MAC Layer Misbehavior on Ad Hoc Networks

N. Nagel, R. Shokranian, and J. L. Bordim

University of Brasilia, Campus Universitário, Asa Norte, 70910-900 Brasilia, DF, Brazil

Summary
One of the major challenges in ad hoc networks is to ensure nodal collaboration. Nevertheless, collaboration may lead to undesired results when nodes can exploit their siblings for their own benefits. Until recently, the research community focused on ensuring nodal cooperation at the network layer. In this work we focus on node behavior at the Medium Access Control (MAC) sublayer. More specifically, in this work focus on mechanisms to evaluate nodal activity and verify whether a node is adhering to the protocol rules or not. We show that misconduct at the MAC layer can have serious implications in terms of throughput. Our main contribution is a mechanism that enables nodes to monitor neighboring activity and assess their conduct based on the observed data.

Key words:
Misbehavior, Collaborative Computing, Trust, Reputation, Ad hoc networks.

1. Introduction

Wireless networks have grown and developed in the last decades. They can now be found in offices, hospitals, and in the houses of millions of people. Although they are present in our daily life, most of them are dependent of a centralized infrastructure. Ad hoc networks are formed by mobile autonomous devices that are capable of configuring among them to form a network without the help of a fixed infrastructure. The creation of a network became easy and has a low cost. Ad hoc networks also demand for more specialized routing algorithms and efficient means to deal with medium access, which requires elaborated solutions.

Ad hoc networks inherits most of the traditional wireless networks problems, such as interference, low reliability, low bandwidth, high influence of the environment for the correct functionality of the network, limited resources in terms of battery and processor power, and low service coverage [3]. Since the transmission distance of the nodes is limited, communication among nodes which are outside the radio range of each other is possible only if there is cooperation. In such context, cooperation means that each node must relay data to other nodes, which implies in more battery and processor power usage. As the resources are limited, cooperation can be expensive, and this can cause some nodes to have a selfish behavior. A selfish node may cooperate when it is somehow rewarded. Such nodes are also termed as *misbehaving* nodes. A misbehaving node uses the knowledge of the underline protocols for their own benefits. In a cooperative environment, such behavior can have serious impacts on the entire network [11]. Thus, it is important to identify misbehaving nodes and define mechanisms to prevent and deal with them. It is worth noting that a *malicious node*, in contrast, uses the knowledge of the underline protocols for wrongdoing actions.

The ad hoc network dynamics and the lack of infrastructure discard the use of centralized mechanisms for control access, authentication, or even traffic control. Such schemes, must therefore, be implemented in a distributed way. There are many proposals that use cryptography for safe routing. However these proposals are limited and do not consider the lack of infrastructure and resources [10]. Another option is the use of reputation and trust systems, similar to Ebay [9]. These schemes can be applied to ad hoc networks to prevent misbehavior and stimulate node cooperation [2, 4, 8, 13]. However, reputation systems need to be robust to identify and prevent network service degradation.

Until recently, the research community was concerned in observing nodal behavior at the network layer. This was achieved by checking whether or not a node was correctly forwarding packets [6]. However, a misbehaved node can act not only in the network layer but also in other layers. This paper is focuses on studying this problem of misbehavior at the *Medium Access Control* (MAC) sublayer. When a node wishes to use the channel, it must wait a random time before transmitting, known as *backoff*. This period is generated randomly and independently for each node with the aim to prevent collisions. It is possible that a misbehaving node does not respect the minimum *Contention Window* (*CW*) and, as a result, transmits more than the expected. Although there are other works in this direction, as it shall be discussed in Section 2, none of them deal with the amount of information a node has to gather in order to have a higher confidence in classifying a neighboring node as a misbehaving node. In other words, this work attempts to estimate the amount of data one has to collect in order to better judge its neighbors' conduct. Obviously, the more data one collects the better the accuracy will be. In this work we are interested in answering the following question. *What is the minimum*

Manuscript received November 5, 2009
Manuscript revised November 20, 2009

amount of information one should gather, in order to have high confidence, to characterize a sibling as a misbehaving one? This work gives insights in this direction.

The reminder of this work is organized as follows: Section 2 presents the related works. Section 3 details the model assumed here and underlines techniques than can help identifying misbehaving nodes. Section 4 shows some simulations results based on the techniques presented earlier and Section 5 concludes this work.

2. Preliminaries

A network is usually represented by a graph $G=(V, E)$, where V is a set of vertices, and E is a set of edges between the vertices $E \subseteq \{(u,v) \mid u, v \in V\}$. Throughout this work, the number of nodes in the network is denoted by N, *i.e.*, $N=|V|$, and $N[v]$ represents the number of nodes adjacent to node v. In what follows, we present some details of the IEEE802.11 DCF [1] operation and present related works.

2.1 The IEEE802.11 DCF Operation

The IEEE802.11 [1] *Distributed Coordination Function* (DCF) is a contention based MAC protocol. The *request to send* (RTS) and *clear to send* (CTS) exchange is used to inform neighboring nodes about the eminent start of communication. Each of these packets contains the proposed duration of communication and the destination address. Neighboring nodes that overhear any of these packets must themselves defer communication for the proposed duration. This mechanism is known as *Virtual Carrier Sensing* and is implemented through the use of the *Network Allocation Vector* (NAV) variable. DATA and ACK packets follow the successful RTS/CTS exchange.

When the MAC protocol at a node *s* receives a request to transmit a packet, both physical and virtual carrier sensing are performed. If the medium is found idle for an interval of DIFS (DCF *Inter Frame Space*) time, then *s* chooses a random backoff period for additional deferral. When the backoff period expires (*i.e.*, reaches zero), *s* transmits the packet. If a collision occurs, a new backoff interval is selected. A *Short Inter Frame Space* (SIFS) is used to separate transmissions belonging to the same dialog.

2.2 Related Work

The task of relaying a message in an ad hoc network has a high cost, both in terms of battery and processing power [6]. As such resources are limited; a node cooperates with other nodes presuming that it will receive the same treatment when necessary. This assumption is valid in a collaborative environment and serves as motivation for a node to forward other node's packets. However, other mechanisms could also be used to ensure cooperation. Proposals based on credits, such as those in [5] and [15], try to incentive nodal cooperation. The idea is to use a type of credit, or virtual payment, to reward a node for its service, that is, a node receives a virtual payment to forward other nodes' packets.

Another way of ensuring nodal cooperation is to select nodes which are prone to collaborate. However, for this idea to work, one has to know whether or not a node is keen to collaborate. This can be achieved via a *reputation system*, which works by monitoring neighboring activities and classifying them as cooperative or not. Cooperative nodes are then selected to perform tasks, such as routing. The works in [4, 8] and [13] are example of reputation systems. The scheme proposed in [13] enforces cooperation by restricting selfish nodes to obtain certain resource. The protocol is based in a monitoring mechanism combined with a reputation table present at each node. The scheme in [8] tries to establish trust among devices which are willing to cooperation. Here, the reputation of a node is evaluated based on its behavior when forwarding packets. In [4], each node is composed of four components: monitor, trust manager, reputation manager, and routes manager. The monitor observes the transmission channel and identifies packets detour. When this happens the reputation manager is called. The trust manager sends an alert message to warn the existence of a malicious node. Each alert received is filtered to verify the trust level of the alert message.

The above works attempt to evaluate the reputation on a given node based on its behavior. However, these works are focused on events generated at the network layer. The work proposed in [4], termed DOMINO, is a misbehavior detection system centered at the MAC layer events. Basically, its main goal is to detect and identify stations that increase their bandwidth by changing some protocol properties. The system has two stages: *data collection* and malicious *nodes identification*. The first stage is done in regular time periods where traffic from the stations is collected. Based on the collected information, the second stage verifies whether a node is misbehaving or not. A number of tests are used to evaluate behavior of a node, such as:

i. Number of successful transmissions and overall number of collisions;
ii. Average backoff time observed for each neighboring node;

iii. Respect to frame space intervals (such as *DCF Inter Frame space* - DIFS);
iv. Bogus information on RTS/CTS frames to set the Network Allocation Vector longer than necessary.

DOMINO is tailored for *infra-structured mode,* that is, it considers the existence of an access point where the detection system is present. When extending DOMINO to an ad hoc setting, a number of problems have to be dealt with. For instance, in the first test listed above, a node can be misclassified, as the characteristics of the IEEE802.11 traffic pattern is known to be busty. That is, a node may send several packets when it gains channel access. This conduct, however, does not characterize misbehavior. Also, in the presence of interferences, it is quite difficult to estimate the backoff of a neighboring node. Furthermore, assuming that the purpose of misbehaving node is to take advantage over other nodes to increase its throughput, a node would gain nothing by reserving more time than necessary for its transmission. Hence, increasing the NAV would just make other stations silent.

As mentioned before, in this work we are interested in knowing the amount of evidence that a node has to collect in order to characterize a sibling as a misbehaving one. Surely enough, to attain this goal, a certain amount of information must be collected. Here, we are interested in establishing a direct link between the collected data and the misconduct level for a given node. As mentioned before, this work attempts to estimate the amount of data one has to collect in order to better judge its neighbors' behavior. To the best of our knowledge, none of the previous works have addressed this problem. Also, in this work we propose a scheme to collect evidence that a node is misbehaving. Such information is gathered at the MAC sublayer. The next section details how this can be achieved.

3. Identifying Misbehaving Nodes

In this section we first show how a node can effectively collect information about its siblings without resorting to complex schemes that can have major costs in terms of battery usage, memory, communication or processing power. As mentioned earlier, this work focuses in trying to identify selfish nodes at the MAC sublayer. More precisely, our work focuses on identifying misconduct when a node modifies its *CW* in order to improve its chance to gain channel access.

After a successful transmission, a node would select a backoff value randomly from zero up to *CW*-1. Thus, by selecting a lower value, a node is likely to wait a shorter period of time to attempt to gain channel access. However, due to the characteristics of the CSMA/CA, a node may obtain channel access consecutive times, which does not characterize misbehavior. In other words, even a node that adhere to the protocol rules, over a certain period of time, may gain channel access more than others. Indeed, the IEEE802.11 has such burst characteristic. Hence, we are interested in knowing the limit at which a node's conduct is within the protocol parameters and when it's above. Here we focus on ad hoc networks and our scheme attempts to preserve, as much as possible, the node's resources such as battery and processing power.

Trying to estimate a neighboring station backoff, as performed in [4], can have a high cost. Furthermore, a malicious node may intentionally disrupt control or data packets, forcing its neighboring nodes (transmitting nodes) to increase their contention windows after recovering from collisions. In such case, the backoff used by the malicious node may even adhere to the protocol. Nevertheless, the throughput of the malicious node will be higher. Our scheme explores this idea. Thus, in this work, we observe the throughput of the neighboring nodes and from that we estimate the neighboring stations backoff. In our approach, each node observes the activity of its neighbors. More precisely, each node collects and record the number of successful transmissions over a period of time. Note that, in order to save battery power, a node may monitor the RTS (and/or CTS) and the corresponding ACK control packets. Based on the collected information, the observing node n would compare the number of successful transmissions for each neighboring node m ($\in$ N[n]). For each neighboring node m, node n would check whether or not the following inequality holds:

$$T_m(t) \geq T_n(t) + S_n(t) \times \delta \tag{1}$$

In Eqn. 1, t represents the current time, $T_m(t)$ is the number of transmissions completed by node m and $T_n(t)$ is the number of transmissions completed by n. To accommodate fluctuations, a threshold is included, which is given by the $S_n(t) \times \delta$, where $S_n(t)$ is the observer standard deviation and δ is the tolerance, ($0 \leq \delta \leq 1$). The tolerance is used to calibrate the threshold. The smaller the tolerance is, the stricter the scheme becomes. The tolerance calibration is important since a low tolerance value can be enough for a given setting but the same value can leave a malicious node undetected in another. Conversely, a high tolerance value can be necessary when it's difficult to detect misbehaving nodes. This, however, may take longer to detect dishonest activity.

Based on the gathered information, the *misbehavior score* (*Ms*) of an observed node m ($\in$ N[n]) is defined as Ms_m, 0

$\leq Ms_m \leq 1$. If the Inequality (1) holds, then $Ms_m(t) = 1$, else $Ms_m(t) = 0$. Obviously, a node may change its behavior over the time. So, classifying a node as a collaborative or non-collaborative is related to the actions that are currently happening. However, if a node misbehaved in the past, and now is collaborating, a way to provide redemption should be devised. Nevertheless, the past actions should be taken into consideration as well. For this purpose we introduce an aging mechanism which is based on a weighted mean of the misbehavior score, which we called *misbehavior* level, computed as shown below:

$$M_n(t) = 2 \times \frac{\sum_{i=0}^{t-1} (i+1) \times Ms_n(i)}{(t+1) \times (t+2)} \qquad (2)$$

The use of an aging mechanism is important as it can provide redemption to nodes that misconduct in the past but are now well behaved. Furthermore, as mentioned above, the CSMA/CA may cause nodes to have transmissions bursts, which may classify them as misbehaved nodes. Eqn. 2, provides means to allow nodes to redeem themselves.

The next section presents the simulations results obtained with the misbehavior classification and identification presented above.

4. Simulation Results

In order to show the performance of our scheme, in this section we evaluate it in a number of settings. For each simulation we have scenarios composed of two, eight, and thirty two nodes. The results are drawn from an average of a hundred simulations, which we call as a simulation *run*. We fixed the amount of time for each simulation as a function on the number of nodes. Thus, a simulation with eight nodes takes four times more than that for two nodes. This work considers ad hoc networks in with the following characteristics:

- The nodes are static;
- Nodes communicate in single-hop;
- The simulations include a single malicious node;
- All nodes are continuously trying to transmit.

Prior to the data transmission, that is, when the node gain channel access, each transmitting node obeys the 4-way handshake with RTS/CTS followed by DATA and ACK packets. In what follows, each node selects its backoff independently from [*CWmin*, *CWmax*], where *CWmin* and *CWmax* are assumed to be in the initial state.

4.1. Successful Transmissions

To evaluate the number of successful transmissions, we try to estimate the *CW* values a malicious node would use to pass undetected and when it would be easily spotted. One important question to answer is how much a misbehaved node could transmit more than a normal node without being considered malicious. For this threshold, we use the standard deviation. However in some cases this threshold could be too large or too small. Thus, our goal is to observe the percentage of successful transmissions by the malicious node as well as by the well-behaved ones.

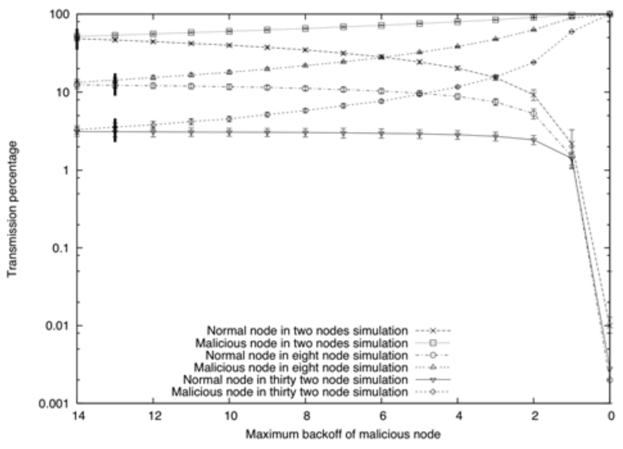

Fig. 1. Channel usage with a varying number of nodes and a single malicious node.

Figure 1 shows the simulation results for two, eight and thirty two nodes in the presence of a single malicious node. The *x*-axis shows the *CWmax* value selected by the malicious node and the *y*-axis shows the averaged number of transmissions for the misbehaved node as well as for the well-behaved nodes. The standard deviation for each point is also shown in the figure. As can be observed, when a malicious node lowers its contention window slightly, it can pass undetected as the number of successful transmissions is very close to the well-behaved nodes. With an increase on the number of nodes, the standard deviation increases as well, which makes it harder to identify malicious activity when the contention window is within bounds of the well behaved nodes. The values at which the malicious node activity becomes evident are shown with a vertical mark in the figure. With thirty two nodes, the malicious node would only be detected if its *CWmax* was set to be less than 12, up to this value, there could misclassification.

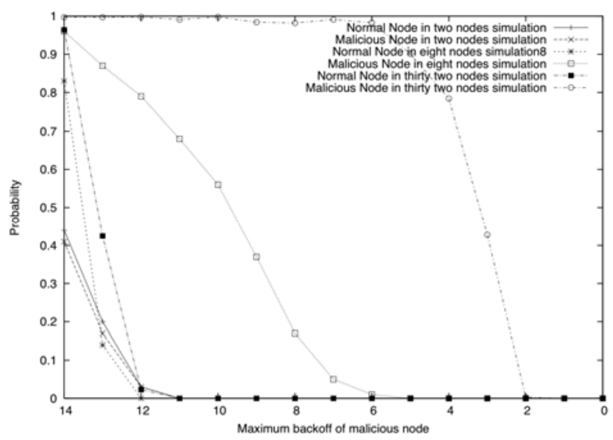

Fig. 2. Probability of successful transmissions in the presence of a malicious node.

In order to better understand the channel usage we have analyzed the number of possible successful transmissions using the binomial distribution. The results are shown in Figure 2. For each point we consider the standard deviation and we use the binomial distribution to calculate the probability for this event to occur in a normal case. If the probability is too low, it means that the node is misbehaving. It is important to note that with the binomial distribution, misbehavior may occur when a node diverges from the mean (above or lower). For instance, with thirty two nodes, as shown in the figure, the averaged number of transmissions of the well-behaved nodes is within bounds until the malicious node selects a *CWmax* below 6. With eight nodes, the malicious node can be detected, with a probability of less than 15%, if it chooses a *CWmax* of 13. This means that if a malicious node is not too greedy, it can pass undetected with high probability. Also, the more nodes are in the vicinity, the more aggressive the malicious node can be without being spotted.

Observing Figure 2, and comparing it with the values obtained in Figure 1, one can see that the malicious node curve reaches a very low value near the point at which the normal nodes and selfish nodes curve start to deviate. By observing these results we can see that the well-behaved nodes can only detect that there is something wrong when the misbehaving node is cheating with a much lower *CWmax* than that used by the other nodes. The identification of misbehavior gets harder with the increase of nodes. In other words, when the behavior of a malicious node deviates slightly, it can take long time to identify it, and the larger the number nodes, the harder it becomes.

4.2. Misconduct Grading

The goal of this section attempts to verify if Equation 2 correctly grades a node based on observed activity. However, when a node starts collecting data, it has little information about its neighbors. Thus, to start, the observer may classify its neighbors as well behaved at the beginning. Here, we have fixed the tolerance δ using values ranging from zero (0) up to 1.0. The results are shown for *CWmax* values of 15, 14, and 13. We are not presenting the results for all possible *CWmax* values of the observed node as lowered values only make the curves to converge faster. Figures 3, 4 and 5, shows the misconduct level for four, eight and thirty two nodes. The *x*-axis shows the simulation time (in seconds) and the *y*-axis shows the malicious level of the observed node.

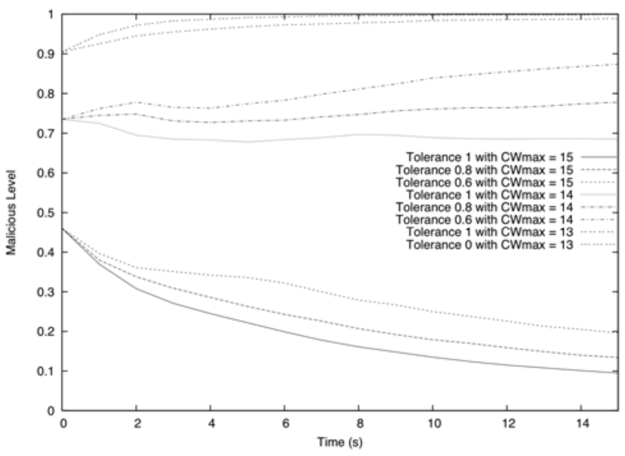

Fig. 3. Misconduct grading with 2 (two) nodes using *CWmax* = 15, 14, 13.

We consider a *CWmax* value of 15 as being a correct backoff value. As can be seen in Figure 3, a *CWmax*=15 makes the curves will converge to zero, showing that the node is well-behaved (for all values of δ). When *CWmax*=14, a lower tolerance (δ=0.6) makes the curves to converge to one quicker than a higher tolerance (δ=1.0). The convergence time can be reduced by decreasing δ. As discussed before, the more a node learns about its environment, the better its classification will be. Thus, as the time goes by and the observer gathers information about its neighboring activity, the malicious level for well-behaved nodes will tend to zero while the curve for a malicious nodes will tend to one. Based on the results and simulations we have conduced, we found that an appropriate value for the tolerance would be as follows:

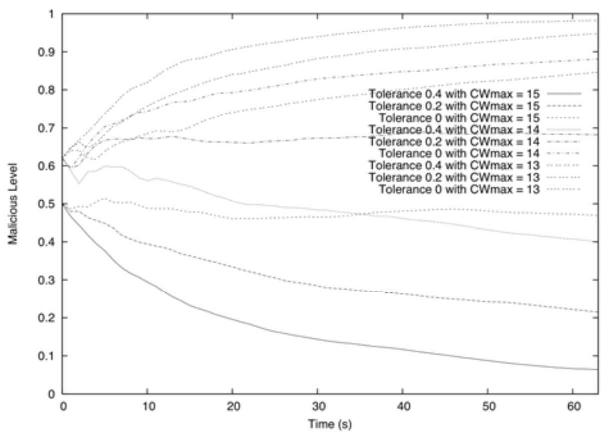

Fig. 4. Misconduct grading with 8 (eight) nodes using *CWmax* = 15, 14, 13.

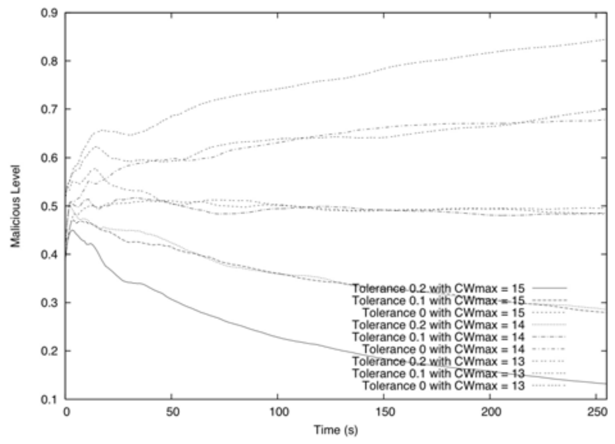

Fig. 5. Misconduct grading with 32 (thirty two) nodes using *CWmax* = 15, 14, 13.

$$\delta(n) = \frac{2C}{\mathrm{N}}, \tag{3}$$

where N is the number of nodes and C is a constant. The constant value will depend on the network parameters, such as number of transmissions the amount of collected data. As can be seen in Figure 3, an appropriate tolerance for two nodes is near 0.8. So for N = 2 we can use δ (2) = 0.8, that will give us that C = 0.8. Using C = 0.8 for the other scenarios with normal communications, would give us the following tolerance values: δ (2) = 0.8, δ(4) = 0.4, δ (8) = 0.2, δ (16) = 0.1, δ(32) = 0.05. If we observe the Figures 3, 4 and 5, we can see that this tolerance values guarantees that normal behavior curves decreases and malicious behavior curves increases as time goes by.

The results show that a higher tolerance may classify borderline nodes as well-behaved (see the case with δ=0.4 and *CWmax* = 14 in Fig. 3). Obviously, there is a trade-off between the level of confidence one wishes and the level of tolerance used. The more tolerant the system is, the higher the chance to misclassification of a misbehaved node. On the other hand, the more strict the system is, the chance to misclassify a well-behaved node increases as well. We have shown that Eqn. 1 and Eqn. 2 are appropriate to grade the misbehavior level of a node. Furthermore, by adjusting the appropriate parameters, one can tune the system as desired.

5. Conclusion and Future Work

An ideal reputation system for ad hoc networks should be concerned in verifying nodal activity in all layers. Nevertheless, up to now most of the reputation systems proposed in the literature focus on a given layer, usually the network layer. As we have shown, a malicious node could easily act in another layer without being detected. Indeed, the MAC layer can be quite easy to trick and would provide the cheater with much higher throughput, as we have shown.

Although in this work we have shown that a lower contention window provides advantages for a misbehaved node, it should be clear that even when a node adhere to the protocol rules, in terms of backoff, a node may resort to other mechanisms to obligate neighbors to increase their contention window values. The results have shown that the higher the number of nodes, associated with greediness of the malicious node, may leave a malicious node undetected for a good amount of time. In this case, identifying malicious nodes may not be possible without increasing the chance of misclassify a well-behaved node. So, one need to careful and avoid taking premature decision as the chance of misclassification can be high, particularly if the misbehaving node is a careful cheater.

Acknowledgments

This work is supported in part by Fundação de Empreendimentos Científicos e Tecnológicos (FINATEC), Conselho Nacional de Desenvolvimento Cientifico e Tecnológico (CNPq) and Fundação de Apoio a Pesquisa do DF (FAP-DF).

References

[1] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11", http://standards.ieee.org, 1999.

[2] S. Bansal and M. Baker, “Observation-based cooperation enforcement in ad hoc networks,” Stanford University, CA, Tech. Rep., July 2003.

[3] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, “Mobile Ad Hoc Networking”, Wiley-IEEE Press, 2003.

[4] S. Buchegger and J. L. Boudec, “Performance analysis of the CONFIDANT protocol: Cooperation of nodes fairness in dynamic ad-hoc networks,” in Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC). Lausanne, CH: IEEE, June 2002.

[5] L. Buttya ́n and J. Hubaux, “Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks,” Swiss Federal Institute of Technology, Lausanne, Department of Communication Systems, Tech. Rep. DSC/2001, 2001.

[6] S. S. Frank Kargl, Andreas Klenk and M. Weber, “Advanced detection of selfish or malicious nodes in ad hoc networks,” University of Ulm, Dep. of Multimedia Computing, Ulm, Germany, Tech. Rep., 2004.

[7] M. S. Gast, 802.11 Wireless Networks: The Definitive Guide. O’Reilly, 2002.

[8] Q.He, D.Wu, and P. Khosla, “SORI: A secure and objective reputation-based incentive scheme for ad-hoc networks,” in Proceedings of IEEE Wireless Communications and Networking Conference (WCNC2004), vol. 2. IEEE, March 2004, pp. 825–830.

[9] A. Josang, R. Ismail, and C. Boyd, “A survey of trust and reputation systems for online service provision,” Decis. Support Syst., vol. 43, no. 2, pp. 618–644, 2007.

[10] X. Li, A. Nayak, I. Ryl, D. Simplot, and I. Stojmenovic, “Secure mobile ad hoc routing,” Advanced Information Networking and Applications Workshops, 2007, AINAW ’07. 21st International Conference on, vol. 2, pp. 737–742, May 2007.

[11] S.Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in Proceedings ACM/IEEE International Conference on Mobile Computing and Networking (MobiCOM), vol. 2, August 2000, pp. 255–265.

[12] J.-P. H. Maxim Raya and I. Aad, “Domino: A system to detect greedy behavior in IEEE 802.11 hotspots,” LCA, School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland, Tech. Rep., 2004.

[13] P. Michiardi and R.Molva, “CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks,” Institut Eurecom, France, Tech. Rep. EURECOM+816, December 2001.

[14] Y. Xiao, “Ieee 802.11e: QoS provisioning at the mac layer,” Wireless Communications, IEEE vol. 11, no. 3, pp. 72–79, June 2004.

[15] S. Zhong, Y. Yang, and J. Chen, “Sprite: A simple, cheat-proof, credit- based system for mobile ad hoc networks,” in In Proceedings of IEEE INFOCOM’03, vol. 3, San Francisco, CA, 30 March–3 April 2003, pp. 1987–1997.

Nicole Nagel received BSc degree in Computer Science from the University of Brasilia, in 2008. She is now pursuing her master’s degree in computer science. She’s currently working as a developer with projects related to banking automation and budget management. Her interests include mobile computing, security, trust and reputation systems, and ad hoc networks.

Ruzbeh Shokranian received BSc degree in Computer Science from the University of Brasilia, in 2008. He has experience in projects related to banking automation and is interested in trust and security issues in mobile ad hoc networks. Currently he is doing a Professional Exchange in Thessaloniki, Greece.

Jacir Luiz Bordim received BSc in Computer Science from University of Passo Fundo (1994) , MSc degree in Computer Science from Nagoya Institute of Technology (2000) and Ph.D. degree in Information Science by Japan Advanced Institute Of Science And Technology (2003). He worked as a researcher at ATR-Japan from 2003 to 2005. Since 2005 he is an Assistant Professor with the Department of Computer Science at University of Brasília. Dr. Bordim is a member of the editorial board of the IEICE Transactions on Information and Systems. Dr. Bordim has published extensively in many international conferences and journals, with over 50 peer-reviews papers published. His interest includes mobile computing, collaborative computing, trust computing, distributed systems, opportunistic spectrum allocation, MAC and routing protocols.