/10.1002/wcm.398

[7] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom ’00. New York, NY, USA: ACM, 2000, pp. 255–265. [Online]. Available: http://doi.acm.org/10.1145/345910.345955

[8] S. Buchegger and J.-Y. Le Boudec, “Performance analysis of the confidant protocol,” in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking &amp; computing*, ser. MobiHoc ’02. New York, NY, USA: ACM, 2002, pp. 226–236. [Online]. Available: http://doi.acm.org/10.1145/513800.513828

[9] R. Boyd, “Mistakes allow evolutionary stability in the repeated prisoner’s dilemma game,” *Journal of Theoretical Biology*, vol. 136, no. 1, pp. 47 – 56, 1989.

[10] Q. He, D. Wu, and P. Khosla, “Sori: a secure and objective reputation-based incentive scheme for ad-hoc networks,” in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 2, march 2004, pp. 825 – 830 Vol.2.

[11] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, ser. The Kluwer International Series in Engineering and Computer Science, T. Imielinski and H. F. Korth, Eds. Springer US, 1996, vol. 353, pp. 153–181, 10.1007/978-0-585-29603-6_5. [Online]. Available: http://dx.doi.org/10.1007/978-0-585-29603-6_5

[12] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen, “Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks,” *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 8, pp. 4628 –4639, oct. 2009.

[13] L. Anderegg and S. Eidenbenz, “Ad hoc-vcg: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents,” in *Proceedings of the 9th annual international conference on Mobile computing and networking*, ser. MobiCom ’03. New York, NY, USA: ACM, 2003, pp. 245–259. [Online]. Available: http://doi.acm.org/10.1145/938985.939011

[14] M. Just, E. Kranakis, and T. Wan, “Resisting malicious packet dropping in wireless ad hoc networks,” in *Ad-Hoc, Mobile, and Wireless Networks*, ser. Lecture Notes in Computer Science, S. Pierre, M. Barbeau, and E. Kranakis, Eds. Springer Berlin / Heidelberg, 2003, vol. 2865, pp. 151–163, 10.1007/978-3-540-39611-6_14. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-39611-6_14

[15] J. F. Kurose and K. W. Ross, *Computer Networking. A Top Down Approach*, 4th ed. Pearson Education inc., 2008.

[16] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, “Modeling tcp reno performance: a simple model and its empirical validation,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000. [Online]. Available: http://dx.doi.org/10.1109/90.842137

[17] V. Jacobson, “Congestion avoidance and control,” *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, Aug. 1988. [Online]. Available: http://doi.acm.org/10.1145/52325.52356

[18] Omnet++. [Online]. Available: http://www.omnetpp.org/