Adaptation in Reputation Management Systems for Ad hoc Networks

Mohamed Tamer A. Refaei

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Computer Engineering

Dr. Luiz DaSilva, Chair
Dr. Ing-Ray Chen
Dr. Mohamed Eltoweissy
Dr. Scott Midkiff
Dr. Jung-Min Park
Dr. William Tranter

February 2nd, 2007
Arlington, Virginia

Keywords: reputation management, ad hoc network security, node misbehavior

Adaptation in Reputation Management Systems for Ad hoc Networks

Mohamed Tamer A. Refaei

(ABSTRACT)

An ad hoc network adopts a decentralized unstructured networking model that depends on node cooperation for key network functionalities such as routing and medium access. The significance of node cooperation in ad hoc networks makes network survival particularly sensitive to insider node behavior. The presence of selfish or malicious nodes in an ad hoc network could greatly degrade the network performance and might even result in a total communication breakdown. Consequently, it is important for both security and performance reasons to discourage, expose, and react to such damaging misbehavior.

Reputation management systems have been proposed to mitigate against such misbehavior in ad hoc networks. The functions of a reputation management system are to evaluate nodes' quality of behavior based on their cooperation (evaluation), distinguish between well-behaved and misbehaving nodes (detection), and appropriately react to misbehaving nodes (reaction). A significant number of reputation management systems have been proposed for ad hoc networks to date. However, there has been no attempt to consolidate all current research into a formal framework for reputation management systems. The lack of a formal framework is a potential weakness of the research field. For example, a formal comparison of proposed reputation management systems has remained difficult, mainly due to the lack of a formal framework upon which the

comparison could be based. There is also a lack of formal metrics that could be used for quantitative evaluation and comparison of reputation management systems.

Another major shortcoming in this research field is the assumption that the functions of reputation management (evaluation, detection, and reaction) are carried out homogeneously across time and space at different nodes. The dynamic nature of ad hoc networks causes node behavior to vary spatially and temporally due to changes in the local and network-wide conditions. Reputation management functions do not adapt to such changes, which may impact the system accuracy and promptness. We herein recognize an adaptive reputation management system as one where nodes carry out the reputation management functions heterogeneously across time and space according to the instantaneous perception of each of its surrounding network conditions.

In this work, we address the above concerns. We develop a formal framework for reputation management systems upon which design, evaluation, and comparison of reputation management systems can be based. We define and discuss the different components of the framework and the interactions among them. We also define formal metrics for evaluation of reputation management systems. The metrics assess both, the effectiveness (security issues) of a reputation management system in detecting misbehavior and limiting its negative impact on the network, and its efficiency (performance issues) in terms of false positives and overhead exerted by the reputation management system on the network. We also develop ARMS, an autonomous reputation management system, based on the formal framework. The theoretical foundation of ARMS is based on the theory of Sequential Probability Ratio Test introduced by Wald. In ARMS, nodes independently and without cooperation manage their reputation

management system functions. We then use ARMS to investigate adaptation in reputation management systems. We discuss some of the characteristics of an adaptive reputation management system such as sensitivity, adaptability, accuracy, and promptness. We consider how the choice of evaluation metric, typically employed by the evaluation function for assessment of node behavior, may impact the sensitivity and accuracy of node behavior evaluation. We evaluate the sensitivity and accuracy of node behavior evaluation using a number of metrics from the network and medium access layer. We then introduce a time-slotted approach to enhance the sensitivity of the evaluation function and show how the duration of an evaluation slot can adapt according to the network activity to enhance the system accuracy and promptness. We also show how the detection function can adapt to the network conditions by using the node's own behavior as a benchmark to set its detection parameters. To the best of our knowledge, this is the first work to explore the adaptation of the reputation management functions in ad hoc networks.

Acknowledgments

I would like to dedicate this manuscript to my amazing wife Norann. I could not have done this without you. You mean the world to me and I thank you for your love and support thought this journey. I would also like to dedicate it to my father Khairy, my brother Ashraf, my late mother Sawsan, and my uncle Ashraf. Your integrity and dedication in your careers gave me the role model to look up to. I owe you everything that I am. This is also dedicated to Amir, Mona, and Kareem Zaghloul for their help and support. You gave me the strength to fight for my goals and ambitions.

Above all, I would like to thank Dr. DaSilva for being an amazing mentor. You always found a way in your busy schedule for discussions and for reading my half-backed manuscripts. I sincerely thank you for your patience and guidance throughout this journey. It has been a privilege being your student and an honor having you as my advisor. I also like to thank Dr. Eltoweissy for all his help, direction, and endless support anytime of the day. Additionally, I would like to express gratitude to Dr. Midkiff for leading the IREAN program. I also thank Dr. Choi from the George Washington University for giving me the chance to work with her and her students.

I can not forget my friends who were always supportive of me. Mohamed and Rea Abdel-Aal, Ahmed Abdelhalim, Mohamed and Shahira Abouellail, Nancy Elmahdy, Mazen Esmat, Nora and Abeer Elbilawi, Mohannad Gomaa, Khaled Mostafa, Mona Moussa, Tamer Nadeem, and Tamer Sharnouby. It was always refreshing going to Aladdin and all the discussions we had there. Iman Mostafa, I am grateful for all your help with the servers. Finally, it was a privilege sharing Dr. DaSilva's lab with both Vivek and Waltermar. Our discussions were always fruitful.

Table of Contents

List of Figures

List of Tables

Chapter I. Introduction

1. Introduction and Motivation

Ad hoc networks consist of wireless, mobile, battery-operated and, at times, autonomous and heterogeneous nodes. Nodes in an ad how network dynamically form a communication network without the use of existing physical network infrastructure or centralized administration. This self-organizing networking model requires cooperation among nodes to compensate for the lack of infrastructure. Node cooperation is essential to the survival of an ad hoc network since it is the basis of key network functionalities such as medium access, routing, authentication, and security.

Consider, then, what might happen to the network if one or more nodes refuse to cooperate. It becomes important to detect this sort of misbehavior, both to minimize the harm that such nodes may cause to the network and to provide an incentive for nodes to cooperate. We consider a node to misbehave if it acts selfishly, refusing to cooperate in order to preserve its own resources, or maliciously, deliberately trying to disrupt network operation through uncooperative behavior.

Reputation management is an effective tool in discouraging misbehavior and in mitigating the effects of misbehavior if it occurs. Reputation management systems can target security concerns related to internal misbehavior attacks in decentralized and unstructured networks that cryptography-based security solutions are ineffective against. Cryptography-based security solutions mostly secure the network against external attacks. Moreover, some of the common requirements of these solutions, such as bootstrapping, considerable computation and communication capabilities, and trusted

third parties may not be feasible due to the dynamic, resource-constrained, and infrastructureless nature of ad hoc networks. Reputation management systems, on the other hand, are distributed and low-overhead in nature. Also, reputation management systems supersede Intrusion Detection Systems (IDSs), which merely identify attacks on the network, in that they can be used to differentiate services to nodes according to the quality of their behavior. The functions of a reputation management system are to: encourage nodes to cooperate, evaluate nodes' quality of behavior based on their cooperation (evaluation function), distinguish between well-behaved and misbehaving nodes (detection function), and appropriately react to misbehaving nodes (reaction function).

Reputation management systems for ad hoc networks have been a topic of research for several years. However, there is no formal reputation management system framework to date that consolidates all current research in this area. This is considered a potential weakness of the research field. For example, a formal comparison of proposed reputation management systems remains difficult, mainly due to the lack of a formal framework upon which the comparison could be based. A formal framework will provide the basis for design and formal evaluation and comparison of reputation management systems. It will also provide formal metrics to assess, quantitatively, the effectiveness of a reputation management system (security issues) as well as its efficiency (performance issues).

Most of the proposed reputation management systems for ad hoc networks do have merits. However, a major shortcoming is the assumption that reputation management systems at different nodes carry out the reputation management system

functions (evaluation, detection, and reaction functions) homogeneously across time and space. In other words, the evaluation criteria, detection decision factors, and reactive measures are the same at all nodes at all times. The homogeneous application of reputation decisions often results in sub-optimal performance, as it restricts adaptation [1]. Ad hoc networks operate in a dynamic environment where network conditions change from time to time and may be perceived differently from one node to another. At times the network conditions are such that a node j that chooses to act cooperatively can successfully do so by forwarding all traffic routed through it (due to moderate traffic activity, favorable channel conditions, etc.). At other times network conditions are such that node j may not be able to successfully forward traffic routed through it due to inadequate conditions (such as high congestion, channel impairments, etc.). Node behavior may vary from one node to another and from time to time due to changes in the local and network-wide conditions, affecting the ability of the reputation mechanism to distinguish between a node's willful decision not to forward packets (misbehavior) and its inability to do so due to adverse network conditions.

The performance of a reputation management system, which is usually measured in terms of accuracy and promptness of misbehavior detection, may deteriorate if it does not adapt to network conditions. If the perception of cooperative behavior remains the same under both adequate and inadequate network conditions, a node may be incorrectly identified as misbehaving during inadequate network conditions and inappropriate reaction may be taken against it (e.g. isolation from the network). Hence, the evaluation criteria, detection decision factors, and reactive measures should be carried out

heterogeneously across time and space. We herein refer to such a reputation management system as adaptive.

2. Objectives

The main objective of this work is to investigate adaptation in reputation management systems. In order to satisfy this goal, a number of objectives must be met. Initially, a formal framework for reputation management system must be developed. This framework should identify and discuss the necessary components of a reputation management system and how these components interact amongst each other. Thus, the developed framework will provide the basis upon which any reputation management system can be designed and will serve as a basis for evaluation and comparison of any reputation management system(s). The framework will also define formal metrics to assess, quantitatively, the effectiveness and efficiency of a reputation management system.

Secondly, a reputation management system must be developed based on the formal framework. The reputation management system must have a sound theoretical foundation. It should also assume node independence with respect to the reputation management system functionalities. This reputation management system should operate correctly assuming each node independently, without necessarily relying on or consulting other nodes, operates its own reputation management system.

Thirdly, the characteristics of an adaptive reputation management system must be identified. Next, the realization of each characteristic to the different functionalities of the developed reputation management system (i.e. evaluation, detection, and reaction

functions) by each node independent from its peers must be investigated. We will focus on this work on the evaluation and detection functions.

3. Applicability

This research will to apply to static and mobile wireless local area network (LAN) environments. The developed reputation management system framework is meant to serve as a basis for designing and building new reputation management systems for ad hoc networks. The decomposed functionalities of the reputation management system allow the designer to choose the appropriate components to activate based on factors such as the purpose and the deployment model of the reputation management system. The framework will also aid in evaluating and comparing reputation management systems for ad hoc networks.

The reputation management system developed should operate in commercial ad hoc networks where cooperation is key to network survival and nodes are assumed to be autonomous. Most importantly, it should apply to dynamic environments where network conditions change spatially and temporally, requiring nodes to adapt the functionalities of their perspective reputation management system to optimize their operations.

4. Organization

This document is divided into six chapters. Chapter II presents related work. Chapter III presents our proposed framework for reputation management system. Chapter IV presents the theoretical foundation of an autonomous reputation management system that we developed based on the reputation management system framework. Chapter V presents the operations of the autonomous reputation management system and

evaluates it. In Chapter VI we discuss the adaptation of the evaluation and detection functions of the autonomous reputation management system. We conclude by discussing the contribution of this work in Chapter VII, where we also outline potential future work on this topic.

Chapter II. Related Work

1. Attacks on Ad hoc Networks

An ad hoc network is a type of wireless local area network (WLAN) that is primarily characterized as dynamic and infrastructureless. Nodes in an ad hoc network have to compensate for the lack of infrastructure by cooperating in key network functionalities such as routing. Each node is assumed to function as a router for its neighbors' traffic to allow for multi-hop communication. The need for node cooperation as a key to network survival is a unique feature of ad hoc networks. Other networks, such as infrastructure-based WLANs and wireline networks, rely on existing infrastructure and special-purpose hardware to provide key network functionalities such as routing.

Previous work in [2] surveyed the security issues in ad hoc networks indicating that the significance of node cooperation in ad hoc networks makes network survival particularly sensitive to insider node behavior, making it an important security consideration. The threat model described in [3] identifies and classifies types of node misbehavior in ad hoc networks into four different types: failed nodes, badly failed nodes, selfish nodes, and malicious nodes. These four classes can be differentiated with respect to the node's intent and action. A failed node exhibits unintentional passive behavior, where it is unable to participate in cooperation-based functionalities, due to power failures, for example. A badly failed node, on the other hand, indicates unintentional active behavior, where a node may inadvertently advertise inactive routes or unnecessarily overload the network with routing updates. Selfishness is intentional

passive misbehavior, where a node chooses not to fully participate in the packet forwarding functionality to conserve its resources. Selfish nodes are motivated only by their self-interest in conserving their resources and may drop some or all packets forwarded through them accordingly. Selfish nodes do not collude with each other or exert additional effort to camouflage their behavior, such as slander attacks. Finally, maliciousness is intentional active misbehavior, where a node's aim is to deliberately disrupt network operations.

Malicious nodes may attack the link layer, taking advantage of the cooperative nature of the medium access control (MAC) protocol. The protocol requires each pair of communicating nodes to seek a unanimous promise from all other nodes within range to have an exclusive access to the channel. This characteristic is exploited in a number of denial-of-service (DoS) attacks including collision attacks [4] and virtual jamming attacks [5]. In a collision attack, a malicious node ignores the MAC protocol specifications by accessing the medium when other nodes within range are transmitting or receiving data, which causes collisions. The virtual jamming attack is specific to IEEE 802.11. IEEE 802.11 adopts a virtual sensing mechanism whereby a node that wishes to access the medium to transmit data to one of its neighbors must send this neighbor a Request to Send (RTS) frame first. The RTS frame includes a duration field which informs all other nodes within range for how long the node will transmit over the medium so as no one would attempt to access the medium. In the virtual jamming attack, a malicious node transmits RTS frames with the highest possible value in the duration field (32767 μs). This means that by sending 31 RTS frames per second with the maximum value in the duration field, a malicious node can monopolize access to the medium.

Another type of misbehavior at the MAC layer is introduced in [6], where a malicious node may use small DIFS (Distributed coordination function Interframe Space) values, allowing it to access the medium faster than other nodes. A sequential probability ratio test based algorithm was introduced in [4] to detect such misbehavior.

Malicious attackers may also exploit node cooperation to launch attacks on the network layer, where nodes are assumed to cooperate in the routing functionality. Examples of attacks on the routing layer are the blackhole and greyhole attacks [7], wormhole attacks [8], and rushing attacks [9]. In a blackhole attack, the malicious node broadcasts fabricated routing information placing itself on the shortest path of routes to all destinations (e.g. advertising a distance of one hop to all other nodes in the network). This directs a lot of traffic in the malicious node's way. The malicious node drops all (blackhole attack) of some (greyhole attack) of the packets forwarded through it. In a wormhole attack, a malicious node receives packets at one location in the network and tunnels them to another malicious node, with whom it is colluding, at another location in the network. Such attacks typically involve two malicious nodes that are not within range of each other. The attackers collude to position themselves within routes in the network by relaying routing control packets to one another on an out-of-band channel available only to them. The wormhole attack by itself may not disrupt the network. However, malicious nodes usually combine the wormhole attack together with a blackhole or a greyhole attack. The rushing attack is similar to the wormhole attack. The goal of the malicious nodes in both attacks is to increase the probability that routes that include them are discovered first. In a wormhole attack, this is done through an out-of-band channel. In a rushing attack, this is done by using higher transmission power to

increase the attacker's transmission range. The rushing attack is also typically combined with a blackhole or a greyhole attack.

Malicious nodes can also target transport layer protocols as in the case of the jelly fish attack [10]. The jellyfish is a protocol-compliant attack that targets the TCP protocol. It complies with the TCP protocol specification but targets end-to-end congestion control to deteriorate the network performance. Examples of such attacks are re-ordering packets, periodically dropping packets, and varying packet delay of a TCP flow. All such operations are not uncommon in a TCP flow but have detrimental impact on the TCP goodput if maliciously orchestrated.

2. Proposed Solutions

Previous work noted the importance of securing ad hoc networks against attacks such as the ones described in the previous section [11-13]. To address the problem of node misbehavior in ad hoc networks, three classes of solutions have been proposed: secure routing protocols, cooperation incentives, and node behavior evaluation.

2.1. Secure Routing Protocols

The merit of this class of solutions ([7, 14-16]) is to secure the establishment and maintenance of routes in routing protocols such as Ad hoc On-Demand Distance Vector Routing (AODV) [5] and Dynamic Source Routing (DSR) [18] against tampering. The attack model described in [7] classifies misbehavior on the routing functionality into passive and active attackers. A passive attacker may eavesdrop on the network, which is a threat to privacy and anonymity. An active attacker may inject incorrect routing information into the network to cause routing disruption by creating routing loops,

blackholes, greyholes, or even partitioning the network. An active attacker may also attempt to consume resources belonging to other nodes by injecting extra packets in the network, consuming bandwidth and other nodes' energy.

ARIADNE is introduced in [7] to protect DSR against active attacks using TESLA [17]. The protocol assumes a security association exists between all nodes in the network. In [14], SEAD, a protocol that protects Destination-Sequenced Distance-Vector Routing (DSDV) against attacks similar to the ones in [7], is introduced. In particular, the protocol protects DSDV against malicious attempts to create incorrect routing states in other nodes (by changing the destination, metric, or source address of a route discovery or route maintenance packets). The protocol also uses one-way hash functions. In [16], security is introduced as a metric for route computation in AODV. The protocol attempts to establish a route based on a predetermined trust level that is defined by the source node. Any node within the established route must meet the predetermined trust level. In [15], a protocol is introduced to protect route discovery against fabricated, compromised, or replayed routing control packets. This protocol assumes a security association exists between the source and destination nodes only and does not make any assumption about intermediate nodes (they may exhibit malicious behavior). The scheme may fail in the presence of colluding nodes.

In general, securing routing protocols can protect the network against malicious misbehavior at the network layer only, in particular with respect to route discovery and maintenance. It does not protect the network against selfish misbehavior at any layer (including the network layer) or malicious misbehavior at layers other than the network layer.

2.2. Byzantine Fault Tolerance Techniques

The Byzantine Generals Problem is an agreement problem in which a group of generals must decide independently but unanimously whether or not to attack the army of their enemy. The generals are geographically separated. Hence, they must communicate with each other using messages in order to unanimously decide whether to attack or not. The problem complication stems from the assumption that some traitors may be present amongst the generals and may attempt to corrupt the generals' decision. For example, the traitors may forge messages to trick other generals into making a decision that is not consistent with their desires or that of others, or confusing some generals so that conflicting decisions are made (i.e. some generals attack and some do not). The requirements for a solution to the problem is that all loyal generals decide upon the same plan of action (i.e. attack or not) and that a small number of traitors cannot corrupt a unanimous decision by the generals. The Byzantine Generals Problem models real-world environments where nodes in a communication network may misbehave either unintentionally (due to network congestion for example) or intentionally by acting selfishly or maliciously [18, 19]. The goal of a solution is to mask the negative impact of these nodes on the network.

Several approaches have been proposed to the Byzantine Generals Problem mainly focusing on Byzantine fault tolerance techniques [19, 20]. Byzantine fault tolerance attempts to set rules such that loyal generals may have a unanimous agreement on their decision. Some of the solution techniques that assume messages can be forged can mask a bounded number of Byzantine failures (suffer up to m traitors in the presence of $3m+1$ generals) but fail otherwise. Solutions that assume messages can not be forged

may work in the presence of an arbitrary number of traitors. Note that the detection of traitors is a not a requirement of the Byzantine Generals Problem, rather, masking their impact is the main goal. This corresponds to masking the impact of node misbehavior in the network rather than detecting misbehaving nodes and punishing them accordingly. The presence of a single malicious node in the network may have detrimental impact on the network performance. Masking the impact of misbehavior may not be possible without identifying its source.

2.3. Cooperation Incentives

This class of solutions applies to nodes that are rational (i.e. nodes that adopt the behavior that benefits them most). The goal of this class of solutions is to provide incentives for nodes to cooperate in such a way that rational nodes lose if they do not cooperate [21, 22]. In an environment where nodes are autonomous, a node's cooperation level in key network functionalities is influenced by factors like energy consumption. This is shown in [23], where node cooperation in ad hoc networks is studied assuming that nodes' actions are strictly determined by self-interest and that each node has a minimum lifetime constraint.

Credit based systems have been proposed to incentivize nodes to cooperate in packet forwarding by offering them payments in return. Every time a node forwards a packet on behalf of another node it receives a payment from that node. Nodes are also charged when they request others to forward packets on their behalf. For a node to be able to pay others it must have enough credit, and it can obtain credit by forwarding other's packets [21]. In [22], SPRITE, a credit-based system is introduced. A node loses credit for all packets where it is the source and gains credit when it routes packets for

other nodes. This system assumes a centralized server that accounts for all packets received, transmitted, and dropped in the network and takes care of making payments to nodes for their forwarding services and collecting payments from nodes that request forwarding services. Nugglets is another credit based system, introduced in [24]. Nugglets requires tamper-proof hardware to prevent payment fabrication and does not require a centralized payment system. This class of solutions is not designed to prevent attacks by malicious nodes. The goal of a malicious node is to deteriorate network performance irrespective of the cost it incurs. Hence, malicious nodes cannot be motivated to cooperate.

The economics of node cooperation has been studied using game-theory [25]. Game theory provides a framework to study the behavior of rational participants in a strategic interaction. It studies strategic situations where players (i.e. nodes) choose different actions in an attempt to maximize their returns. One game-theoretic strategy that has been applied as a cooperation incentive in ad hoc networks is tit for tat [26]. Tit for tat is based on the English saying meaning "equivalent retaliation." A node using this strategy will initially cooperate, and then respond in kind to other nodes' actions. The node cooperates with other nodes that were previously cooperative, but does not cooperate with others that were not. Recent work has shown that the threat of retaliation may be effective as an incentive for node cooperation [27].

Mechanism design is a branch of game theory that studies the design of incentives for rational nodes to act in a manner that is conducive to reaching the outcome desired by the designer. Typically, each user has a utility that may be different from the overall network utility. Each user chooses a strategy (in the context of this discussion, the

possible strategies are to act cooperatively or to misbehave) that maximizes its utility irrespective of the impact of such strategy on the network's utility. Mechanism design is the art of designing rules (usually in terms of cost and payment) such that choosing a strategy that results in social optimum (i.e. maximizing the overall network utility) is a dominant strategy of each player (i.e. a strategy that maximizes each user's utility as well). Mechanism design has been applied to routing protocols in ad hoc networks to encourage nodes to forward traffic on behalf of one another [28, 29]. These approaches adopt payments and credits along with a Vickrey-Clarke-Groves (VCG) auction mechanism.

2.4. Behavior Assessment

The main goal of this class of solutions is to evaluate other nodes' behavior and build a reputation for each accordingly. This reputation can then be used to build trust in other nodes, make decisions about which nodes to interaction with, and possibly punish a node when needed. Hence, systems that fall under this class of solutions are commonly known as reputation management systems. Reputation management systems have been proposed to address insider misbehavior security concerns (i.e. selfish as well as malicious misbehavior) in self-organized communication systems such as ad hoc networks [30-42], peer-to-peer systems [43], and in artificial intelligence [44]. The goal of a reputation management system in ad hoc networks is to evaluate node behavior, identify misbehaving nodes, and appropriately react to their misbehavior.

In [45, 46], the authors attempted to define an ontology for reputation. Seven concepts are used in [46] to define the several aspects of reputation: reputation nature, reputation roles, information source of reputation, reputation evaluation and

measurement, reputation maintenance, reputation scope, and reputation propagation. Reputation nature distinguishes a reputation according to the nature of the entity it is associated with (e.g. a person, a group of people, a product, a service, an event, etc.) Reputation role identifies the roles of the entities that participate in formation and propagation of reputation. Mainly, these entities are the evaluator, the target, the beneficiaries, and the propagators. The evaluator evaluates the behavior of a target and identifies its reputation accordingly, the beneficiaries are the entities to whom the evaluation of the target is valuable, and the propagators are the ones that propagate reputation information about a target to other entities. Information source of a reputation identifies the source of information used for evaluation. These sources can be based on direct interaction (i.e. first-hand) or based on recommendations or opinions received from others (i.e. second-hand). Reputation measurement identifies a measure for reputation which can merely be as simple as a binary expression such as "good" or "bad". Reputation maintenance identifies when and why reputation may change and what factors may affect reputation over time. Reputation scope distinguishes reputation according to the manner they are employed. Two possible scopes are local, if reputation is assigned and maintained by an individual, or global, if reputation is assigned and shared collectively by more than one entity. Finally, reputation propagation identifies when, to whom, about whom, what, and how to propagate reputation information.

Typically, a reputation management system chooses a metric that is a good indicator of node behavior with respect to the functionality where cooperation is desired (e.g., packet forwarding in ad hoc networks) and monitors it for nodes whose behavior it is evaluating. We call such a metric the *evaluation metric*. Reputation management

systems rely on two types of evaluation metrics. The first, which we call first-hand metrics, uses information that locally observed by the node, which we refer to as first-hand observations, to calculate the evaluation metric. The local information used to calculate the metric can be collected either passively or actively. Passively collected information is usually acquired by promiscuously monitoring neighbors' actions. Actively collected information is based on the receipt of evidence, such as an acknowledgement in the case of routing functionality, which could be used as an indication of node behavior. An example of a system that uses an evaluation metric calculated based on passively collected information is the watchdog mechanism introduced in [39]. This mechanism uses a metric that is calculated based on passive acknowledgements. In promiscuous mode of operation, a node monitors its next hop neighbor through which a packet was forwarded to ensure that it forwards the packet to the next hop node towards the packet's destination. For each neighbor, the metric is calculated based on the ratio of number of packets where a passive acknowledgement is noted to the number of packets forwarded through the neighbor. The drawbacks of the watchdog mechanism are analyzed in [42]. In [34], a testbed is used to evaluate the performance of watchdog mechanisms under a number of misbehavior attacks. On the other hand, reputation management systems proposed in [30, 35, 36, 40] rely on active first-hand metrics. These systems use acknowledgments received for each packet sent as indications of good behavior. The lack of acknowledgements or the presence of retransmissions are used as indications of misbehavior. The ratio of the number of packets sent through a neighbor that were acknowledged by the destination to the number of packets sent through the same neighbor is the metric of interest in these systems.

The second type of evaluation metrics, which we call second-hand metrics, are based on hearsay (the sharing of first-hand observations among nodes in the network) [33, 37, 41], which we call second-hand observations. In [42], the authors qualitatively analyze the drawbacks and attacks against systems that employ second-hand metrics and indicate that the most reliable evaluations are those which a node makes based on local information. Second-hand observations have some potential drawbacks, mainly related to overhead, misreporting, and collusion [42]. The overhead of sharing escalates as the size of the network increases. However, [37] succeeds in limiting sharing to local neighbors, thus reducing communication overhead. Moreover, nodes may lie about the observations they share about other nodes by falsely reporting good or bad behavior. Also, nodes might collude to influence other node's behavior evaluation by spreading false rumors. In [31], the robustness of a reputation management system that employs sharing was increased against false accusations by introducing a Bayesian-based estimation mechanism. Additionally, [31] introduced a node redemption technique that re-admits misbehaved nodes to the network for re-evaluation, and a reputation fading technique that mitigates against sudden exploitation of built-up reputation. On the other hand, systems that employ second-hand metrics can detect node misbehavior more quickly compared to systems that employ first-hand metrics [31, 32]. This is due to the increased amount of information regarding a particular node's behavior.

In [47], the authors develop a model to stimulate cooperation in autonomous ad hoc networks in the presence of selfish and malicious nodes. In [48], an approach that mixes between a reputation-based system and a payment-based system is introduced. Nodes monitor and evaluate their neighbors' behavior. Through a localized collaborative

approach, credit is issued to nodes whose neighbors agree are cooperative. Once misbehaving nodes are detected by the majority of their neighbors, they are issued no credit and hence isolated from the network. In [6], a sequential probability ratio test based algorithm was introduced to detect uncooperative behavior at the MAC layer in ad hoc networks. In [49] the problem of misbehavior at the MAC layer is introduced and formulated as a minmax robust sequential detection problem.

3. Evaluation Metrics for Reputation Management Systems

In this section we discuss the evaluation metrics used to assess the performance of reputation management systems. We classify the performance metrics used to evaluate reputation management systems into efficiency metrics and effectiveness metrics. Efficiency metrics measure the impact of the reputation management system on the performance of the network. Reputation management systems may require exchange of control information amongst nodes (e.g. information used by second-hand metrics). They also perform reputation related tasks (e.g. evaluation of node behavior, isolation of misbehaving nodes) and may store reputation related information. This results in communication, computational, and storage overhead which may impact node as well as network performance. On the other hand, effectiveness metrics measure the ability of a reputation management system to reduce the impact of misbehavior as well as its accuracy in detecting misbehaving nodes. In most cases, the effectiveness as well as the efficiency of evaluation metrics is only defined qualitatively. Tables 1 and 2 list some of the commonly used efficiency and effectiveness metrics in the literature, respectively.

Table 1 Reputation Management System Efficiency Metrics

Overhead	Communication	Some reputation management systems use control packets to perform the reputation management functionalities (e.g. exchange of second-hand reputation information). This results in communication overhead exerted on the network by the reputation management system. ([22] [30] [50] [33] [39])
	Computation	Running the functionalities of reputation management system exert an overhead on a node's central processing unit (e.g. calculating node reputation). This metric measures the extra computation overhead exerted on a node by the reputation management system. ([22] [50])
	Storage	Some reputation management systems require maintaining state information and reputation related values in the node's memory [40]. This results in storage overhead exerted by the reputation management system. ([22] [50])

Table 2 Reputation Management System Effectiveness Metrics

Misbehavior Impact Reduction	Throughput	This metric measures the ability of a reputation management system to reduce the negative impact of misbehaving nodes on network throughput. Typically, this measures the network throughput in the presence and in the absence of the reputation management system. ([22] [30] [33] [36] [37] [39])
Quality of Detection	Exposure Time	This metric measures how fast a reputation management system can detect misbehaving nodes. ([50] [32])
	Exposure Ratio	This is the percentage of misbehaving nodes successfully detected. ([40])
	False Positives	In detecting misbehaving nodes, cooperative nodes may be falsely identified as misbehaving. This metric considers how often such cases occur. ([50] [36] [39] [40])
	False Negatives	In detecting misbehaving nodes, misbehaving nodes may be falsely identified as cooperative. This metric considers how often such cases occur. ([50])
	Robustness to Liars	Malicious nodes that use second-hand metrics are vulnerable against slander attacks [32]. This metric measures the robustness of a reputation management system against such misbehavior. ([32] [38])
Quality of reaction	Cooperation Pay-off	This metric measures the quality of the reaction function in terms of the benefit a node gets from acting cooperatively. It measures the difference between the

		throughput a cooperative node receives as opposed to that of a misbehaving node. ([33] [38])

4. Positioning of Contribution

A significant number of reputation management systems have been proposed in the literature. However, as highlighted by the recent work in [51], the lack of a formal framework for reputation management systems is a weakness. The authors in [51] also highlight that there are comparatively few analytical studies in the realm of reputation management in ad hoc networks. Moreover, no work has proposed (to the best of our knowledge) adaptation of the reputation management system functions. Limiting adaptation in a self-organizing network results in sub-optimal performance as highlighted in [1]. In [52], an adaptive reputation framework was proposed for peer-to-peer systems. However, adaptivity in this work was not referring to the system functionalities. Rather, the score assigned to a peer in a resource is adapted according to its behavior. In other words, the score fluctuates according to the behavior of the peer. This is expected in any reputation management system and is different from our notion of adaptation where the functions or reputation management system adapt to the local and network-wide conditions.

In this work, we develop a framework for reputation management systems. We define and discuss the different components of the framework and the interactions among them. We also define formal metrics for evaluation of reputation management systems. The metrics assess both the effectiveness (security issues) of a reputation management system in detecting misbehavior and limiting its negative impact on the network, and its efficiency (performance issues). We also consider in our work the importance of taking into account the dynamic nature of ad hoc networks, its impact on node behavior, and the

consequential impact on the performance of reputation management systems. Accordingly, we use the formalized framework to develop an adaptive reputation management system. We develop a theoretical model for the reputation management system and show its effectiveness through extensive simulations. We then address how the different functionalities of the reputation management system can adapt to change in node behavior that may result as a consequence of the dynamic nature of ad hoc networks.

5. Summary

In this chapter, we highlighted attacks on ad hoc networks due to internal node misbehavior emphasizing the impact of such misbehavior on the network and recognizing the difference between selfish and malicious behaviors. We then discussed different solution approaches to the node misbehavior problem in ad hoc networks and the ability of each to mitigate against node misbehavior. We drew the attention to reputation management systems as a solution that is capable to evaluating node behavior as well as identifying and appropriately reacting to misbehaving nodes. Finally, we positioned our work in comparison to the different solution approaches proposed in the literature including reputation management systems.

Chapter III. Framework for Reputation Management Systems

1. Formal Framework

In this section we decompose the different components of a reputation management system and discuss the functions of each. In our discussion, we consider a network graph of nodes in $\mathbf{N}= \{1,2,\ldots, N\}$.

1.1. Auditor

The *auditor* component (Figure 1) is responsible for collecting and interpreting observations that are relevant to node behavior with respect to packet forwarding and for periodically reporting them to the *behavior evaluator* component. Usually, there are two types of observations that an auditor would gather, first-hand and second-hand observations.

First-hand observations are in the form of experience gained from direct interactions with other nodes with respect to packet forwarding. One type of first-hand observations is the *watchdog* mechanism [39] (also referred to as *passive acknowledgements*). This mechanism monitors a node's neighbors to make sure they forward packets relayed through them as expected. An observation that a neighbor forwards a packet routed through it is interpreted as an indication of cooperative behavior. An observation that a neighbor does not forward a packet routed through it is interpreted as an indication of misbehavior. Another type of first-hand observations are transport or application layer acknowledgements [40]. Acknowledgements are taken as indication of cooperative behavior of all nodes along a route with respect to packet

forwarding, while a retransmission is taken as an indication of misbehavior somewhere along the route.

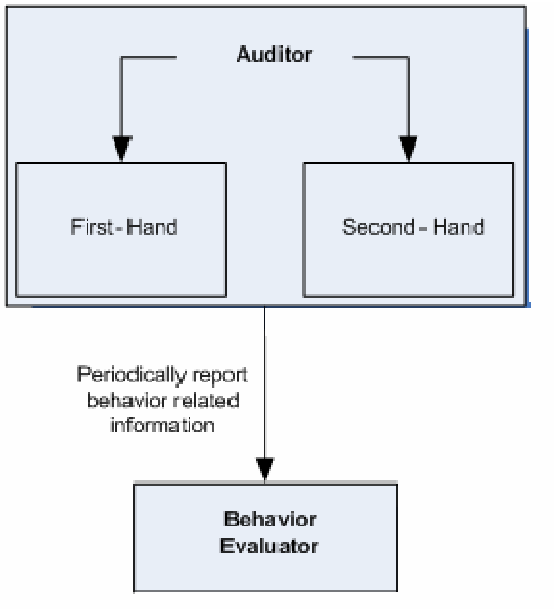

Figure 1 The *auditor* component

Second-hand observations are employed when nodes share their experiences, interacting with one another. Sharing can be local or global. Local sharing is usually preferred in large scale environments due to the high overhead of global sharing in such environments. However, global sharing results in more information reaching more nodes, which aids nodes in making better evaluations of others' behavior. One type of second-hand observations is reputation sharing, where nodes share their perceived reputation of other nodes (e.g. node $i \in \mathrm{N}$ informs its neighbors that another node $j \in \mathrm{N}$ has a good reputation while a third node $k \in \mathrm{N}$ has a bad reputation). The other type is information sharing, where nodes share information that can be used to reflect node behavior (e.g. the number of packets forwarded through a node).

1.2. Behavior Evaluator

The *behavior evaluator* component (Figure 2) is responsible for defining a metric to be used for node behavior evaluation. The metric is defined in terms of the observations gathered at the *auditor* component. The *auditor* component periodically reports to the *behavior evaluator* component observations made that can be used to assess node behavior. For example, the number of packets forwarded by each node and the number of packets routed through each are examples of information that can be gathered by the *auditor* component and reported to the *behavior evaluator* component. This information can then be used to define for example a metric called Packet Forwarding Ratio $= \frac{Number\ of\ Packets\ Forwarded}{Number\ of\ Packets\ Routed}$. Given two nodes $i, j \in \mathrm{N}$, where node i is evaluating the behavior of node j, node i can use this metric to assign a score r_{ij} to node j. The score assigned to node j by node i is a quantification of node j's behavior as perceived by node i based on the observations node i received from the *auditor* component.

The evaluation metric used considers observations differently according to their type and source as well as the longevity of their effect on the node's score. With respect to the observation type, first-hand observations are usually weighted more heavily than second-hand observations. This is because the confidence level in observations made by the node itself is higher than in observations communicated by others. The other weighting factor is the source of the observation. Observations made by some sources may be more trusted than those made by others. For example, a second-hand observation received from a neighbor that has a high score may be weighted more than another

received from a neighbor whose score is low. Also, a first-hand TCP retransmission observation may be weighted higher than the lack of a passive acknowledgement observation due to the possibility of non-symmetric links. Thus, the factors that affect node score given an observation are:

- *Observation Type* (w_t): The weight assigned to an observation based on its type.
- *Observation Source* (w_s): The weight assigned to an observation based on its source.

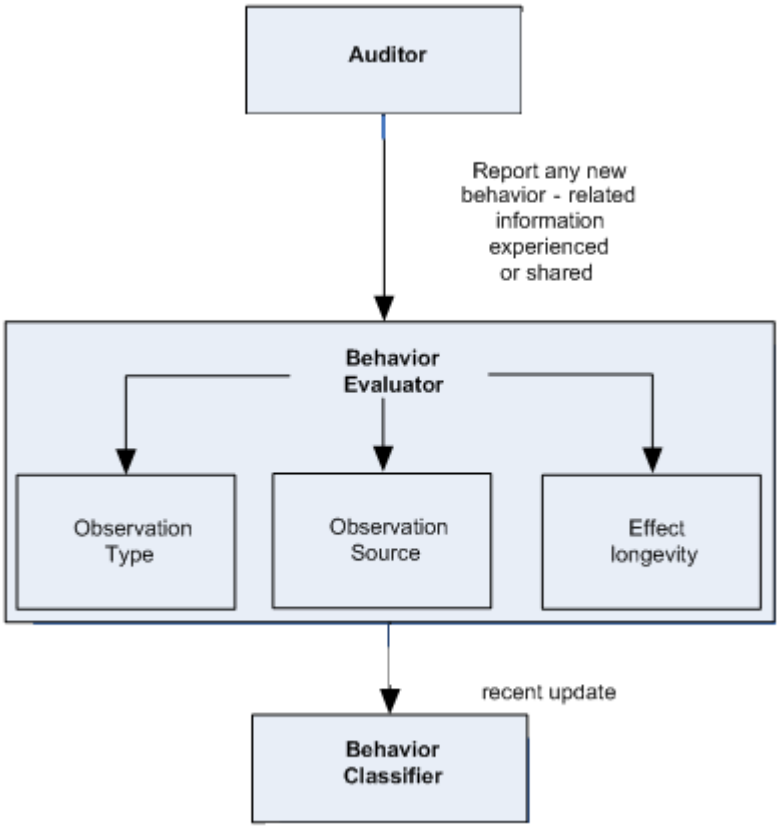

Figure 2 The *behavior evaluator* component

There are two commonly applied policies to define the longevity of the effect of an observation on the score assigned to a node, a short-term or a long-term effect. A short-term observation-effect policy gives more weight to newly received observations compared to observations received in the past. This means that the behavior evaluation

function is very sensitive to changes in node behavior. This has the advantage of mitigating against sudden exploitation attacks, where a node builds up a good reputation over time then misbehaves suddenly [50]. However, any legitimate or unintentional change in node behavior (e.g. due to congestion) may result in false positives.

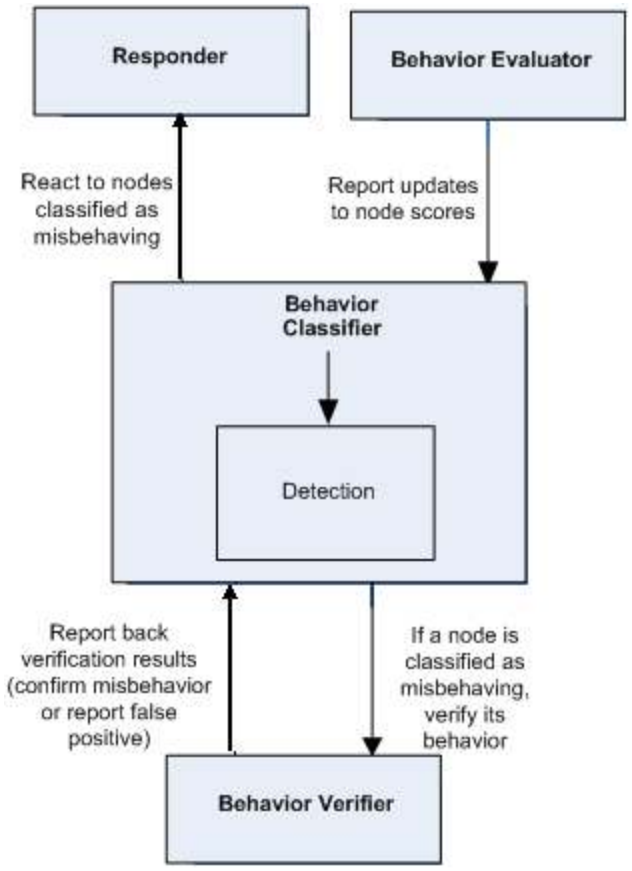

Figure 3 The behavior classifier component

On the other hand, a long-term observation-effect policy gives less weight to newly received observations compared to observations received in the past. This indicates that the evaluation function is not very sensitive to changes in node behavior. This policy results in fewer false positives but it is vulnerable to sudden exploitation attacks. Thus, the weighting factor at the effect longevity function is defined in terms of

an *Observation Effect* (w_e), which reflects whether a short-term or long-term observation-effect policy is followed.

Thus, when the *auditor* component at node i reports to the *behavior evaluator* component observations that reflect the behavior of node j, the *behavior evaluator* component updates node j's old score r'_{ij} to a new score r_{ij} as $r_{ij} = (1 - w_e) r'_{ij} + w_e w_t w_c r_{ij}$.

1.3. Behavior Classifier

The *behavior classifier* component (Figure 3) receives updates from the *behavior evaluator* component about changes in the score assigned to any node whose behavior is evaluated. This component then processes the change and classifies nodes according to their behavior into cooperative and misbehaving nodes. In order to classify node behavior, this component maintains as threshold score r_{thresh}. Any node j whose score r_{ij} as assigned by node i falls below r_{thresh} is classified as a misbehaving node. Otherwise, the node is classified as cooperative. The classification of node behavior need not be binary (either cooperative or misbehaving). Node classification could also be continuous as shown in Figure 4. In such case, the *behavior classifier* component will maintain multiple threshold scores to identify different levels of node misbehavior.

For nodes that are classified as misbehaving, the *responder* component is called to react appropriately to the misbehaving node. The *behavior verifier* component may also take appropriate action(s) to verify the perceived behavior of a node j.

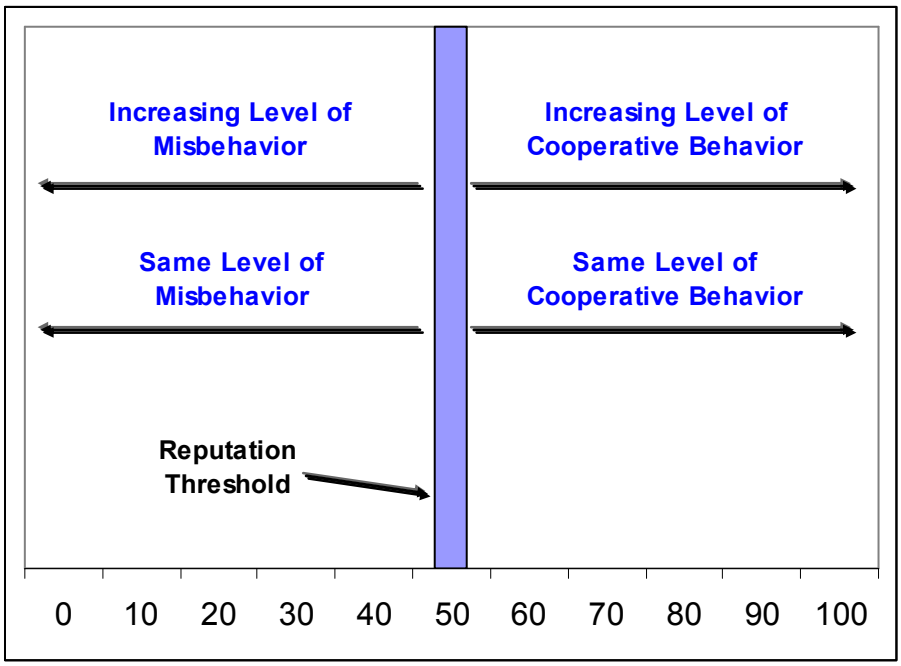

Figure 4 Continuous versus binary behavior classification. In this example the score assigned to nodes range between 0 and 100, where 0 being the worst of misbehavior and 100 being the best of cooperative behavior. $r_{thresh} = 50$ **in the case of the binary behavior classification.**

1.4. Behavior Verifier

The *behavior verifier* component (Figure 5) is responsible for verifying the node behavior classification performed by the *behavior classifier* component. If necessary, the outcome of the verification operation is sent back to the *behavior classifier* component to reclassify the node under verification.

Commonly, there are two methodologies for verification of node behavior, an active verification methodology and a passive verification methodology. Active verification methodology requires the verifier to engage in some communication activity regarding the verified node. An example of that is probing the suspect node to see if it responds. On the other hand, a passive verification methodology does not trigger any external actions. An example of passive verification is to listen in promiscuous mode to determine if the node under verification is forwarding any packets at all.

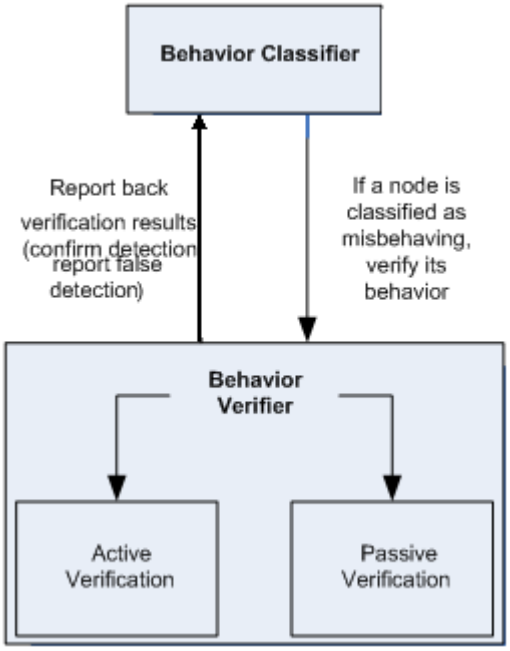

Figure 5 The *behavior verifier* component

1.5. Context Awareness

The *context awareness* component (Figure 6) is an important yet often neglected component of a reputation management system. Due to its anywhere-deployment readiness, node behavior in ad hoc networks will be greatly influenced by the deployment scenario. Factors such as location and node density can greatly impact a node's behavior.

Context awareness comes in three different types: awareness of self conditions, conditions of peers, and network conditions. Self conditions indicate the current state of oneself with respect to packet forwarding. This attempts to evaluate the node's own behavior with respect to packet forwarding by observing for example the number of packets the node was asked to forward, the number of packets that were actually forwarded, and the number of packets the node dropped.

There is nothing definitive that indicates conditions of other peers and network conditions with respect to the basis function of interest (in our case, packet forwarding). However, there are internal observations that could give adequate approximations of peer and network conditions. For example, awareness of the current queue size, packet statistics at the routing layer (received, forwarded, dropped), frame statistics at the link layer (received, retransmitted, dropped), current random backoff timer, and number of collisions, can serve as indications of network conditions. If this information is maintained per neighboring node (i.e. packets received, forwarded, dropped by each node), it can also serve as an indication of peer conditions.

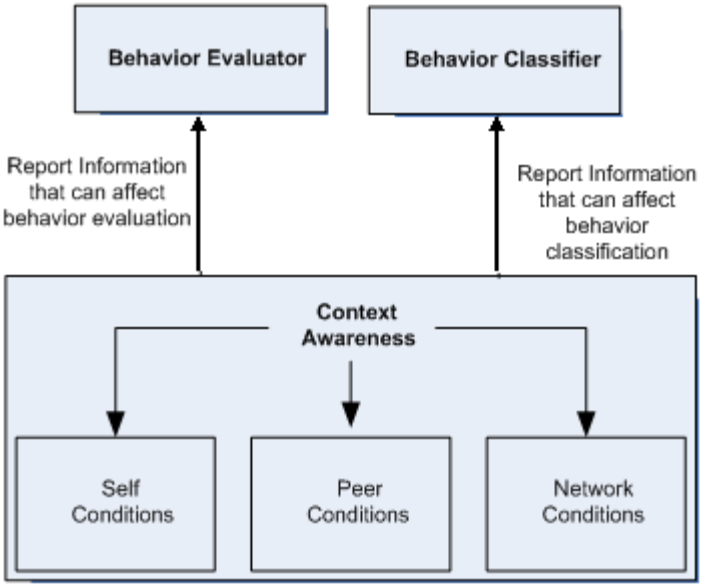

Figure 6 The *context awareness* component

Based on surrounding conditions (self, peer, and network), context awareness in a reputation management system can adapt the reputation management system functionalities to surrounding conditions to enhance the correctness of behavior evaluation. Adaptation can be in the form of adapting the node's behavior (e.g. adapting the node's offered service level) or adapting the behavior evaluation and classification

parameters. One example is node adaptation in the routing basis function in the presence of congestion. When the *context awareness* component senses congestion in the network, it can contact the node's cooperation model to decrease its level of cooperation in the packet forwarding functionality to avoid being penalized for dropping packets. The *context awareness* component can also contact the *behavior classification* component to update its r_{thresh} in order to avoid false positives.

1.6. Responder

The *responder* component (Figure 7) is responsible for taking action upon the detection of a misbehaving node. A response could be one of two forms, informative or disciplinary.

A disciplinary response is a response taken by a node that directly or indirectly affects its cooperation with the node it classified as misbehaving. An example of such response is refusing to participate in routes that include the misbehaving node. This means that all route requests by the misbehaving node will be denied.

An informative response is a response taken by a node to inform other nodes of its perceived detection of the misbehaving node. This is common in reputation management systems that employ reputation sharing, where a node sends out an alert message to other nodes upon detection of a misbehaving node or an indication of node misbehavior. Some reputation management systems adopt an informative response where a node informs its immediate neighbors, while others inform the entire network.

The other function of the *responder* component is to assess the possible effect of the detection of a node's misbehavior on the evaluation of its neighbors. This is an important function motivated by the observation that there is a negative effect on the

behavior of nodes that are neighbors of misbehaving nodes. Therefore, upon detection of a misbehaving node, the perceived behavior classification of its neighbors must be verified. Thus, the reaction module can call upon the *behavior verifier* component to attempt to verify the behavior of these nodes.

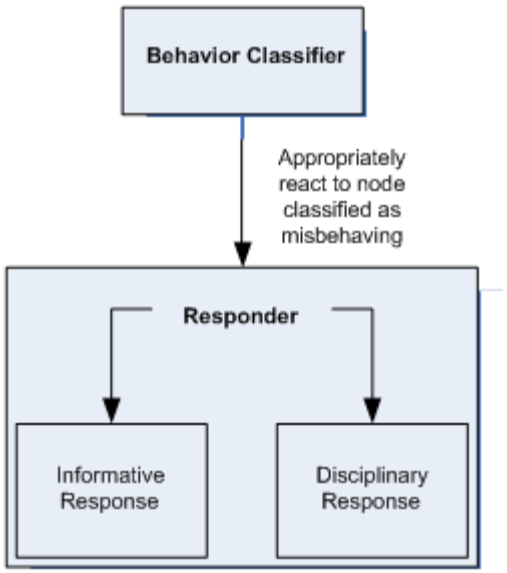

Figure 7 The *responder* component

2. Formal Metrics

There does not seem to be consensus on formal metrics used to evaluate a reputation management system. Each body of work defines evaluation metrics that are most relevant and appropriate to assess its effectiveness (security issues) and efficiency (performance issues). Evaluation metrics are often defined qualitatively. Among the most widely used metrics are throughput and false positives for assessing the effectiveness of a reputation management system, and communication overhead to measure its efficiency. However, throughput was shown in [53] to be a poor metric when used on its own in assessing the impact of selfishness or maliciousness on the network,

since such behavior might improve throughput at times. False positives and overhead on the other hand are important metrics and should be used in the evaluation of any reputation management system.

We define and describe a number of metrics for the quantitative comparison and evaluation of reputation management systems considering that nodes might manage their reputation management system functionalities autonomously. The metrics evaluate the performance of different functions of a reputation management system such as behavior classification, herein referred to as detection, and responder components, its effectiveness in reducing the impact of misbehavior, and its efficiency.

We use the indicator function $1_{\{A\}} = \begin{cases} 1, if\ A\ is\ true \\ 0, if\ A\ is\ false \end{cases}$ in our definition of evaluation metrics. Considering a network graph of nodes in $\mathbf{N}= \{1,2,\ldots,n\}$ with flows $F= \{1,2,\ldots,f\}$, where $d(i,j)$ is the distance in hops between nodes i and j, and misbehaving nodes in $M \subset \mathbf{N}$, the metrics we define are categorized as follows.

2.1. Reputation Management System Effectiveness

2.1.1. Misbehavior Impact

These metrics measure the impact of node misbehavior on the network.

- *Packet Drops (P_{drop})*: Measures the negative impact of misbehavior on the network performance. Since we are concerned with reputation management systems applied to the routing functionality, the impact of selfishness is calculated in terms of the

proportion of packets dropped due to misbehavior. Given P^i_{drop}, the number of data packets dropped by a node i, $P_{drop} = \frac{\sum_{i \in M} P^i_{drop}}{\sum_{i \in N} P^i_{drop}}$, *where* $\sum_{i \in N} P^i_{drop} \neq 0$.

- *Route Acquisition Failures (R_{Fail})*: Also measures the negative impact of misbehavior on the network performance. It measures failed flows, which are flows that were unable to acquire a functional route from source to destination throughout the simulation. Given T_f, the throughput achieved by flow $f \in F$, $R_{Fail} = \frac{\sum_{f \in F} \mathbf{1}_{\{T_f = 0\}}}{|F|}$.

2.1.2. Quality of Detection (Behavior Classification)

These metrics measure the effectiveness of the detection functionality of a reputation management system.

- *Exposure Ratio (EX_{Ratio})*: Measures the quality of a reputation management system's misbehavior detection. It is the average fraction, taken over all misbehaving nodes, of neighbors that correctly detect misbehavior. Detection is limited to neighbors of misbehaving nodes since it is assumed that they are the ones that interact most with them, and hence, have the most knowledge and ability to detect their misbehavior. Define $K = \{j \in M \wedge \exists i \in N \setminus M \text{ such that } d(i, j) = 1\}$ to be the set of misbehaving

nodes that have at least one well-behaved neighbor. Then, for $K \neq \phi$, exposure ratio is defined as $EX_{Ratio} = \frac{1}{|K|} \sum_{j \in K} \frac{\sum_{i \in N \setminus M} \mathbf{1}_{\{d(i,j)=1 \wedge r_{ij} < r_{thresh}\}}}{\sum_{i \in N \setminus M} \mathbf{1}_{\{d(i,j)=1\}}}$.

- *Exposure Rate ($EX_{Rate}(t)$)*: Measures how fast the reputation management system is capable of detecting misbehavior. The promptness of misbehavior detection is a desirable feature to limit the negative impact of misbehaving nodes on the network. At time t, exposure rate is $EX_{Rate}(t) = \frac{|\mathbf{M}| * EX_{Ratio}(t)}{t - t_0}$, $t_{\max} > t > t_0$, where t_0 is the time when the simulation started, $t_{\max}$ is the earliest time when $EX_{Ratio}(t) = 100\%$, and $EX_{Ratio}(t)$ is the exposure ratio at time t.
- *False Negatives (FN)*: Measures the failure of a reputation management system in detecting misbehavior. It is the average fraction, taken over all misbehaving nodes, of neighbors that failed to detect misbehavior. Hence, it is calculated as $FN = 1 - EX_{Ratio}$.
- *False Positives (FP)*: Measures the false alarms (i.e. false accusations) resulting from a reputation management system. It is the fraction of links isolated from the network due to a node falsely identifying one of its neighbors as misbehaving. False positives are quantified as $FP = \frac{\sum_{i \in N \setminus M} \sum_{j \in N \setminus M} \mathbf{1}_{\{i<j \wedge d(i,j)=1 \wedge (r_{ij} < r_{thresh} \vee r_{ji} < r_{thresh})\}}}{\sum_{i \in N \setminus M} \sum_{j \in N \setminus M} \mathbf{1}_{\{i<j \wedge d(i,j)=1\}}}$.
- *Convergence ($Conv(t)$)*: This metric is meant to indicate whether assessment of node behavior reaches a steady state. If a steady state is reached, the rate of change in node

reputation should be slower as the number of interactions increase. $Conv(t) = \sum_{i \in \mathrm{N}} \left[\sum_{(j \in \mathrm{N} \wedge d(i,j)=1)} abs(r_{ij}(t) - r_{ij}(t-1)) \right]$, where $r_{ij}(t)$ is the reputation value assigned by a node i to its neighbor j at time t, and $abs(x)$ is the absolute value of x. An implicit assumption here is that reputation values are updated at discrete times, which is the case in all practical reputation management protocols.

2.1.3. Quality of Reaction

These metrics measure the effectiveness of the reaction function of the reputation management system in isolating misbehaving nodes

- *Avoidance (Avoid):* Define $\mathrm{R}^{s \leftrightarrow d} = \{h_0, ..., h_k\}$ as an ordered set of nodes representing an active route (i.e. a route that is currently in use) from source node $s = h_0$ up to the last hop h_k before the destination node d ($k+1$ is the number of hops on the route). Let the ordering relationship on $\mathrm{R}^{s \leftrightarrow d}$ be $\forall x, y \in \mathrm{R}^{s \leftrightarrow d}, x < y \Leftrightarrow hop(x, s \leftrightarrow d) < hop(y, s \leftrightarrow d)$, where $hop(x, s \leftrightarrow d)$ is the number of hops between source node s and node x on the route from source node s to destination node d. A route $\mathrm{R}^{s \leftrightarrow d}$ is considered "infected" if $\exists h_i \in \mathrm{R}^{s \leftrightarrow d}$ such that $h_i \in \mathrm{M}$, $0 < i \le k$, otherwise it is "clean." We define avoidance as the ratio of infected routes to all active routes $Avoid = \dfrac{\sum_{i \in \mathrm{N} \setminus \mathrm{M}} \sum_{j \in \mathrm{N} \setminus \mathrm{M}} 1_{\{\exists R^{i \leftrightarrow j} such\ that\ R^{i \leftrightarrow j} \cap \mathrm{M} \neq \phi\}}}{\sum_{i \in \mathrm{N} \setminus \mathrm{M}} \sum_{j \in \mathrm{N} \setminus \mathrm{M}} 1_{\{\exists R^{i \leftrightarrow j}\}}}$.

- *Penalty (Pen):* As mentioned earlier, throughput is considered a poor metric for assessing the impact of misbehavior on the network. However, it could be used to assess the penalty on misbehaving nodes. Once detected, misbehaving nodes are usually penalized with suspension or isolation from the network, which limits their interactions with their neighbors. To measure the effectiveness of such penalty, we measure the ratio of the average throughput achieved by flows sourced at misbehaving nodes to the average throughput achieved by all flows. For each flow $f \in \mathrm{F}$, let T_f be the throughput achieved by the flow, s_f be the source of the flow, and d_f be its destination. Define $\mathrm{F}^{\mathrm{M}} = \{f : f \in \mathrm{F} \wedge (s_f \in \mathrm{M} \vee d_f \in \mathrm{M})\}$ to be the set of flows where the source or destination nodes are misbehaving. Penalty is defined as

$$Pen = \frac{\frac{1}{|\mathrm{F}^{\mathrm{M}}|}\sum_{f \in \mathrm{F}^{\mathrm{M}}} T_f}{\frac{1}{|\mathrm{F}|}\sum_{f \in \mathrm{F}} T_f}.$$

2.2. Reputation Management System Efficiency

These metrics measure the overhead of a reputation management system.

- *Communication Overhead ($P_{comm\_overhead}$)*: Measures the communication overhead exerted by a reputation management system on the network. Given P^i_{data}, the number of data packets originated at and sent by a source node i and P^i_{RMS}, the number of packets originated at and sent by the reputation management system at node i, overhead is $P_{comm\_overhead} = \frac{\sum_{i \in \mathbf{N} \setminus \mathbf{M}} P^i_{RMS}}{\sum_{i \in \mathbf{N} \setminus \mathbf{M}} (P^i_{data} + P^i_{RMS})}$, *where* $\sum_{i \in \mathbf{N} \setminus \mathbf{M}} (P^i_{data} + P^i_{RMS}) \neq 0$. For

fairness of evaluation, overhead includes reputation management system packets sent by well-behaving nodes only.

- *Computation Overhead* ($P_{comp\_overhead}$): If we assume the amount of CPU resources required to process a reputation management system packet is μ and that required to process a data packet is λ, this can be defined in a similar way as communication overhead:

$$P_{comp\_overhead} = \frac{\sum_{i \in \mathbf{N} \setminus \mathbf{M}} P_{RMS}^{i} * \mu}{\sum_{i \in \mathbf{N} \setminus \mathbf{M}} (P_{data}^{i} * \lambda + P_{RMS}^{i} * \mu)}, \quad where \quad (\sum_{i \in \mathbf{N} \setminus \mathbf{M}} P_{data}^{i} * \lambda + P_{RMS}^{i} * \mu) \neq 0 .$$

- *Storage Overhead* ($P_{storage\_overhead}$): If we assume the storage required per reputation management system packet is μ' units of memory and per data packet is λ' units of memory, this can be defined in a similar way as communication overhead:

$$P_{storage\_overhead} = \frac{\sum_{i \in \mathbf{N} \setminus \mathbf{M}} P_{RMS}^{i} * \mu'}{\sum_{i \in \mathbf{N} \setminus \mathbf{M}} (P_{data}^{i} * \lambda' + P_{RMS}^{i} * \mu')}, \quad where \quad (\sum_{i \in \mathbf{N} \setminus \mathbf{M}} P_{data}^{i} * \lambda' + P_{RMS}^{i} * \mu') \neq 0 .$$

3. Summary

In this section we presented a formal framework for reputation management systems. We presented and discussed the different components that construct a formal framework for reputation management system, the functionalities and parameters of each component, and the interactions among these components. The framework can be used to describe, design, and compare reputation management systems. We also introduced

formal metrics that can be used for quantitative comparison and evaluation of reputation management systems from a performance and security perspectives.

Chapter IV. Sequential Probability Ratio Test for Detecting Node Misbehavior in Ad hoc Networks

This work is published in the proceedings of the IEEE International Conference on Communications (ICC 2007), Glasgow, Scotland. M. Tamer Refaei, Y. Rong, L. DaSilva, and H. Choi "Detecting Node Misbehavior in Ad hoc Networks."

1. Network Model

Consider an ad hoc network with nodes $N = \{1,2,\ldots,n\}$ where nodes are either selfish or cooperative. We assume the network has selfish nodes in $M \subset N$. Hence, $|N|$ is the number of nodes in the network, $|M|$ is the number of selfish nodes, and $|N| - |M|$ is the number of cooperative nodes.

Packet drops occur due to two classes of factors: node misbehavior (one or more nodes along a route act selfishly, dropping packets); and network environment factors (under which we include congestion at the network layer, contention at the data link layer, physical communication impairments such as fading, etc.). Let P_{drop} represent the probability that a data packet will be dropped due to network environment factors. For simplicity of presentation, we assume that all nodes $i \in N$ will experience the same dropping probability due to environmental factors. The extension to the model to consider different drop probabilities at different nodes is straight-forward. As for packet drops due to misbehavior in the network, we assume that each selfish node $i \in M$ will drop a data packet forwarded through it with probability $P_{selfish}$. We assume a communication environment where detection and isolation of misbehaving nodes is crucial to the survival of the network. Typically, this is the case when the negative

impact of node selfishness (modeled as $P_{selfish}$) on network performance is much greater than deteriorating channel and/or network conditions (modeled as P_{drop}).

Consider, for now, symmetric routes in the network (we recognize the existence of asymmetric routes; as we will see later, as long as a reasonable proportion of all routes is symmetric, it is possible to accurately detect infected routes and misbehaving nodes). Define $R^{s\leftrightarrow d} = \{h_0,...,h_k\}$ as an ordered set of nodes representing the route from source node $s = h_0$ up to the last node h_k before the destination node d ($k+1$ is the number of hops on the route). Let the ordering relationship on $R^{s\leftrightarrow d}$ be $\forall x, y \in R^{s \leftrightarrow d}, x < y \Leftrightarrow hop(x, s \leftrightarrow d) < hop(y, s \leftrightarrow d)$, where $hop(x, s \leftrightarrow d)$ is the number of hops between source node s and node x on the route from source node s to destination node d. Let $R_i^{s\leftrightarrow d} \subset R^{s\leftrightarrow d}$ be the portion of the route $R^{s\leftrightarrow d}$ between source node s and destination node d that starts at node h_i and ends at node h_k. Hence, $R_i^{s\leftrightarrow d} = \{h_i,...,h_k\}$ $0 \le i \le k$ ($k-i+1$ is the number of hops on $R_i^{s\leftrightarrow d}$). A route $R_i^{s\leftrightarrow d}$ is considered "infected" if $\exists h_j \in R_i^{s\leftrightarrow d}$ such that $h_j \in \mathrm{M}$, $i < j \le k$; otherwise it is "clean."

We assume some mechanism is available to provide feedback on end to end packet delivery. If a reliable transport layer protocol such as TCP is adopted, this feedback is provided at the transport layer in the form of TCP acknowledgements. Otherwise, application layer acknowledgements could be used.

2. Detection of Infected Routes

2.1. Overview of Sequential Probability Ratio Test (SPRT)

Consider two hypotheses H_1 and H_0, where either H_1 or H_0 is true but not both, and two corresponding conditional probabilities, $P[x \mid H_1]$ and $P[x \mid H_0]$. To decide whether H_1 or H_0 is true we make a sequence of observations $x_1, x_2, \ldots$. For each observation $x_n, n \geq 1$, if $P[x_n \mid H_0] \neq 0$ we calculate the ratio $\frac{P[x_n \mid H_1]}{P[x_n \mid H_0]}$ and accumulate the value $T_n = T_{n-1} * \frac{P[x_n \mid H_1]}{P[x_n \mid H_0]}$, where $T_0 = 1$. We then examine the value of T_n. If T_n is large, it implies that the sequence of observations $x_1, \ldots, x_n$ made so far are more likely to have been generated under H_1 than under H_0. If T_n is small, the converse is true. If T_n is not sufficiently small or large to choose between H_1 and H_0 we make another observation x_{n+1}.

In making a decision between H_1 and H_0, it is possible to erroneously decide that H_1 is true while in reality H_0 is true (a false positive), or that H_0 is true while in reality H_1 is true (a false negative). To limit the probability of false positives by α and that of false negatives by β, we select two threshold values A and B, with $B < A$. After making a sequence of observations $x_1, \ldots, x_n$ a decision is made that H_1 is true if $T_n \geq A$, or that H_0 is true if $T_n \leq B$ and the test terminates. We make an additional observation x_{n+1} if $B < T_n < A$. It was shown in [54] that the values of A and B are

bounded by $A \leq \frac{1-\beta}{\alpha}$ and $B \geq \frac{\beta}{1-\alpha}$ and that by using values of $A = \frac{1-\beta}{\alpha}$ and $B = \frac{\beta}{1-\alpha}$ the test provides adequate level of precision.

2.2. Sequential Probability Ratio Test for Detection of Infected Routes

In this section we define our model for detection of infected routes. The mathematical formulation of the model considers routes to be symmetric. While we recognize the existence of asymmetric routes, we will later see that as long as a reasonable proportion of routes are symmetric our model works well, achieving high accuracy in detection of infected routes. We also note that ad hoc routing protocols typically attempt to construct routes that are symmetric [55] [56] [57].

Let H_1 be the hypothesis that a given route is infected and H_0 be the hypothesis that the route is clean. We need to identify which hypothesis holds. As mentioned earlier, we assume some mechanism is available to provide feedback on end to end packet delivery. If TCP is adopted, we use packet ACKnowledgements as observations of successful packet-delivery events and packet RETransmissions as observations of failed packet-delivery events. Let $T(i, R_i^{s \leftrightarrow d})$ be an evaluation of route $R_i^{s \leftrightarrow d}$ by node $h_i \in R^{s \leftrightarrow d}$. Upon encountering an ACK or a RET observation x_n at node h_i about route $R_i^{s \leftrightarrow d}$, the node evaluates

$$T(i, R_i^{s \leftrightarrow d})_n = \begin{cases} T(i, R_i^{s \leftrightarrow d})_{n-1} * \frac{P[ACK \mid H_1]}{P[ACK \mid H_0]} if\ x_n == ACK \\ T(i, R_i^{s \leftrightarrow d})_{n-1} * \frac{P[RET \mid H_1]}{P[RET \mid H_0]} if\ x_n == RET \end{cases} \quad \text{for} \quad n \geq 1 \quad , \quad \text{with}$$

$T(i, R_i^{s\leftrightarrow d})_0 = 1$. $P[ACK]$ and $P[RET]$ are the probabilities of observing an ACK or a RET on $R_i^{s\leftrightarrow d}$ at h_i, respectively. To determine whether the route $R_i^{s\leftrightarrow d}$ is infected or clean, node $h_i \in R^{s\leftrightarrow d}$ performs the following test: if $T(i, R_i^{s\leftrightarrow d})_n \geq A$ a conclusion is made that H_1 is true, and if $T(i, R_i^{s\leftrightarrow d})_n \leq B$ a conclusion is made that H_0 is clean. Note that some asymmetric routes may result in false negatives (infected routes erroneously classified as clean), or false positives (clean routes erroneously classified as infected). We will see in section 4.2 that as long as a reasonable proportion of routes are symmetric the impact of such cases on the system's accuracy is insignificant. In the next section we will determine how $T(i, R_i^{s\leftrightarrow d})$ is evaluated.

2.3. Evaluating $T(i, R_i^{s\leftrightarrow d})$

For each instance of a data packet transmitted by a source node of a flow, our model considers $P[ACK] + P[RET] = 1$ at any intermediate node on a route, and that ACK and RET events are mutually exclusive for this packet instance. In other words, the only two possible fates of an instance of a data packet transmitted by the source node are: successful delivery to the destination, and hence an ACK is generated by the destination in response, or packet loss and a RET is generated by the source. Hence, at any node h_i on a route $R^{s\leftrightarrow d}$ either an ACK or a RET should be observed for each instance of a data packet sent by node s, but not both. This is supported by the following assumptions:

- Any observation $x_n == ACK$ witnessed by a node h_i on $R^{s\leftrightarrow d}$ will also be witnessed by all nodes h_j where $k \geq j > i$. By the same token, any observation

$x_n == RET$ witnessed by a node h_i on $R^{s\leftrightarrow d}$ will also be witnessed by all nodes h_j where $0 \le j < i$.

- Each node h_i on a route $R^{s\leftrightarrow d}$ performs the test $T(i, R_i^{s\leftrightarrow d})$ to evaluate $R_i^{s\leftrightarrow d}$ only, and not the complete route $R^{s\leftrightarrow d}$. Hence any RET event for an instance of a data packet where an ACK has already been observed at node h_i (i.e. the ACK might have been dropped on $R^{s\leftrightarrow d} \setminus R_i^{s\leftrightarrow d}$) is treated by node h_i as a new data packet.

Accordingly, we can say that $P[ACK \mid H_1] + P[RET \mid H_1] = 1$ and that $P[ACK \mid H_0] + P[RET \mid H_0] = 1$.

Consider an ACK or RET observation made by node $h_i \in R^{s\leftrightarrow d}$ about a data packet sent on $R_i^{s\leftrightarrow d}$. $P[ACK] = P[Data\_Delivered] * P[ACK\_Observed \mid Data\_Delivered]$, where $P[Data\_Delivered]$ indicates the probability that the data packet was delivered at the destination. This means that the data packet was not dropped by any node on the route $R_i^{s\leftrightarrow d}$ due to network environment factors or as a result of selfish misbehavior. On the other hand, $P[ACK\_Observed \mid Data\_Delivered]$ indicates the probability that the ACK corresponding to a successfully delivered data packet was observed at node h_i, which means that it was not dropped by any node on the route $R_i^{s\leftrightarrow d}$ due to network environment factors or as a result of selfish misbehavior.

The value of $P[Data\_Delivered]$ as well as the value of $P[ACK\_Observed \mid Data\_Delivered]$ for a node $h_i \in R^{s\leftrightarrow d}$ are affected by the

number of selfish nodes on the route $R_i^{s\leftrightarrow d}$. The higher the number of selfish nodes on the route the lower the values of $P[Data\_Delivered]$ and $P[ACK\_Observed \mid Data\_Delivered]$. Let $a = \mid R_i^{s\leftrightarrow d} \cap \mathrm{M} \mid$ be the actual number of selfish nodes on the route $R_i^{s\leftrightarrow d}$, noting that $a = 0$ if the route is clean and $1 \le a \le \min(k - i, \mid \mathrm{M} \mid)$ otherwise. Let $P_{\Pi(selfish)} = 1 - (1 - P_{selfish})^a$ be the aggregate impact of selfishness on infected route $R_i^{s\leftrightarrow d}$. Accordingly, $P[Data\_Delivered] = (1 - P_{drop})^{k-i} * (1 - P_{\Pi(selfish)})$ and $P[ACK\_Observed \mid Data\_Delivered] = (1 - P_{drop})^{k-i+1} * (1 - P_{\Pi(selfish)})$. We can then calculate $P[ACK] = (1 - P_{drop})^{2(k-i)+1} * (1 - P_{\Pi(selfish)})^2$. Hence, we can find the values of $P[ACK \mid H_0]$, $P[ACK \mid H_1]$, $P[RET \mid H_0]$, and $P[RET \mid H_1]$ as follows:

- If $R_i^{s\leftrightarrow d}$ is clean then $P[ACK \mid H_0] = (1 - P_{drop})^{2(k-i)+1}$ and $P[RET \mid H_0] = 1 - (1 - P_{drop})^{2(k-i)+1}$.
- If $R_i^{s\leftrightarrow d}$ is infected then $P[ACK \mid H_1] = (1 - P_{drop})^{2(k-i)+1}(1 - P_{\Pi(selfish)})^2$ and $P[RET \mid H_1] = 1 - [(1 - P_{drop})^{2(k-i)+1}(1 - P_{\Pi(selfish)})^2]$.

Hence, given observation x_n at node h_i about route $R_i^{s\leftrightarrow d}$:

- If $x_n == ACK : T(i, R_i^{s\leftrightarrow d})_n = T(i, R_i^{s\leftrightarrow d})_{n-1}(1 - P_{\Pi(selfish)})^2$
- If $x_n == RET : T(i, R_i^{s\leftrightarrow d})_n = T(i, R_i^{s\leftrightarrow d})_{n-1} * \frac{1 - [(1 - P_{drop})^{2(k-i)+1}(1 - P_{\Pi(selfish)})^2]}{1 - (1 - P_{drop})^{2(k-i)+1}}$

3. Performance Analysis

We now evaluate the model discussed in the previous section to assess its ability to accurately distinguish between infected and clean routes. We realize an implementation of the model that does not require perfect knowledge of P_{drop}, $P_{\Pi(selfish)}$, or the exact number of hops on a route and implement it using ns-2 to evaluate its accuracy.

3.1. Model Implementation

We realize an implementation of the model that estimates model parameters as follows:

- P_{drop}: Each node monitors the network conditions surrounding it for packet drop and packet forward events and estimates the value of P_{drop} accordingly.
- $P_{\Pi(selfish)}$: The value of $P_{\Pi(selfish)}$ is not known because the exact number of misbehaving nodes (i.e. the value of a in the model) on a route is not known. However, we assume that $P_{\Pi(selfish)}$ is a network parameter that is set according to the network objective. Let $\widetilde{P}_{\Pi(selfish)}$ be the value of this network parameter and let $\overline{P}_{\Pi(selfish)}$ be the actual value of $P_{\Pi(selfish)}$ on a route. $\widetilde{P}_{\Pi(selfish)}$ is set according to the aggregate level of selfishness on a route that the network can tolerate. For example, if the value of $\widetilde{P}_{\Pi(selfish)}$ is set to $\widetilde{P}_{\Pi(selfish)} = 50\%$ then the objective of the network is to detect infected routes where the aggregate selfishness on the route

$\overline{P}_{\Pi(selfish)} \geq 50\%$. This also implies that the network can tolerate infected routes where the aggregate selfishness $\overline{P}_{\Pi(selfish)} < 50\%$.

- The number of hops from a node to a route's destination is available to the node in its routing table. The number of hops from a route's destination to a node may not be known with certainty. A first order approximation is to assume that the *total number of hops on the route = 2 * the number of hops on the forward route*.
- We consider that some routes may be constructed asymmetrically by the routing protocol.

An evaluation record $T(i, R_i^{s\leftrightarrow d})$ is maintained for each active route[1] $R_i^{s\leftrightarrow d}$ at a node $h_i \in R^{s\leftrightarrow d}$. The record is initialized and updated at node h_i for each observation x_n about $R_i^{s\leftrightarrow d}$ as discussed in section 3.2. Whenever $R_i^{s\leftrightarrow d}$ is updated in the routing table of node h_i the value of $T(i, R_i^{s\leftrightarrow d})$ is reset to its initial value. Hence, each instance of a route is evaluated independently. The model's A and B thresholds are maintained at each node and a decision whether a route is infected or clean is made as in section 3.2.

Table 3 Model Parameters

α	β	A	B
10%	10%	9	0.11

3.2. Evaluation

In this section we evaluate our model implementation. Our goal is to assess the ability of the model implementation to accurately distinguish between infected and clean

[1] A route is active if its routing table entry has not expired.

routes given its estimation of the model's parameters. Since $P_{\Pi(selfish)}$ is an important model parameter whose value is set according to the network objective, we also evaluate how the increase in $\overline{P}_{\Pi(selfish)}$, given a particular value of $\widetilde{P}_{\Pi(selfish)}$, impacts the accuracy of distinguishing between infected and clean routes. Accordingly, we use two sets of simulation settings:

- *Setting S1*: $\overline{P}_{\Pi(selfish)} = 75\%$ and $\widetilde{P}_{\Pi(selfish)} = 50\%$.
- *Setting S2*: $\overline{P}_{\Pi(selfish)} = 100\%$ and $\widetilde{P}_{\Pi(selfish)} = 50\%$.

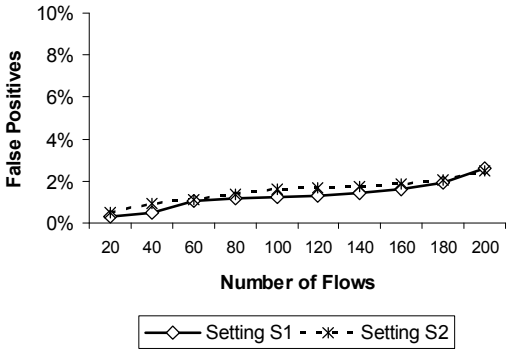

Figure 8 Model Accuracy: False Positives

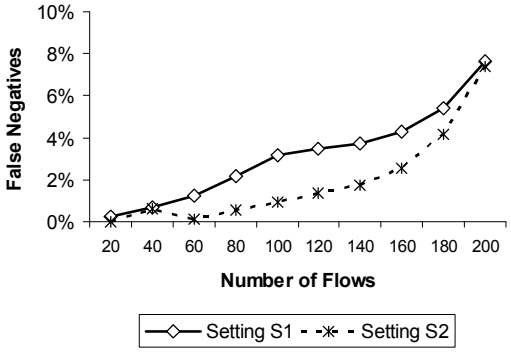

Figure 9 Model Accuracy: False Negatives

For both settings we adopt the model parameter values shown in Table 1. For each setting we run 250 ns-2 simulations using 5 different TCP traffic loads (each traffic load is run 50 times with flow source and destination pairs selected at random for each run). Traffic is generated for each flow using a CBR (Constant Bit Rate) traffic generator that generates approximately 300 data packets per flow. The network considered is composed of 64 nodes arranged in an 8X8 grid. The communication range is set such that each node has 4 neighbors, with the exception of edge nodes, which have 3 neighbors, and corner nodes, which have 2 neighbors. The network adopts IEEE 802.11

[58] for medium access and AODV [55] for routing. We use false positives and false negatives as defined in [54] as the basis for accuracy evaluation. We observe the decisions made by each node (i.e. H_1 or H_0) and examine the false positives and false negative that result from all decisions made by all nodes.

We compare the false positives and false negatives resulting from each setting of the model implementation to the theoretical bounds of α and β. Figure 8 shows that false positives in both settings are bounded by α under all traffic loads. It also shows that false positives are almost identical for both settings. False positives occur when a clean route is identified as infected due to retransmissions on the route. Since the route is actually clean, the value of $\overline{P}_{\Pi(selfish)}$ does not impact retransmissions. Hence, false positives are likely to be the same for a given value of $\widetilde{P}_{\Pi(selfish)}$ irrespective of the value of $\overline{P}_{\Pi(selfish)}$. With respect to false negatives, Figure 9 shows that settings *S1* and *S2* are bounded by β. Figure 9 shows that false negatives decrease as the value of $\overline{P}_{\Pi(selfish)}$ increases. A false negative occurs when an infected route is falsely identified as clean because of ACK events observed on the route. As $\overline{P}_{\Pi(selfish)}$ increases the likelihood that an ACK event is observed on an infected route decreases. Hence, false negatives decrease.

Simulation results show that even though the model parameters are estimated with some error, false negatives and false positives remain bounded by α and β respectively. We noticed that given a value of $\widetilde{P}_{\Pi(selfish)}$ that the system can tolerate, system accuracy increases as $\overline{P}_{\Pi(selfish)} - \widetilde{P}_{\Pi(selfish)}$ increases. In our simulations, we considered that some

routes may be asymmetric. An assessment of the proportion of asymmetric routes in the simulations revealed that between 10%-20% of all routes were constructed asymmetrically by the routing protocol. The accuracy achieved by the model suggests that as long as the proportion of symmetric routes is reasonably high, asymmetric routes have insignificant impact on the model's accuracy. Since routing protocols typically attempt to construct routes symmetrically [55] [56] [57], it is reasonable to assume that a large proportion of routes will be symmetric.

4. Detecting Selfish Nodes from Infected Routes

We now consider identifying selfish nodes from the routes reported as infected by the model. While the detection of infected routes reduces the impact of misbehavior (by avoiding these routes, for example), it does not fully eliminate the impact of misbehavior (new infected routes may emerge). To eliminate the impact of misbehavior it is important to detect and react to misbehaving nodes by isolating them from the network, for example. We propose two approaches to detect selfish nodes on infected routes: a centralized approach and a localized approach. The centralized approach assumes knowledge of all infected routes in the network, which is the case if a central authority exists to which nodes report routes they identify as infected, or if nodes exchange such information with one another. The localized approach assumes that each node knows only about routes it identified as infected and only knows about its next hop neighbors on these routes. To assess the effectiveness of each approach we use three metrics: exposure (proportion of selfish nodes identified as such), false positives (proportion of cooperative nodes falsely identified as selfish), and false negatives (proportion of selfish nodes falsely

identified as cooperative). In defining these metrics, we use the indicator function

$$1_{\{A\}} = \begin{cases} 1, if\ A\ is\ true \\ 0, if\ A\ is\ false \end{cases}.$$

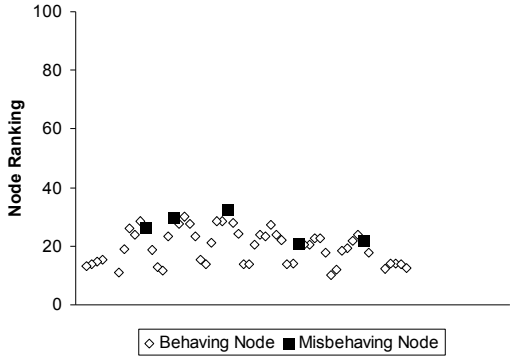

Figure 10 Ranking of MFN - 60 flows -

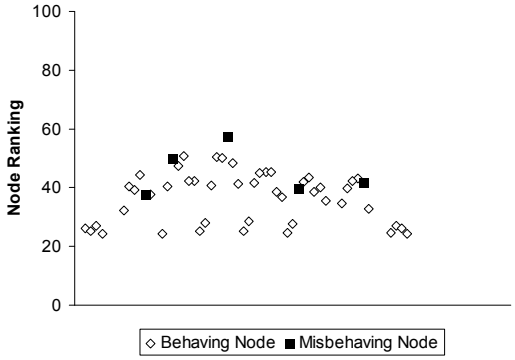

Figure 11 Ranking of MFN - 100 flows -

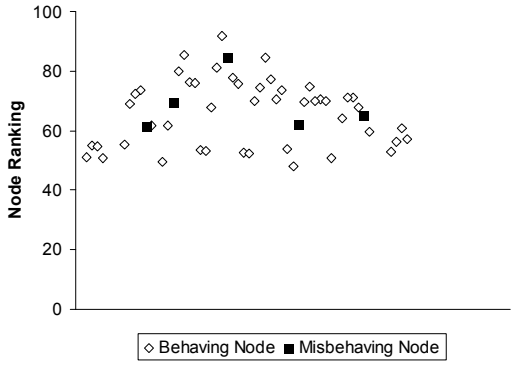

Figure 12 Ranking of MFN - 200 flows -

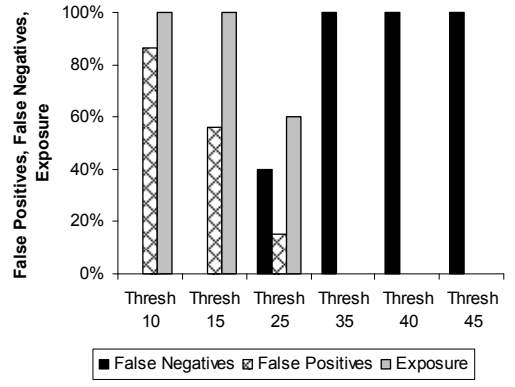

Figure 13 False Positives, False Negatives, and Exposure for 60 flows

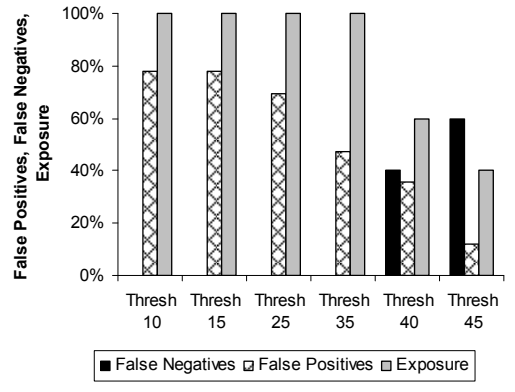

Figure 14 False Positives, False Negatives, and Exposure for 100 flows

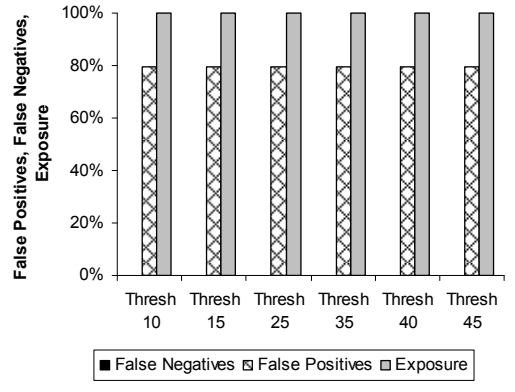

Figure 15 False Positives, False Negatives, and Exposure for 200 flows

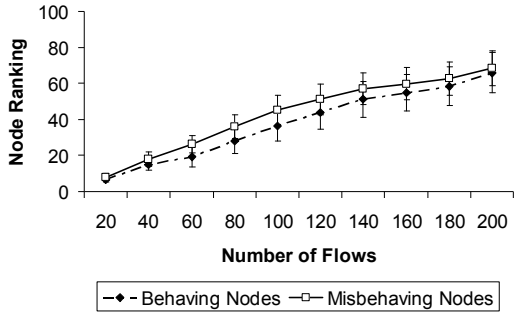

Figure 16 Average and standard deviation of rankings of behaving and misbehaving nodes (centralized approach)

4.1. Centralized Approach

This approach assumes knowledge of all routes identified as infected by all nodes in the network. We construct a sorted list of nodes that occur most frequently in infected routes. Nodes positioned high in the list are identified as selfish. This approach works as follows:

- Define $R_{infected} = \bigcup_{i \in \mathrm{N}} R^{\mathrm{i}}_{infected}$ as the set of all routes identified as infected, where $R^{\mathrm{i}}_{infected}$ is the set of infected routes identified by node $i, \forall\, i \in \mathrm{N} \setminus \mathrm{M}$. Let $RI = |R_{infected}|$.

- Let $G(i) = \sum_{r \in R_{infected}} 1_{i \in r}$ be the ranking of node $i, \forall\, i \in \mathrm{N}$ based on the frequency of its appearance in infected routes. For example, $G(i) = 2$ if node i appears in two routes in $R_{infected}$.

- Using G, we construct MFN, the list of *most frequent nodes* in $R_{infected}$. Most if not all selfish nodes should be in this list.

Figure 10, Figure 11, and Figure 12 show a scatter of the average ranking of the nodes in MFN under different traffic loads for $P_{\Pi(selfish)} = 100\%$ ($P_{\Pi(selfish)} = 75\%, 50\%$ show similar results). The set of R_{infected} was constructed out of 50 simulation runs. The figures show that it is difficult to draw a clear line between the rankings of selfish and cooperative nodes. Figure 16 further illustrates this observation by comparing the average ranking of selfish and cooperative nodes indicating that the two are very close in value. Given TH, the threshold ranking between selfish and cooperative nodes, we define the following metrics based on the metrics defined in chapter III:

- Exposure Ratio: $\frac{\sum_{i \in \text{MFN}} 1_{G(i) \geq TH \wedge i \in \text{M}}}{|\text{M}|}$
- False Positives: $\frac{\sum_{i \in \text{MFN}} 1_{G(i) \geq TH \wedge i \in \text{N} \setminus \text{M}}}{|\text{N} \setminus \text{M}|}$
- False Negatives: $\frac{\sum_{i \in \text{MFN}} 1_{G(i) < TH \wedge i \in \text{M}}}{|\text{M}|}$

Figure 13, Figure 14, and Figure 15 show the values of exposure, false positives and false negatives for six different values of TH under different traffic loads. Note that a good mechanism for identifying misbehaving nodes should achieve high exposure, low false positives, and low false negatives. From the figures, it is clear that it is not possible to get both high exposure and low false positives using the centralized approach. To achieve 100% exposure, false positives was at a minimum of 55.93% at 60 flows, 47.46% at 100 flows, and 79.66% at 200 flows. This is because the rankings of selfish and cooperative nodes are within close range, which makes it hard to identify selfish nodes with reasonable accuracy.

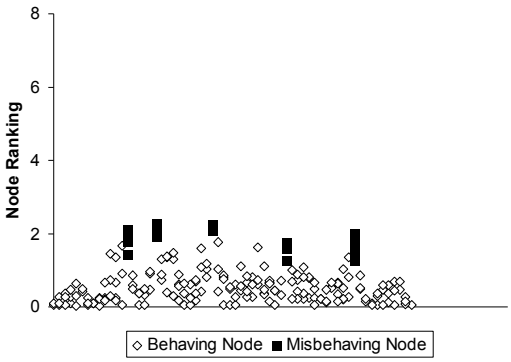

Figure 17 Ranking of nodes for their neighbors - 60 flows -

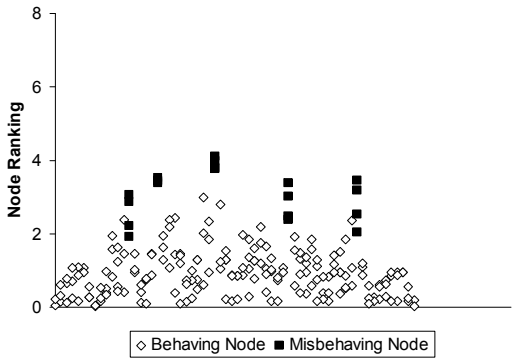

Figure 18 Ranking of nodes for their neighbors - 100 flows -

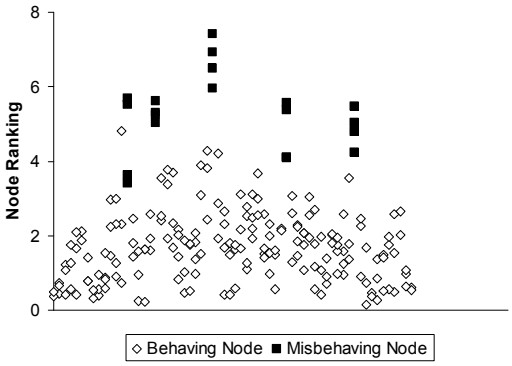

Figure 19 Ranking of nodes for their neighbors - 200 flows -

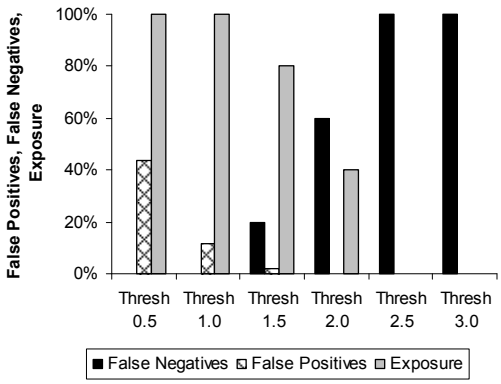

Figure 20 False Positives, False Negatives, and Exposure for 60 Flows

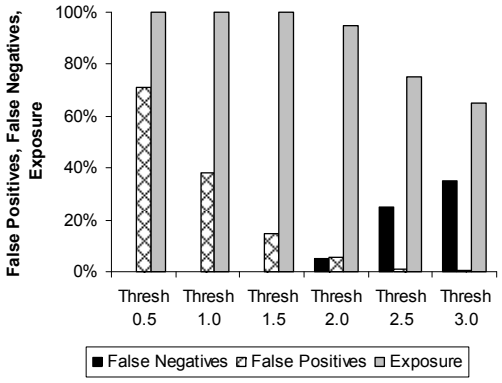

Figure 21 False Positives, False Negatives, and Exposure for 100 Flows

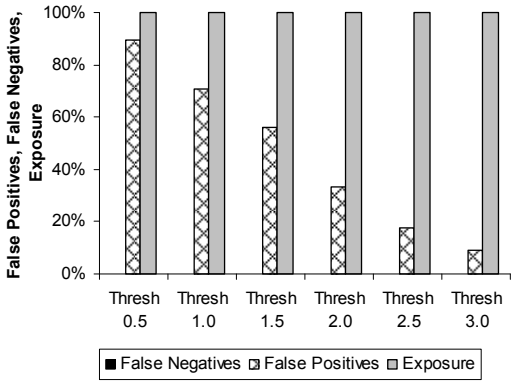

Figure 22 False Positives, False Negatives, and Exposure for 200 Flows

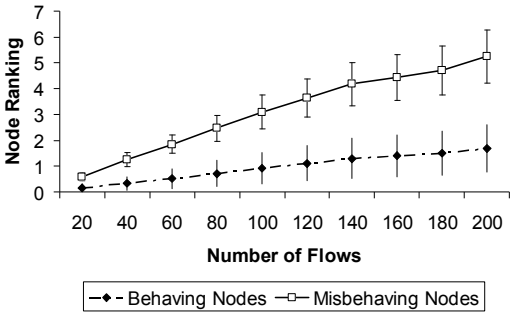

Figure 23 Average and standard deviation of rankings of behaving and misbehaving nodes (localized approach)

4.2. Localized Approach

The localized approach assumes that each node $i, \forall i \in \mathrm{N} \setminus \mathrm{M}$ has a set of routes $R^{\mathrm{i}}_{infected}$ that it identified as infected and that it knows only its next hop neighbor on each of these routes. Let $d(i, j)$ be the distance in hops between nodes i and $j, \forall i, j \in \mathrm{N}$. Each node ranks its next hop neighbors on these routes as follows:

- Let $R^{\mathrm{i}}_{infected}$ be the set of next hop neighbors on infected routes identified by a node $i, \forall i \in \mathrm{N} \setminus \mathrm{M}$.
- Define $G_i(j) = \sum_{r \in R^{i}_{\mathrm{inf}\,ected}} 1_{j \in r}$, the ranking assigned by node i to node j based on the frequency of its appearance as a next hop to node i in $R^{\mathrm{i}}_{infected}$, $\forall i \in \mathrm{N} \setminus \mathrm{M}, j \in \mathrm{N}$ where $d(i,j) = 1$.

Figure 17, Figure 18, and Figure 19 show a scatter of the rankings of nodes by their neighbors averaged over 50 simulation runs under different traffic loads for $P_{\Pi(selfish)} = 100\%$ ($P_{\Pi(selfish)} = 75\%, 50\%$ show similar results). The figures show that

the rankings of selfish nodes are higher than those of cooperative nodes and the difference between both becomes clearer as the traffic load increases. Figure 23 further illustrates this observation by plotting the average ranking of a selfish node versus that of a cooperative node. Using six different values of TH (threshold ranking between selfish and cooperative nodes), we evaluate the localized approach under different traffic loads by measuring its exposure, false positives, and false negatives based on the metrics defined in chapter III as follows:

- Exposure: $\frac{\sum_{i\in N\setminus M}\sum_{j\in M} 1_{d(i,j)=1 \wedge G_i(j)\geq TH}}{\sum_{i\in N\setminus M}\sum_{j\in M} 1_{d(i,j)=1}}$
- False Positives: $\frac{\sum_{i\in N\setminus M}\sum_{j\in N\setminus M} 1_{d(i,j)=1 \wedge G_i(j)\geq TH}}{\sum_{i\in N\setminus M}\sum_{j\in N\setminus M} 1_{d(i,j)=1}}$
- False Negatives: $\frac{\sum_{i\in N\setminus M}\sum_{j\in M} 1_{d(i,j)=1 \wedge G_i(j)< TH}}{\sum_{i\in N\setminus M}\sum_{j\in M} 1_{d(i,j)=1}}$

Figure 20, Figure 21, and Figure 22 show that high exposure and low false positives can be achieved in the localized approach. To achieve 100% exposure, false positives was at a minimum of 11.48% at 60 flows, 14.97% at 100 flows, and 9.09% at 200 flows. The choice of TH plays an important role in determining the value of exposure and false positives. To maintain false positives below 10% for example, a threshold value of 1.5 for 60 flows should be used, 2.0 for 100 flows, and 3.0 for 200 flows. This corresponds to an exposure of 80% at 60 flows, 95% at 100 flows, and 100% at 200 flows. Accordingly, the value of TH can be a system parameter that is adapted according to the traffic load a node is experiencing.

5. Summary

In this chapter, we developed a model based on the Sequential Probability Ratio Test to characterize how nodes can differentiate between routes that include misbehaving nodes (infected routes) and routes that do not. An advantage of the model is that the number of observations required to evaluate a route need not be determined in advance, which suits well the dynamic nature of ad hoc networks. We then outlined a centralized and a localized approach to detect misbehaving nodes on infected routes identified by the model. Our evaluation shows that the localized approach is not only the better architectural choice for ad hoc networks but also results in a more accurate exposure of misbehaving nodes while incurring low false positives and low false negatives.

Chapter V. Autonomous Reputation Management System

Part of this work was publiched in the proceedings of the IEEE Mobiquitous 2005, San Diego, CA, pp 3 – 11. M. Tamer Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy "A Reputation-based mechanism for Isolating Selfish Nodes in Ad Hoc Networks."

1. Autonomous Reputation Management System

In this section we propose and analyze an *Autonomous Reputation Management System* (ARMS). ARMS is developed based on the Sequential Probability Ratio Test discussed in the previous section. ARMS relies on the principle of autonomy, where a node evaluates its neighbors' behavior using first-hand evaluation metrics. This is a departure from relying on second-hand metrics in the evaluation of node behavior, which has a number of drawbacks related to overhead, misreporting, and collusion [59]. While most existing work on reputation management for routing in ad hoc networks implicitly assume source routing [41] [33] [39], we developed a system that supports hop-by-hop routing and we show that by making simple, localized decisions, the network can achieve high exposure of misbehaving nodes with a low number of false positives and little communication overhead.

We analyze both the effectiveness and efficiency of our proposed system. Effectiveness relates to security metrics, such as how fast nodes are able to detect misbehavior, while efficiency relates to performance metrics, such as the overhead imposed by the system. We also conduct a comprehensive analysis of the benefit and cost of employing second-hand metrics for behavior evaluation as opposed to first-hand metrics only. While [59] advocated, based on a qualitative analysis, the use of first-hand metrics in order to avoid the drawbacks of second-hand metrics, [60] showed that employing second-hand metrics expedites misbehavior detection but did not discuss at

what cost. In our study, we analyze the benefit and cost of second-hand evaluation metrics compared to first-hand evaluation metrics using the effectiveness and efficiency metrics that we defined earlier. Our conclusions indicate that ARMS reduces the impact of misbehavior and promptly detects misbehaving nodes while producing very low false positives. We also conclude that, while relying on second-hand metrics expedites the detection and isolation of misbehaving nodes, the cost posed on the network due to misbehavior is high. The work in this chapter makes two main contributions: the introduction and evaluation of an autonomous reputation management system, and a comprehensive analysis of the benefit and cost of employing second-hand metrics as opposed to first-hand metrics in the proposed system in the presence of selfish and some forms of malicious behavior.

2. ARMS Description and Architecture

In ARMS, when a node forwards a packet through one of its neighbors, it holds this neighbor accountable for the successful delivery of the packet to the final destination. A node assigns a score to each of its neighbors based on the delivery of packets it forwards, through the neighbor, toward their final destination. This evaluation is executed by each node along a route, from the source node to the node that serves as the last hop on this route before the destination. For each successfully delivered packet, each node along the route increases the score of its next hop neighbor through which it forwarded the delivered packet. This rewards the neighbor for its good action, since it forwarded the packet successfully to the next hop towards the destination, and for its good decision, since it selected a reliable next hop to forward the packet through. Conversely, packet delivery failures result in a penalty, where each node along the route

(up to the node where the packet was lost) lowers the score of its next hop neighbor through which the lost packet was forwarded. The penalty applied by a node on its next hop neighbor is due to the neighbor's bad action if it failed to forward the packet successfully to the next hop towards the destination, or for its poor decision if it selected an unreliable next hop to forward the packet through. Packet delivery success or failure is inferred from destination feedback.

To discourage misbehavior and motivate nodes to build up their score, each node determines whether to forward or drop a packet based on the reputation score of the packet's previous hop (i.e. forwarding node). This motivates nodes to cooperate by forwarding packets for other nodes in order for other nodes to forward packets for them. A node whose score falls below a pre-determined threshold is identified as misbehaving. Once a node is independently identified as misbehaving by all its neighbors, all packets forwarded through it or originating at it will be discarded by these neighbors. The result is the isolation of the misbehaving node.

2.1. Design Objectives of ARMS

In the design of ARMS, our objectives can be summarized as follows:

- *Autonomous*: The most reliable reputation scores are those which are assigned based on one's own experience [59]. Hence, nodes in ARMS evaluate the behavior of other nodes independently relying on first-hand metrics only.
- *Low overhead*: Complex reputation systems may be too inefficient to use in ad hoc networks given nodes' resource limitations [59]. Therefore, we designed behavior evaluation in ARMS to rely only on nodes' own experiences, which exerts no communication overhead on the network.

- *Fully distributed*: A distributed solution is most adequate for the nature of ad hoc networks. ARMS is distributed in the sense that it is implemented at every node in the network without centralized control or a trusted third party.
- *Localized*: Localized solutions scale much better in a distributed environment. ARMS is localized since nodes are concerned with evaluating the behavior of their immediate neighbors only.
- *Low false positives*: False positives may have a negative impact on the network and the performance of any reputation management system. The objective of ARMS is to minimize false positives that may occur by introducing behavior verification.
- *Modular*: The architecture of ARMS defines different independent components where each is designed to handle a specific task within the larger system. This makes the system more flexible to additions, replacements, or modifications of components.

2.2. Architecture of ARMS

The architecture of ARMS was designed to conform to the components of the formal framework discussed in the Chapter III. It comprises five of the framework's components: auditor, behavior evaluator, behavior classifier, behavior verifier, and responder (Figure 24). The functions of each component are defined in terms of the parameters shown in Table 4.

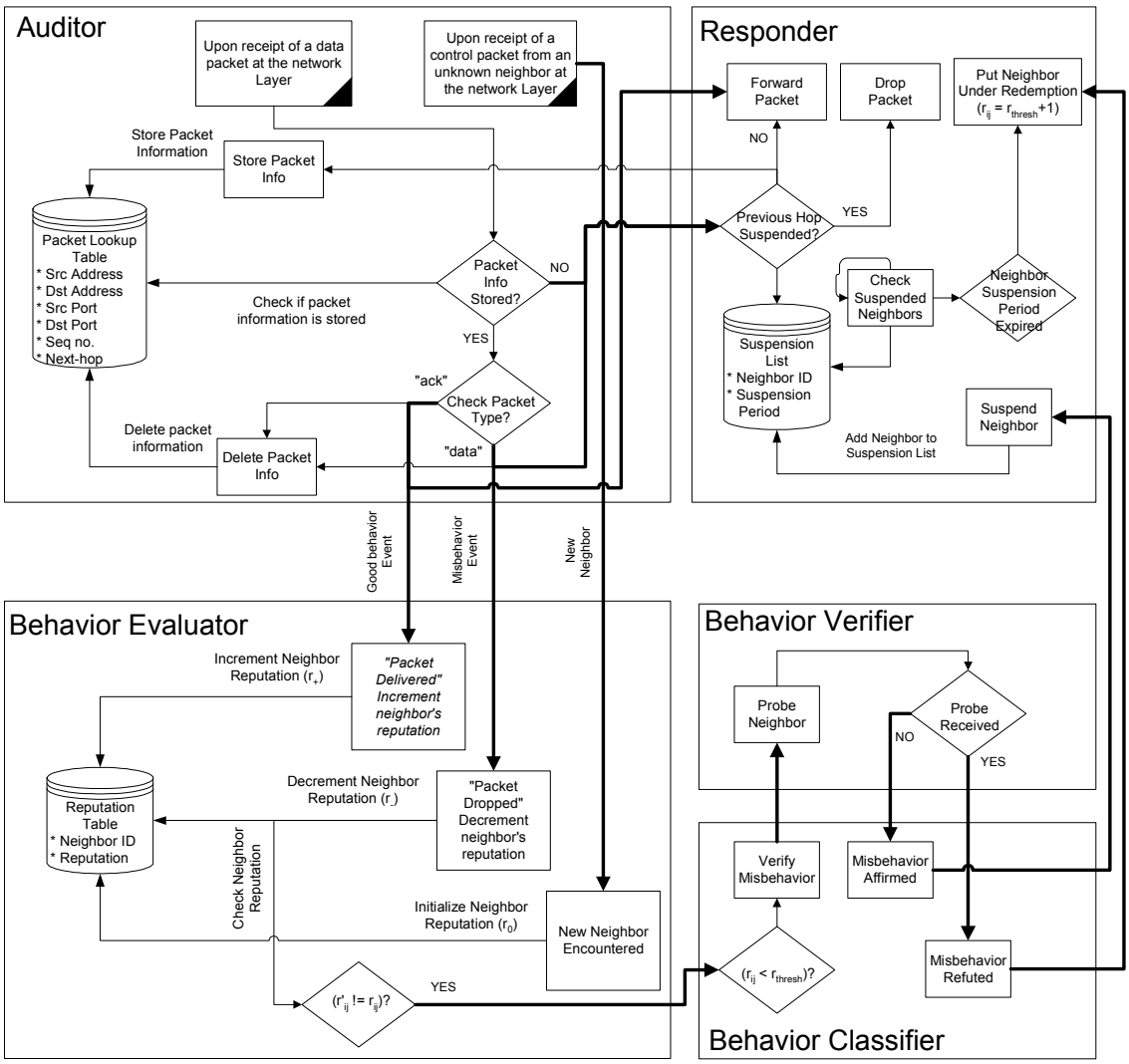

Figure 24 Architecture of ARMS:
The five components of ARMS and their interactions

Table 4 Parameters of ARMS
Note that $r_{\max} > r_0 > r_{thresh} > r_{\min}$.

Reputation Score (r_{ij})	Each node i maintains a reputation score r_{ij} for each immediate neighbor j $\{\forall i, j \in N : d(i,j) = 1\}$, where $d(i,j)$ is the distance in hops between nodes *i* and *j*.
Initial Reputation Score (r_0)	Upon encountering a new neighbor *j*, node *i* initializes its reputation score to $r_{ij} = r_0$.
Reputation Score Increment (r_+)	Upon encountering an indication of good behavior of node *j*, node *i* increments its reputation score r_{ij} by r_+.

Reputation Score Decrement ($r\_$)	Upon encountering an indication of bad behavior of node *j*, node *i* decrements its reputation score r_{ij} by $r\_$.
Maximum Reputation Score ($r_{\max}$)	Indications of good behavior of a node result in incrementing the reputation score of that node up to a maximum value of $r_{\max}$.
Minimum Reputation Score ($r_{\min}$)	Indications of bad behavior of a node result in decrementing the reputation score of that node up to a minimum value of $r_{\min}$.
Reputation Threshold (r_{thresh})	A node whose reputation score falls below r_{thresh} is marked as misbehaving.

2.2.1. Auditor

Within ARMS, this component gathers only observations that can be made locally about inbound and outbound traffic (i.e. first-hand observations). Each node maintains a lookup table that stores information about recent outbound data packets forwarded through it, including sequence numbers, source and destination addresses and port numbers, and the address of the next hop. Upon receiving an inbound packet at node *i*, the *auditor* takes action based on the packet type. If the received packet was a data packet, node *i* checks whether the packet is a retransmission (indicated by the presence of stored packet information). If it is, node *i* interprets this event as a misbehavior event by the neighboring node through which the original packet was forwarded. It then notifies the *behavior evaluator* component accordingly. The information related to the original outbound data packet stored in the lookup table is then updated. On the other hand, if the packet was an acknowledgement relayed through node *j*, node *i* verifies whether node *j* was the node through which the corresponding data packet was forwarded (this information is stored in the lookup table). If so, node *i* interprets this event as a good behavior event by node *j* and notifies the *behavior*

evaluator component accordingly. The information related to the acknowledged outbound packet stored in the lookup table is then deleted. Finally, if a control packet is received from a neighbor of which node *i* was unaware, node *i* interprets this event as a new neighbor event and notifies the *behavior evaluator* component accordingly.

2.2.2. Behavior Evaluator

The *behavior evaluator* component is responsible for defining a first-hand metric to be used for node behavior evaluation. The metric is defined in terms of the notification events received from the *auditor component*. Using parameters from Table 4, the first-hand metric is defined at node *i* as follows:

- If a new node notification event about a neighboring node *j* is received, node *i* creates an entry for this neighbor in its reputation table and initializes its reputation score to $r_{ij} = r_0$.
- Upon receiving a misbehavior notification event about neighboring node *j*, node *i* updates the reputation score of node *j* to $r_{ij} = \begin{cases} r_{ij} - r_-, & \text{if } r_{ij} - r_- > r_{\min} \\ r_{\min}, & \text{if } r_{ij} - r_- \le r_{\min} \end{cases}$.
- Upon receiving a good behavior notification event about neighboring node *j*, node *i* updates the reputation score of node *j* to $r_{ij} = \begin{cases} r_{ij} + r_+, & \text{if } r_{ij} + r_+ < r_{\max} \\ r_{\max}, & \text{if } r_{ij} + r_+ \ge r_{\max} \end{cases}$.

In all cases, if r_{ij} is updated, the *behavior classifier* at node *i* is notified of the new value of node *j*'s reputation score.

2.2.3. Behavior Classifier

The *behavior classifier* receives updates about changes to the reputation score of neighboring nodes from the *behavior evaluator*. This component then evaluates this

change and classifies nodes accordingly. Currently, ARMS uses a simple binary classification policy. Nodes are classified according to their behavior as either cooperative or misbehaving. Upon receiving an update to the reputation score r_{ij} of a node *j* at node *i*, this component acts in conjunction with the *behavior verifier* component. If $r_{ij} < r_{thresh}$, node *j* is classified as a misbehaving node and a notification is sent to the *behavior verifier* component to verify this classification. If the *behavior verifier* reaffirmed the node's classification as misbehaving, the *responder* component is notified about the misbehaving node and the affirmation by the *behavior verifier*. If the *behavior verifier* refuted the node's classification as misbehaving, the *responder* component is notified about the misbehaving node and the refutation by the *behavior verifier*. If $r_{ij} \geq r_{thresh}$, on the other hand, then node *j* is classified as a cooperative node.

2.2.4. Behavior Verifier

In ARMS, we employ a verification policy where nodes use probes to verify the behavior classification of their immediate neighbors. Probes are sent covertly along with regular traffic in a loopback from the verifying node to the node under verification. The node under verification is expected to send the probe back to the verifying node. Probes do not look different from regular data packets. A verifying node may use any data packet where it is the intended receiver (i.e. the verifying node's address is the destination of the data packet) as a probe. It would send this data packet through the neighbor under verification acting as a forwarding node[2]. Since probes are regular data packets, a malicious node should not be able to distinguish between a probe and a regular data packet.

[2] A verifying node may forge its MAC address when probing a node under verification, to further disguise the probe as "regular" traffic.

Based on our verification policy, probes are only sent when needed and are not sent periodically. A node that was recently flagged as misbehaving is probed once to ascertain its misbehavior. If the probe packet was received back at the verifying node then the *behavior classifier* component is notified that the node under verification is not misbehaving as claimed. If not, then the *behavior classifier* confirms that the node is misbehaving. Note that a lost probe could result in a false positive. However, the impact of such cases on the system's performance is low since probes are only sent along a single hop.

2.2.5. Responder

The architecture of ARMS integrates a reaction component rather than decoupling it from behavior evaluation. This is motivated by the fact that the lack of reaction to misbehavior impacts the capability of a reputation management system to accurately evaluate node behavior.

We employ a disciplinary response policy by placing the nodes whose misbehavior was confirmed by the *behavior verifier* under suspension for a period of time. While under suspension, all participation requests from/to the misbehaving node with respect to the routing functionality are denied. Thus, all RREQ, RREP, and traffic packets forwarded from the node under suspension are dropped. Also, no packets are forwarded through the misbehaving node. Nodes whose suspension period expired and nodes whose misbehavior was refuted by the *behavior verifier* are put under redemption. This allows these nodes to be carefully re-admitted into the network. The goal is to reduce the impact of false positives. Nodes' behavior might seem bad at times due to

external influences such as congestion or proximity to misbehaving nodes (the impact of misbehaving nodes on false positives is discussed in a later section).

A re-admitted node will be closely monitored. If node *j* was just re-admitted by node *i*, its reputation score will be set to $r_{ij} = r_{thresh} + 1$. If node *j* maintained its misbehavior and dropped the first packet forwarded through it by node *i*, its reputation score will be updated to $r_{ij} = r_{ij} - r_{\_} < r_{thresh}$. Thus, it will immediately be re-identified as misbehaving and will be put under suspension for a longer suspension period. Node *j* needs to successfully deliver packets forwarded through it by node *i* to increase its reputation score away from r_{thresh}.

3. Evaluation of ARMS

Our evaluation of ARMS aims to achieve two objectives. The first objective is to study the effectiveness of ARMS in mitigating node misbehavior. We also study the false positives generated by ARMS and the sensitivity of false positives to proximity to misbehaving nodes, false negatives, and other false positives. The second objective is to conduct an efficiency analysis of ARMS, studying the overhead it exerts on the network. We study the storage overhead of ARMS and the impact of the verification policy on the performance of ARMS. Our evaluation utilizes the network simulator NS-2 to simulate an ad hoc network containing misbehaving nodes.

3.1. Simulation Setup

We simulate an ad hoc network of *64* nodes arranged in an *8X8* grid. The communication range is set such that each node has *4* neighbors, with the exception of edge nodes, which have *3* neighbors, and corner nodes, which have *2* neighbors.

We randomly generate sets of *100, 200, 300, 400*, and *500* CBR (Constant Bit Rate) TCP flows, each sending 500Kbits of data. For each set, *100* simulations are executed, each for 600 seconds, with flow source and destination pairs selected at random for each run. Source and destination nodes are cooperative. We rely on TCP acknowledgements and retransmissions as indications of successful and failed packet delivery events, respectively.

Since ARMS uses a first-hand metric that is defined based on observations of destination-based acknowledgements, it is independent of the choice of the ad hoc routing protocol, and it can be applied to source-routing or hop-by-hop routing protocols. In our simulations, we select Ad-hoc On-demand Distance Vector (AODV) as our ad hoc routing protocol. AODV's specifications allow a node to maintain a blacklist of neighbors [55]. Neighbors identified as misbehaving (i.e. nodes whose reputation score is below the threshold) are included in this blacklist. All route requests (RREQs) and route replies (RREPs) forwarded by such neighbors will be ignored. Moreover, our protocol allows a node to trigger a RREQ if it detects a route entry in its routing table with a blacklisted next hop neighbor, thus eliminating all misbehaving nodes from routes.

To evaluate ARMS, we use two scenarios that differ in terms of the fraction of misbehaving nodes in the network. The first scenario assumes 5 selfish nodes, while the other assumes 10 selfish nodes. We will refer to these scenarios as the *5-selfish-node* scenario and *10-selfish-node* scenario throughout this document. Based on our pre-simulation analysis, reported in [61], we adopt simulation parameters $r_0 = 50$, $r_{\min} = 0$, $r_{\max} = 100$, $r_{thresh} = 35$, $r_+ = 1$, and $r_- = 2$. The suspension period of detected

misbehaving nodes is initially set to 10 seconds and is increased by 10 seconds every time a node moves from redemption to suspension. The suspension period is reduced by 10 seconds whenever packets forwarded through that node are acknowledged.

Note that we conducted a number of NS-2 simulations in [61] to assess the effect of node mobility on the performance of an earlier version of our proposed system. Consistent with the expectation that the performance of any reputation management system will deteriorate with mobility, our findings showed that as the average node speed increases exposure ratio and exposure rate will decrease. This is due to the decrease of the average interaction time between nodes as their speed increases. However, the extent of the misbehavior impact in terms of packet drops decreased as well.

3.2. Effectiveness Analysis

3.2.1. Reducing Impact of Misbehavior

Node misbehavior negatively impacts network performance. Reducing the extent of this impact is the goal of any reputation management system. In order to assess the effectiveness of ARMS in reducing the impact of misbehavior, we use packet drops by misbehaving nodes and route acquisition failures as measures of network performance in the presence and absence of ARMS. Our findings indicate that the negative impact of misbehavior on the network is significantly decreased by ARMS and that this decrease is a function of traffic load.

In the *5-selfish-node* and *10-selfish node* scenarios and the absence of ARMS, the ratio of packet drops due to misbehavior to the total number of packet drops remains constant as we increase the number of flows in the network from 100 to 500 flows

(Figure 25 and Figure 26). Moreover, an increase in the number of flows increases the average number of routes going through misbehaving nodes, which increases route acquisition failures (Figure 27 and Figure 28). In the absence of ARMS, increasing the number of flows amplifies the impact of misbehavior on the network.

As the number of flows increases, ARMS is more effective in improving the network's resilience to misbehavior. The increase in traffic increases interactions between nodes, which allows each node to investigate the behavior of its neighbors and evaluate the behavior of each more accurately. Whether the increase in interaction is in terms of an increase in interaction frequency within a fixed interaction time period, or an increase in interaction time given a fixed interaction frequency, nodes make use of these interactions to make a more accurate assessment of one another's behavior. Hence, they can distinguish between cooperative and misbehaving nodes and isolate the latter.

Figure 25 and Figure 26 show the ratio of packet drops due to misbehavior to the total number of packet drops is reduced when ARMS is introduced. In the absence of ARMS, packet drops due to misbehavior account for as much as 40% of total packet drops in the *5-selfish-node* scenario and 72% in the *10-selfish-node* scenario. The presence of ARMS reduces this ratio as a function of traffic load achieving, for 500 flows, a 79% reduction for the *5-selfish node* scenario, and a 74% reduction for the *10-selfish node* scenario (Figure 25 and Figure 26).

In the absence of ARMS, route acquisition failures increase proportionally to the increase in the number of flows and the misbehavior rate (Figure 27 and Figure 28). For 500 flows, route acquisition failures reached 23% in the *5-selfish node* scenario and 52% in the *10-selfish node* scenario. The implementation of ARMS significantly reduces this

negative impact, reducing route acquisition failures by 92% in the *5-selfish node* scenario and 84% in the *10-selfish node* scenario.

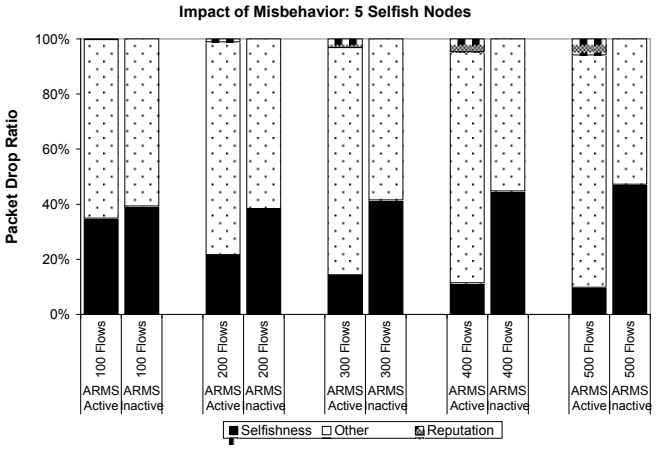

Figure 25 Impact of Misbehavior: Packet Drop Ratio

Compare packet drops due to misbehavior when ARMS is active vs. when it is inactive for 5 selfish nodes

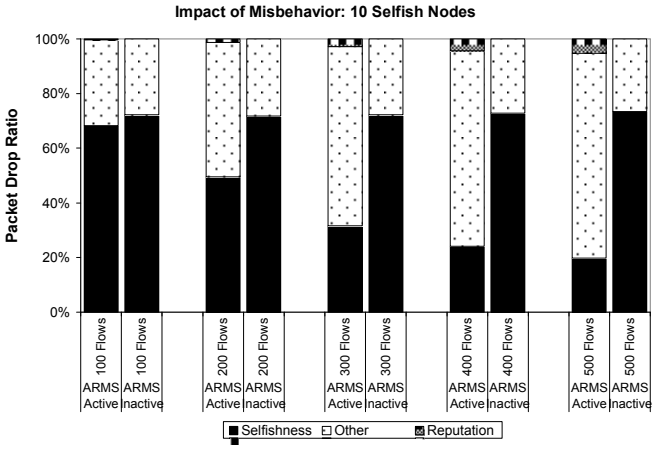

Figure 26 Impact of Misbehavior: Packet Drop Ratio

Compare packet drops due to misbehavior when ARMS is active vs. when it is inactive for 10 selfish nodes

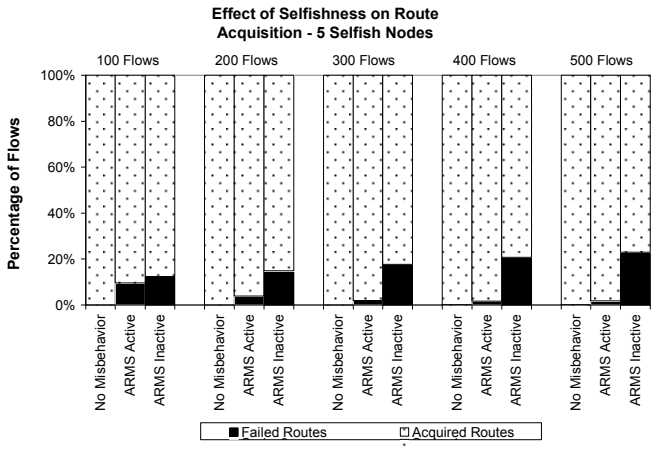

Figure 27 Impact of Misbehavior: Route Acquisition Failure

Compare the route acquisition ratio when no-misbehaving nodes exist and ARMS is inactive, when misbehaving nodes exist and ARMS is inactive, and when misbehaving nodes exist and ARMS is active for 5 selfish nodes.

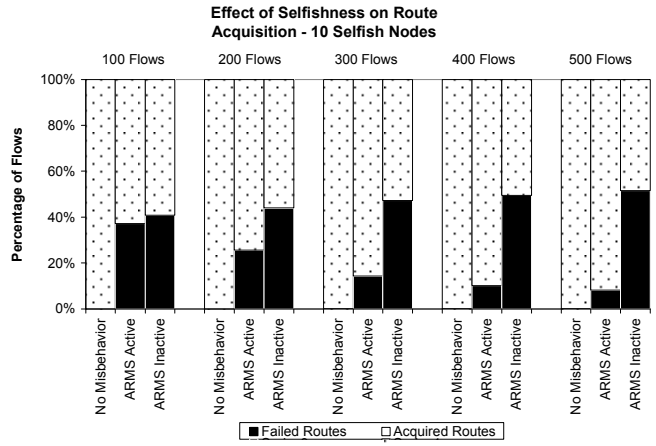

Figure 28 Impact of Misbehavior: Route Acquisition Failure

Compare the route acquisition ratio when no-misbehaving nodes exist and ARMS is inactive, when misbehaving nodes exist and ARMS is inactive, and when misbehaving nodes exist and ARMS is active for 10 selfish nodes.

3.2.2. Detecting Misbehaving Nodes

In measuring the performance of any reputation management system, its effectiveness depends upon its ability to detect misbehaving nodes in a timely manner. In our evaluation of ARMS, we measured the misbehavior detection in terms of exposure ratio, which measures the ability of the reputation management system to detect misbehaving nodes, and exposure rate, which measures the speed of the reputation management system in detecting misbehaving nodes. With respect to exposure ratio, we take into consideration that we employ a node redemption policy, so an unexposed node does not necessarily mean a false negative. Simulation results show that ARMS is effective in detecting and isolating misbehaving nodes and that its effectiveness improves with an increase in traffic load.

False negatives decrease at a fast rate as the traffic load increases. This can also be explained in terms of the increased interactions between nodes that come as a benefit to misbehavior detection. In our simulations false negatives reach 0% at 300 flows for both the *5-selfish-node* and *10-selfish-node* scenarios (Figure 29 and Figure 30). Thus, 100% of misbehaving nodes are exposed and are either suspended or under redemption. From the figures, the proportion of misbehaving nodes that are exposed and under suspension added to the proportion of misbehaving nodes that are under redemption increases proportionally to the traffic load. The proportion of exposed nodes that are under suspension increases monotonically with traffic load while the proportion of exposed nodes that are under redemption does not. Hence, as the traffic load increases misbehaving nodes spend more time under suspension than under redemption.

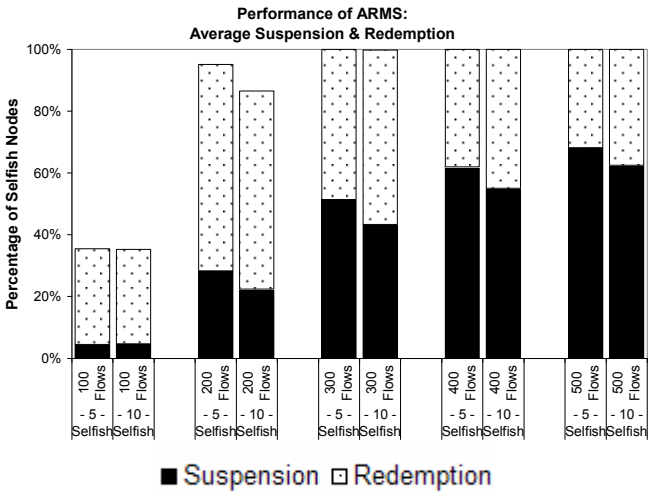

Figure 29 Effectiveness of ARMS: Average Exposure Ratio

Shows average exposure ratio and redemption ratio for 5 and 10 selfish nodes.

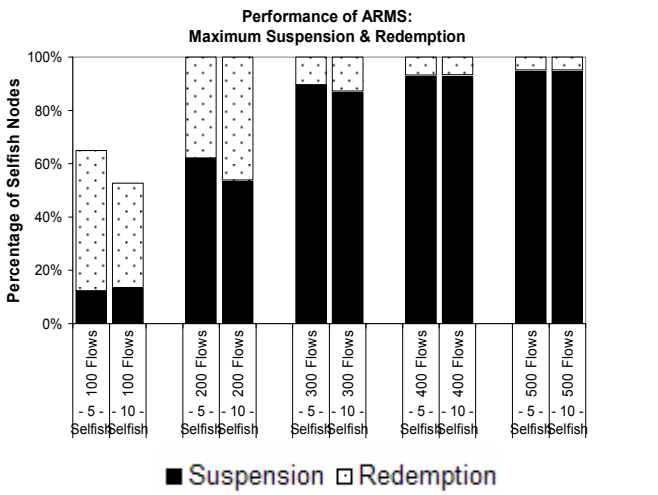

Figure 30 Effectiveness of ARMS: Maximum Exposure Ratio

Shows maximum exposure ratio and redemption ratio for 5 and 10 selfish nodes.

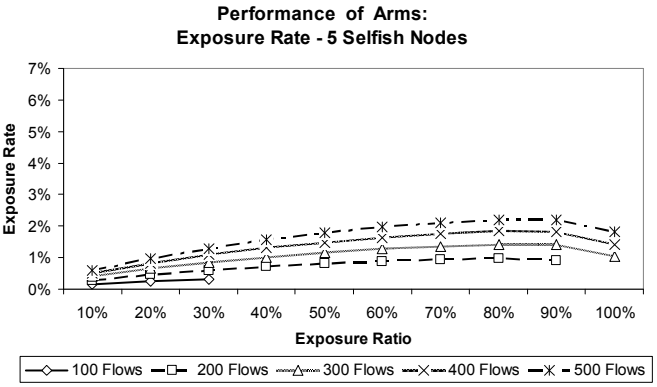

Figure 31 Effectiveness of ARMS: Shows exposure rate for 5 selfish nodes.

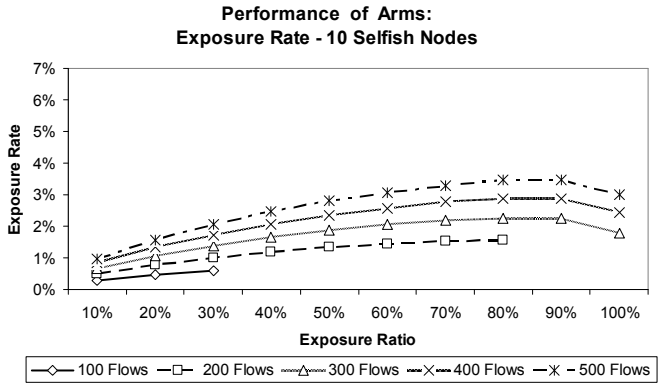

Figure 32 Effectiveness of ARMS: Shows exposure rate for 10 selfish nodes.

In order to determine the promptness with which the reputation management system detects misbehaving nodes, we measured the exposure rate of ARMS (Figure 31 and Figure 32). In both the 5-selfish-node and 10-selfish-node scenarios, the exposure rate of misbehaving nodes increases as the traffic load increases. The figure also shows that the exposure rate in the 10-selfish-node scenario is close to double that of the 5-

selfish-node scenario, which indicates that exposure rate is directly proportional to the misbehavior rate.

3.2.3. False Positives

Node misbehavior may have a negative impact on the perceived behavior of other nodes in the network (we will further study the correlation between false positives and proximity to misbehaving nodes in the next section). This may result in false positives, where a cooperative node may be perceived as misbehaving. False positives may also occur when traffic load increases beyond the network capacity, which results in nodes unintentionally dropping packets due to link and routing layers failures or channel impairments. We evaluate ARMS in this section based on false positives. Our findings indicate that ARMS maintains very low false positives under all misbehavior rates and traffic loads investigated.

We evaluate the performance of ARMS in terms of the average and maximum false positives produced in our two scenarios (Figure 33 and Figure 34). ARMS showed very low false positives, with a maximum of 1% at 500 flows in the *10-selfish-node* scenario. This is a benefit of the verification policy employed, which comes to the rescue of cooperative nodes that might be falsely tagged as misbehaving.

When a node falsely identifies one of its neighbors as misbehaving, it suspends that neighbor, dropping all traffic forwarded through it or originating at it. This results in packet drops due to false positives. This negative impact of ARMS on the network is maintained very low at low traffic loads, reaching about 0.3% of all packet drops at 100 flows, but increases with traffic load to about 5% of all packet drops at 500 flows for both

misbehavior rates scenarios. However, this percentage is insignificant compared to the reduction in packet drops due to misbehavior (Figure 25 and Figure 26).

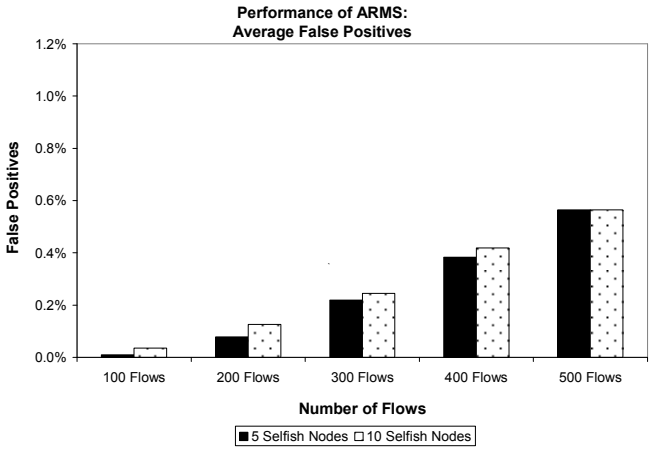

Figure 33 Performance of ARMS: Shows average false positives for 5 and 10 selfish nodes.

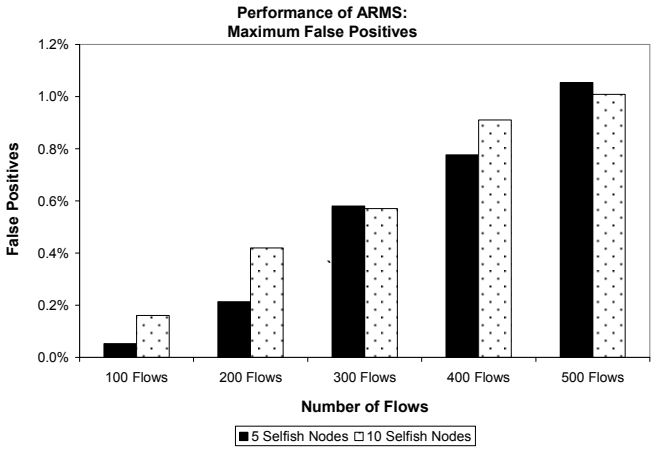

Figure 34 Performance of ARMS: Shows maximum false positives for 5 and 10 selfish nodes.

3.2.3.1 False Positives Proximity

False positives occur when a cooperative node is falsely accused as misbehaving by one of its neighbors. We define the set of falsely accused nodes (FAN) as $\text{FAN} = \{i : i \in \text{N/M} \wedge \exists\, j \in \text{N/M}$ such that $(d(i,j) = 1 \wedge r_{ji} < r_{thresh})\}$. Nodes in FAN are penalized and their participation level in the routing functionality is negatively impacted accordingly. A number of factors could negatively impact a node's perceived behavior to the point where it is falsely accused of misbehavior. Among these factors is proximity to three types of nodes: misbehaving nodes, other nodes in FAN, and neighbors of misbehaving nodes (NMNs) that did not expose their misbehavior (the set NMN is defined as $\text{NMN} = \{i : i \in \text{N/M} \wedge \exists j \in \text{M}$ such that $(d(i,j) = 1 \wedge r_{ij} \geq r_{thresh})\}$). Figure 35 shows how a cooperative node $i \in \text{N/M}$ may be falsely identified as misbehaving by its

neighbor node $j \in$ N/M (i.e. $i \in$ FAN) as a result of being a neighbor of a misbehaving node $m \in$ M, a node $f \in$ FAN, or a node $n \in$ NMN.

In this section we investigate the link between proximity to these three node types and a node being falsely accused as being misbehaving. We categorize nodes in FAN based on their distance in hops from the closest misbehaving node, other nodes in FAN, and nodes in NMN (Figure 36 and Figure 37). As expected, our findings indicate that most nodes in FAN are within one hop away from a misbehaving node, another node in FAN, or a node in NMN.

Our results show that a significant number of nodes in FAN are also in NMN. This is shown in Figure 36, where at 100 flows 50% of the nodes in FAN in the *5-selfish-node* scenario and 57% in the *10-selfish-node* scenario are within 0 hops away from a node in NMN node, which indicates that they are in NMN themselves. This percentage decreases for higher traffic loads. Since nodes in NMN are neighbors of misbehaving nodes and are also in FAN, their chances in participating in routing or getting their own packets through the network is at least cut in half (a node in our topology can have 4, 3, or 2 neighbors based on its location). This is because two of their neighbors, the misbehaving node due to its misbehavior and the false accuser as a disciplinary response, will refuse to participate with them in routing or packet forwarding. Our results also indicate that at 100 flows 80% of the nodes in FAN are within 1 hop of a node in NMN in the *5-selfish-node* scenario and 87% in the *10-selfish-node* scenario. This ratio decreases for higher traffic loads for both scenarios but is always greater than 50%.

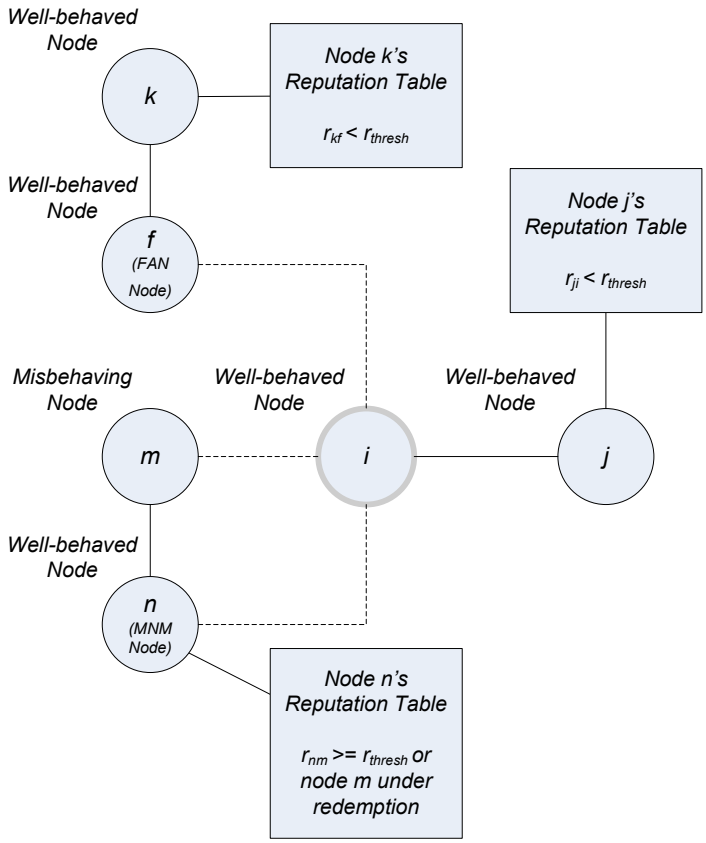

Figure 35 A Falsely Accused Node (node $i \in \mathrm{N} \setminus \mathrm{M}$).

A dashed line indicates a possible connection between two nodes and a solid line indicates an actual connection. Node $i \in \mathrm{N} \setminus \mathrm{M}$ could be falsely identified as misbehaving by node $j \in \mathrm{N} \setminus \mathrm{M}$ as a consequence of being a neighbor of a misbehaving node $m \in \mathrm{M}$, a node $f \in \mathrm{FAN}$, or a node $n \in \mathrm{NMN}$.

Our results also indicate that at 100 flows 66% of nodes in FAN are one hop away of a misbehaving node in the *5-selfish-node* scenario and about 51% for 200-500 flows (Figure 36 and Figure 37). The *10-selfish-node* scenario shows an almost constant value of about 74% of nodes in FAN are within one hop of a misbehaving node for all flow sets. The percentage of nodes in FAN that are within 2 hops from a misbehaving node is at least 86% in the *5-selfish-node* scenario and about 96% in the *10-selfish-node* scenario for all flows sets.

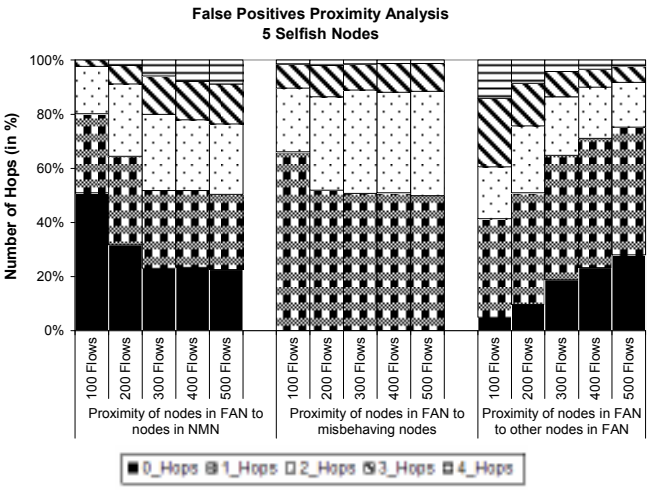

Figure 36 False Positives Proximity Analysis: results shown are for 5 selfish nodes when ARMS is active

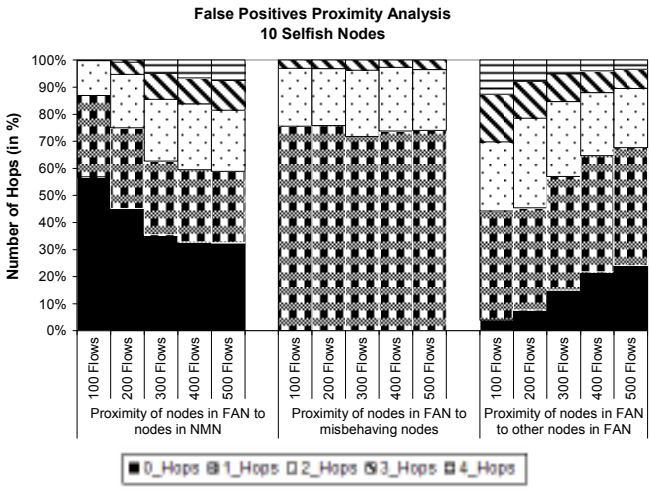

Figure 37 False Positives Proximity Analysis: results shown are for 10 selfish nodes when ARMS is active

Finally, our results indicate that 36% of all nodes in FAN are within one hop of another node in FAN in the *5-selfish-node* scenario at 100 flows (Figure 36 and Figure 37). This value increases to 41% at 200 flows and levels at about 47% for 300-500 flows. The percentage of nodes in FAN that are within one hop of another node in FAN in the *10-selfish-node* scenario is very close for all flow sets with an average of 41%. Note that about 4% of the nodes in FAN at 100 flows are falsely identified as misbehaving by more than one neighbor in both scenarios. This is indicated by the fact that 4% of nodes in FAN are within 0 hops away of another node in FAN. This ratio increases with the increase in traffic load until it reaches 24% in the *5-selfish-node* scenario and 28% in *10-selfish-node* scenario at 500 flows.

3.3. Performance Analysis

3.3.1. Overhead

Overhead measures the computation, communication, and storage requirements of a system that are exerted on the network by the proposed scheme. With respect to

ARMS, we will focus on communication overhead as a result of probes used by the system's verification component, and storage overhead due to the storage of outgoing packet signatures by the auditor component.

With respect to probing, the overhead on the network was calculated in terms of the ratio of probe packets sent to total number of probes packets and data packets sent. As shown in Figure 38, the highest probing overhead was in the *10-selfish-node* scenario at 500 flows and was less than 0.25% of total packets. In all cases, the probing overhead of the *10-selfish-node* scenario was higher than that of the *5-selfish-node* scenario. This of course is due to the increased misbehavior rate in the *10-selfish-node* scenario. Note that the relative difference between the two scenarios decreases as the traffic load increases. Looking at Figure 33, we noticed a similar trend comparing the false positives of the *5-selfish-node* scenario to that of the *10-selfish-node* scenario. The relative difference in percentage of false positives between the *5-selfish-node* scenario and the *10-selfish-node* scenario decreases as the traffic load increases. At high traffic loads, the percentage of false positives is similar for both scenarios. Since false positives generate probes, the increase in false positives results in an increase in probing. Hence, the increase in percentage of false positives in the *5-selfish-node*, which occurs at a faster rate compared to the *10-selfish-node* scenario, accounts for the increase in probing overhead, which also occurs at a faster rate compared to the *10-selfish-node* scenario.

In terms of storage overhead, nodes have to store packet traces in order to distinguish between delivered and retransmitted packets. The size of the lookup table could be minimized by using hashing algorithms. For 1000 table entries, the size of the table is about 20Kbytes. By using a hash value of 2bytes ($1/2^{16}$ collision probability), the

table size could be further reduced to 2Kbytes. Moreover, we conducted a number of simulation studies in order to observe the impact of limiting the size of the lookup table on the system performance in terms of exposure. We tested against two heuristics. The first limits the size of the lookup table to 1000 entries and uses a FIFO approach for new entries in excess of the table size. The second assigns an expiration time per table entry that is based on estimated flow round-trip time (RTT) at each node. Upon timing out, the entry is eliminated from the lookup table. For the first heuristic, there is a tradeoff between the lookup table size and the performance of ARMS in terms of exposure. If the upper limit on the lookup table size is set to a small value, the residency of entries in the table will be short, which will limit the ability of ARMS to evaluate node behavior. The second heuristic on the other hand has a better performance since the residency of an entry varied according the estimated RTT of the flow. This enhanced the performance of ARMS compared to the first heuristic. Our results showed that employing the second heuristic has negligible impact on the system's effectiveness.

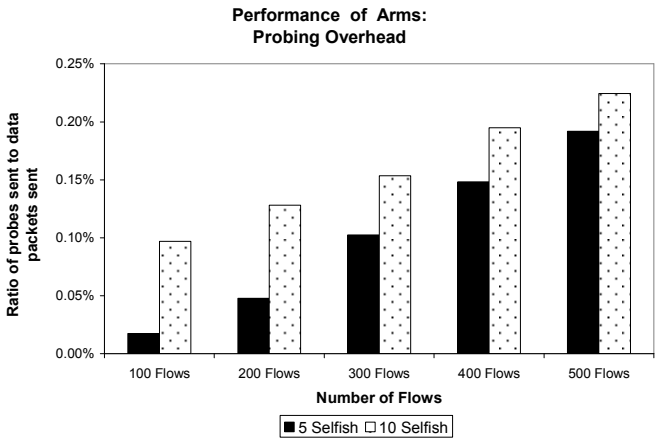

Figure 38 Overhead of the Verification Policy. Overhead is measured as the ratio of probes sent to total packets (probes + data) sent.

4. Analysis of Second-hand vs. First-hand Observations

It is expected that the system's effectiveness in terms of exposure rate and exposure ratio will be enhanced when a second-hand evaluation metric is employed [60]. Second-hand metrics allow nodes to use hearsay (the sharing of first-hand observations among nodes in the network) as mentioned in chapter II. This will result in faster exposure of misbehaving nodes [60]. However, if sharing is done prematurely, shared observations might include false accusations which could increase false positives.

For comparison purposes, we extended ARMS to use second-hand evaluation metric. We allow nodes to share negative observations about one another. There are different policies of sharing that determine what observations to share, when to share, and with whom to share. In our implementation of sharing in ARMS we evaluate two independent policies. The first policy shares observations in the form of detection events while the second shares negative events. The detection-event sharing policy shares nodes' suspension decisions with their neighbors while the negative-event sharing policy shares events that negatively influence a node's reputation score, mainly packet drop events. The motivation behind the detection-event sharing policy is to increase the confidence level in shared observations (i.e. minimize propagation of false positives) and to reduce sharing overhead at the expense of observation freshness. Hence, a node i decides to share a negative observation about node j if and only if $d(i,j) = 1 \wedge r_{ij} < r_{thresh}$ and the outcome of verification was affirmative. Thus, sharing is not immediate, but accumulated until a node is classified and affirmed as misbehaving. It is also not very frequent and false accusations are filtered out by behavior verification in order to avoid premature sharing of negative observations, which could increase false positives. We

adopt an approach whereby when a node *i* receives a first-hand observation about a node *k*, it updates its score as discussed earlier in section 2.2.2. If node *i* receives from node *j* a negative observation about node *k*, it will update the score of node *k* as a function of its own perceived score of nodes *j* and *k* as $r_{ik} = r_{ik} - [(r_{ik} - r_{thresh}) \times (r_{ij} / r_{\max})]$. This function was selected somewhat arbitrarily, but with two important features:

- The weight given to hearsay is directly proportional to the reputation of the node that is providing the observation.
- The significance of shared observations outweighs that of first-hand observations since the observations shared are an accumulation of first-hand observations made at the sharing node, which expedites misbehavior exposure.

The negative-event sharing policy, on the other hand, aims to share observations of misbehavior events as they occur to ensure nodes are promptly aware of them. For every packet dropped by node *k* for its neighbor node *j*, node *j* informs its neighbors of this negative observation. Hence, the shared observations are fresh but they are unfiltered, which decreases their accuracy. Each node *i* that receives the shared observation adjusts the reputation score of node *j* to $r_{ik} = r_{ik} - \frac{r_{-}}{2}$. This policy was selected somewhat arbitrarily, but with the important feature that the receipt of a shared negative observation has less negative impact on a neighbor's reputation than the direct observation of misbehavior. Since sharing events triggered by this policy are atomic and less accurate compared to detection-event sharing policy, the penalty applied by the behavior evaluation function is set to be lower than the one applied based on first-hand negative observations.

In both policies, sharing is localized, reaching only nodes that are 2 hops away from the sharing node in order to get to the neighbors of the node which the shared observation refers to. The shared observation of node j sent by node i is received at a node $k \in \mathbf{N}$ if and only if $d(i,k) \leq 2$. Since nodes evaluate the behavior of their immediate neighbors only, only nodes $k \in \mathbf{N}$ where $d(j,k) = 1$ make use of the shared observation. A node that receives negative observation shared by one of its neighbors about itself will disregard it since it is in its own best interest not to forward it any further.

In the rest of this discussion, we distinguish between three system configurations:

- FH-only: The reputation management system using a first-hand evaluation metric.
- SH-detection-event-sharing: The reputation management system using a second-hand evaluation metric based on sharing of all detection-events.
- SH-negative-event-sharing: The reputation management system using a second-hand evaluation metric based on sharing of negative-events.

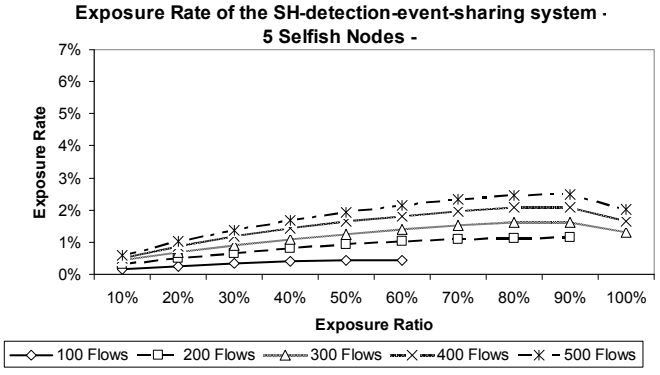

Figure 39 Exposure Rate of the SH-detection-event-sharing system (compare to Figure 31: exposure rate of FH-only)

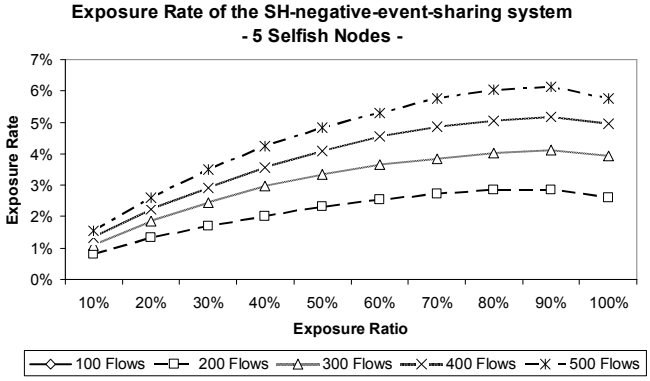

Figure 40 Exposure Rate of the SH-negative-event-sharing system (compare to Figure 31: exposure rate of FH-only)

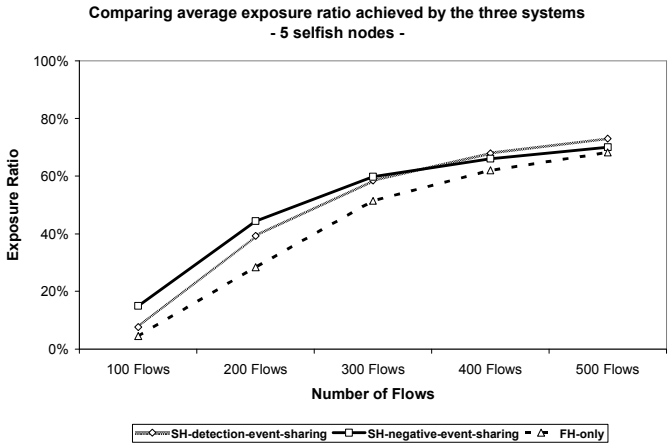

Figure 41 Average exposure Ratio of the SH-detection-event-sharing and SH-negative-event-sharing systems compared to the FH-only system.

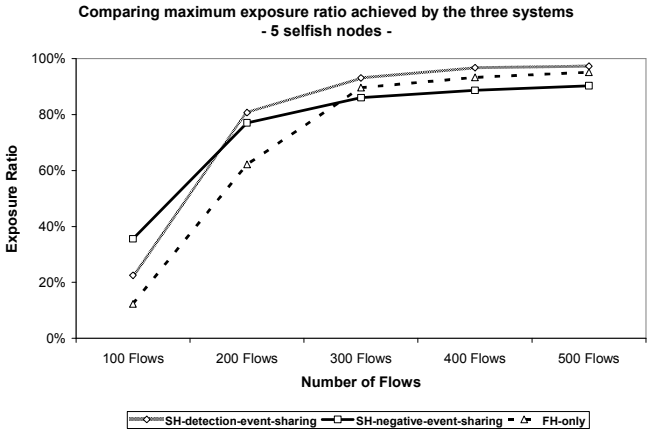

Figure 42 Maximum exposure Ratio of the SH-detection-event-sharing and SH-negative-event-sharing systems compared to the FH-only system.

4.1. Evaluation in the Presence of Selfish Nodes

Our evaluation criteria of both the SH-detection-event-sharing and the SH-negative-event-sharing systems will be based on the *5-selfish node* scenario only and will focus on exposure rate, exposure ratio, false positives, and overhead (we observed similar results in the *10-selfish node* scenario, here omitted to avoid redundancy). Our results indicate that employing a second-hand metric enhances exposure rate (Figure 39 and Figure 40). The exposure rate of the SH-detection-event-sharing system slightly outperforms the FH-only system. The SH-negative-event-sharing system outperforms both. This is expected due to the freshness of events shared by the SH-negative-event-sharing. Moreover, the average exposure ratio for the SH-negative-event-sharing system starts out better than the other two systems but is outperformed by both as the traffic load increases (Figure 41). From Figure 42, the maximum exposure ratio reached at the SH-negative-event-sharing system outperforms that of the SH-detection-event-sharing system at 100 flows only. Furthermore, the FH-only system outperforms SH-negative-event-

sharing system starting at 300 flows. The difference, however, between the three policies in terms of exposure ratio is slim. Furthermore, both the SH-negative-event-sharing and the SH-detection-event-sharing systems reduce the impact of misbehavior in terms of packet drops as shown in Figure 43. The ratio of packet drops due to misbehavior is lowest in the SH-negative-event-sharing system, followed by the SH-detection-event-sharing system. Figure 44 also shows the reduction in the number of packet drops due to misbehavior by the SH-negative-event-sharing and the SH-detection-event-sharing systems. The reduction is insignificant at low traffic loads but increases to reach 55.9% at 500 flows for the SH-negative-event-sharing system and about 24.6% at the same traffic load for the SH-detection-event-sharing system.

Figure 45 and Figure 46 show false positives for the three systems. False positives resulting from the SH-negative-event-sharing system are significantly higher than the other two systems. Avoiding premature sharing of negative events, as mentioned before, is a policy employed by the SH-detection-event-sharing system. The SH-negative-event-sharing system, on the other hand, shares all negative events (i.e. all packet drops) that occur irrespective to their cause (i.e. misbehavior, congestion, etc.), which results in the propagation of false accusations. This results in an increase in false positives, which increases with the traffic load. Moreover, the sharing overhead of the SH-negative-event-sharing system is very high compared to the SH-detection-event-sharing system (Figure 47). We also note that the overhead of sharing is a function of the average node degree in the network topology, which is somewhat low in the topology used in our simulations (average is 3.5). The overhead will increase as the average node

degree increases, which raises scalability concerns for the SH-negative-event-sharing system and the SH-detection-event-sharing systems.

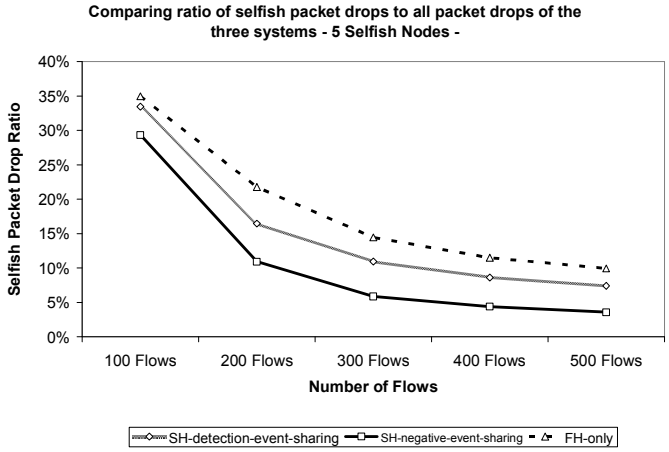

Figure 43 Ratio of selfish packet drops to all packet drops of the SH-detection-event-sharing and SH-negative-event-sharing systems compared to FH-only system.

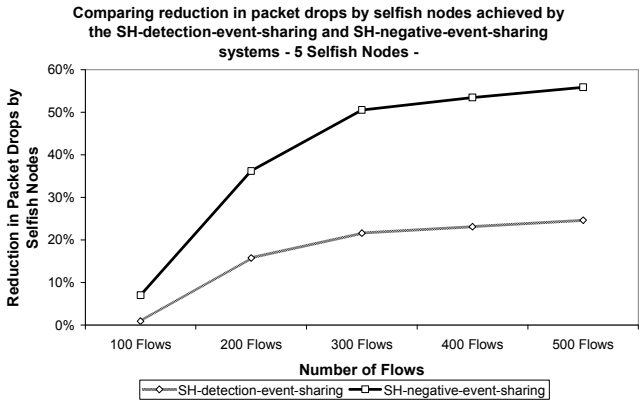

Figure 44 Reduction in packet drops by selfish nodes achieved by the SH-detection-event-sharing and SH-negative-event-sharing systems.

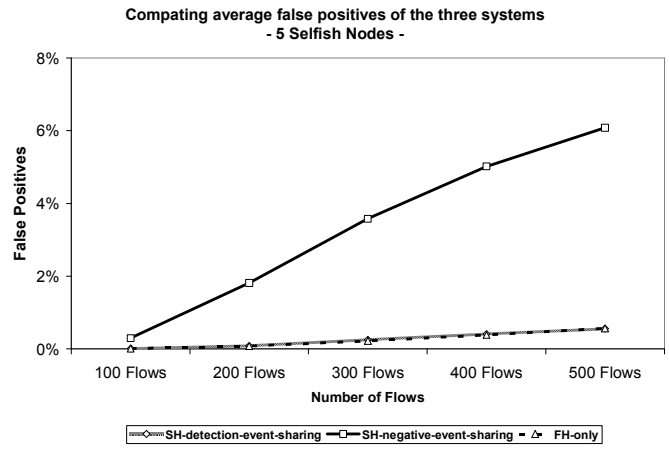

Figure 45 Average false positives of the SH-detection-event-sharing and SH-negative-event-sharing systems compared to FH-only system.

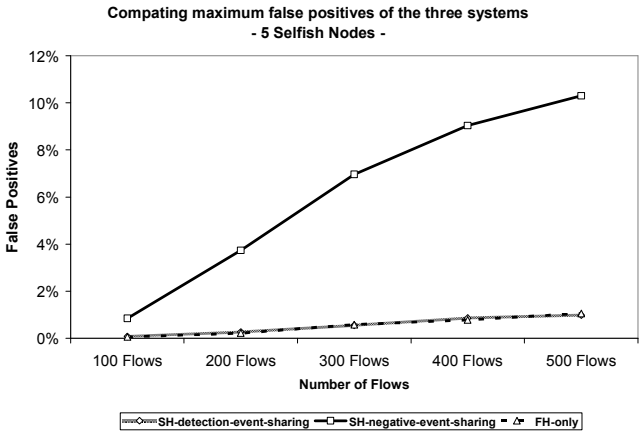

Figure 46 Maximum false positives reached by the SH-detection-event-sharing and SH-negative-event-sharing systems compared to FH-only system.

In order to analyze the joint benefit and the cost of employing second-hand metrics, we defined a benefit-cost ratio function (BCR) . Considering a system S that employs a second-hand metric and another $S^{\sim}$ that employs a first-hand metric. The

relative benefit to the network is defined in terms of the reduction in packet drop due to misbehavior when a second-hand metric is employed $B(S,S^{\sim}) = \sum_{i \in M}^{S^{\sim}} P_{drop}^{i} - \sum_{i \in M}^{S} P_{drop}^{i}$, where P_{drop}^{i} is the number of packets dropped by a misbehaving node *i*. On the other hand, the relative cost is defined in terms of the increase in packet drops due to false positives (which is an indirect negative impact of misbehavior) when a second-hand metric is employed $C(S,S^{\sim}) = \sum_{i \in N \setminus M}^{S} P_{FP}^{i} - \sum_{i \in N \setminus M}^{S^{\sim}} P_{FP}^{i}$, where P_{FP}^{i} is the number of packets dropped due to false positives by a cooperative node *i*. Hence, $BCR = \frac{B(S,S^{\sim})}{C(S,S^{\sim})}$, *where* $C(S,S^{\sim}) \neq 0$. BCR measures the gained benefit compared to the cost exerted on the network. Employing a second-hand metric is beneficial if $BCR > 1$.

Figure 48 compares the BCR of the SH-negative-event-sharing system over FH-only system to the BCR of the SH-detection-event-sharing system over the FH-only system. From the figure, the BCR of the SH-negative-event-sharing system is always less than 0.5 for all traffic loads. The number of packets dropped due to false positives resulting from the negative-event sharing policy is higher than the reduction in packets dropped by misbehaving nodes. The increase in false positives is due to the low accuracy of the shared observations since a large number of packet drops occur that are unrelated to misbehavior. Also, the overhead of the SH-negative-event-sharing system (Figure 47) is a disadvantage of the policy, reaching 23% of total traffic. Hence, the benefit does not justify the cost for the SH-negative-event-sharing system and an FH-only system is a better option.

On the other hand, BCR for the SH-detection-event-sharing system is high for most flow sets. BCR ranges between 3.65 and 5.80 between 100 and 400 flows. It peaks at 500 flows reaching 77.17, where the increase in P_{FP}^{i} is almost negligible compared to the reduction in packet drops. Figure 47 also shows that the overhead from using this system is low. Hence, it is clear that the benefit from the SH-detection-event-sharing system significantly supersedes the cost in the presence of selfishness, which makes it a more attractive solution compared to an FH-only system.

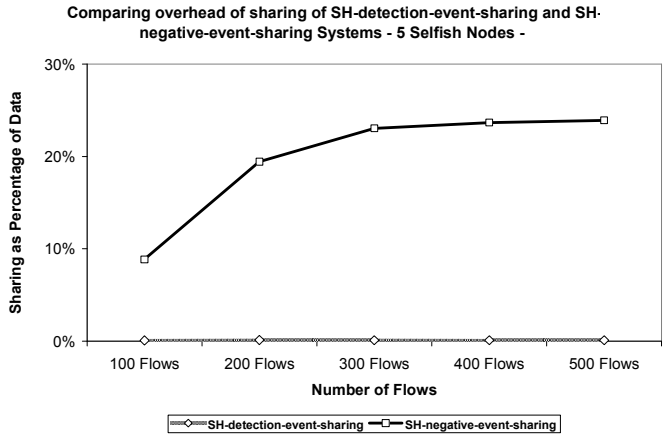

Figure 47 Overhead of sharing by SH-detection-event-sharing and SH-negative-event-sharing systems.

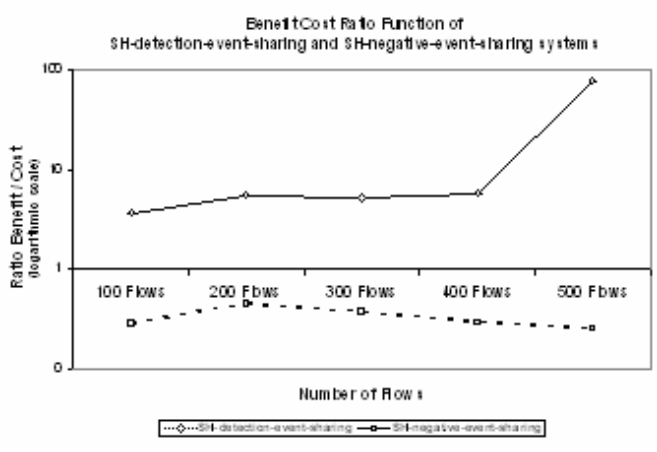

Figure 48 Benefit Cost Ratio Function of SH-detection-event-sharing and SH-negative-event-sharing systems.

4.2. Evaluation in the Presence of Malicious Nodes

As sharing introduces more complexity to a reputation management system it also exposes the network to more complex misbehavior types. These misbehavior types attempt to reduce the precision of the behavior evaluation functionality and hence increase the impact of attacks on the routing functionality. Thus, malicious behavior such as slander attacks must be considered when second-hand metrics are employed. Accordingly, we investigate the usage of second-hand metrics in the presence of a

blackhole malicious attack [62] where the attackers are capable of also launching slander attacks by sharing false observations about their neighbors. An attacker maintains the rate of the slander attack at a level similar to that of sharing observations so as not to raise suspicions. In our evaluation, we used the SH-detection-event-sharing system only since the SH-negative-event-sharing system has already shown low BCR.

In order to measure the impact of malicious misbehavior and slander attacks on the network, we define cumulative misbehavior impact (CMI) as $CMI = \frac{\sum_{i \in M} P^i_{drop} + \sum_{i \in N \setminus M} P^i_{FP}}{\sum_{i \in N} P^i_{data}}$, *where* $\sum_{i \in N} P^i_{data} \neq 0$. CMI measures the proportion of data packets sent that gets dropped resulting, directly or indirectly, from malicious slander misbehavior. Since slander attacks are expected to increase the number of false positives, the measurement of the impact of malicious behavior needs to incorporate packets dropped as a result of false positives in the evaluation of the SH-detection-event-sharing system. The SH-detection-event-sharing system is considered good if $CMI \approx 0$.

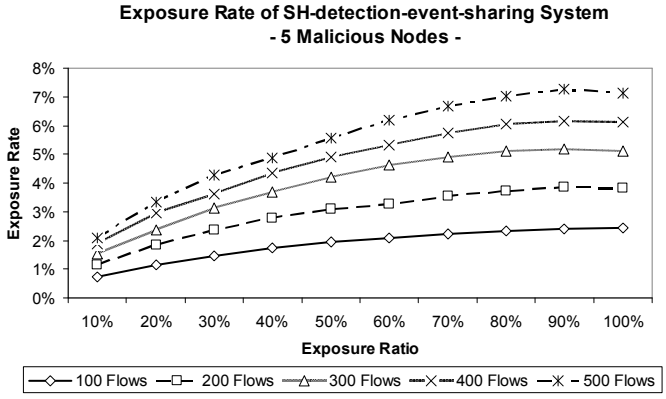

Figure 49 Exposure rate achieved by the SH-detection-event-sharing system.

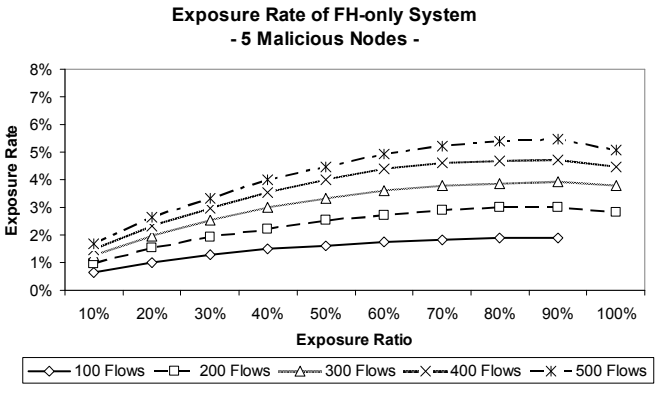

Figure 50 Exposure rate achieved by the FH-only system.

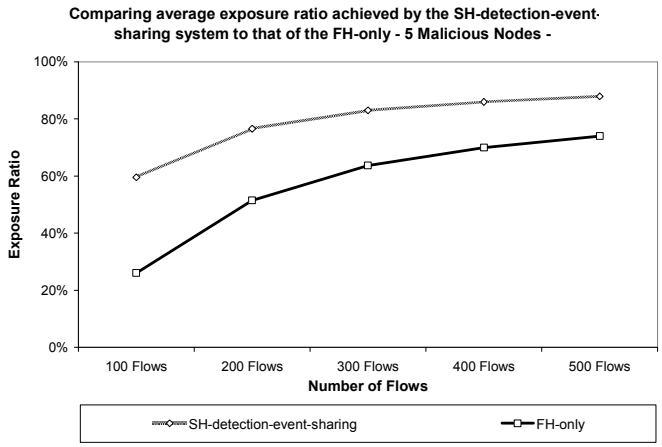

Figure 51 Average exposure achieved by the SH-detection-event-sharing system is higher than the FH-only system.

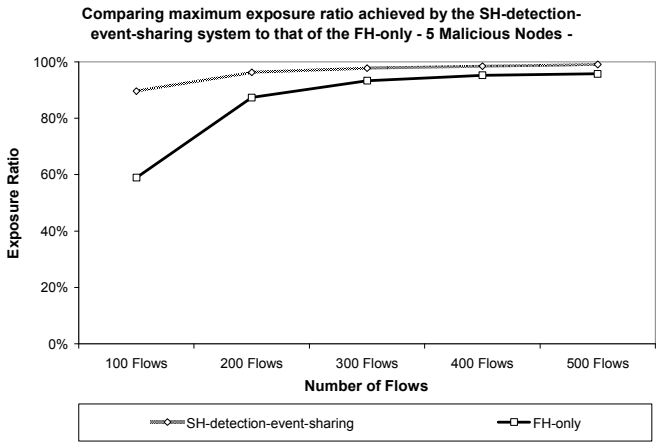

Figure 52 Maximum exposure achieved by the SH-detection-event-sharing system is higher than the FH-only system.

As expected, the exposure rate and exposure ratio of SH-detection-event-sharing system outperforms the FH-only system in the presence of 5 malicious-slanderers (Figure 49, Figure 50, Figure 51, and Figure 52). The impact of misbehavior is also reduced as a result (Figure 53). However, looking at false positives (Figure 54 and Figure 55), the SH-detection-event-sharing system incurred much higher false positives compared to the FH-only system. The introduction of liars in the network resulted in an increase in false accusations which lead to the increase of false positives. To increase the impact of their attack on the network, malicious nodes exploited the fact that shared observations are used by the SH-detection-event-sharing system and spread lies about their neighbors. This led to the decrease of the victim's score and hence increased false positives and also reduced the value of observations shared by the victims. Moreover, Figure 56 shows the overhead of sharing by the SH-detection-event-sharing system which constituted about 20% of the total number of packets encapsulating shared observations plus data packets sent in the network.

Using our BCR and CMI functions, we analyzed the benefit and cost of the SH-detection-event-sharing system in the presence of malicious nodes (Figure 57). Our results show that the impact of false positives that result from the malicious slander attackers on the network reduced the value of BCR. BCR peaks at 100 flows where it reaches 2.5 but is reduced below 0.5 starting at 200 flows reaching 0.2 at 500 flows. Hence, the increase in packet drops due to false positives exceeds the reduction in packet drops by malicious nodes by a factor of 5 at 500 flows. The CMI function shows complementary results, where the SH-detection-event-sharing system shows a lower CMI value than the FH-only system at 100 flows. CMI of the SH-detection-event-sharing system increases almost linearly starting at 200 flows to exceed the FH-only system, which decreases to a constant value starting at 200 flows (Figure 57). Accordingly, the benefit and cost analysis of the SH-detection-event-sharing system indicates that the cost exerted surpasses the benefit gained, which suggests that reliance on a FH-only system only is a better option.

In conclusion, employing a second-hand metric expedites detection of misbehaving nodes and decreases their direct impact on the network compared to employing a first-hand metric. However, sharing observations allows malicious attackers to exploit the sharing function by launching slander attacks, which increases false positives. The cost of false positives added to the overhead of sharing results in a significant cost that supersedes the benefit of employing a second-hand metric.

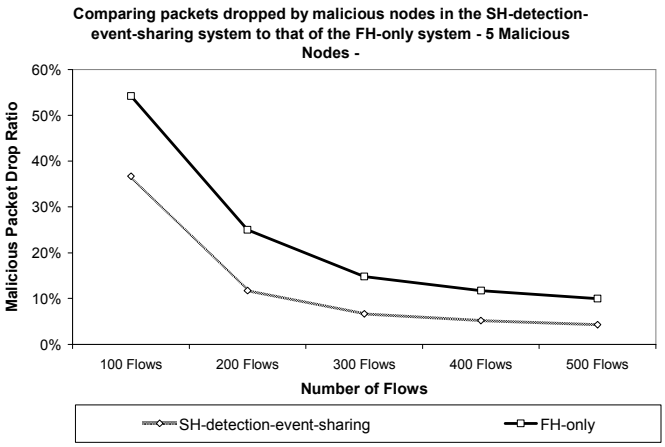

Figure 53 Packets dropped by malicious nodes. SH-detection-event-sharing vs. FH-only.

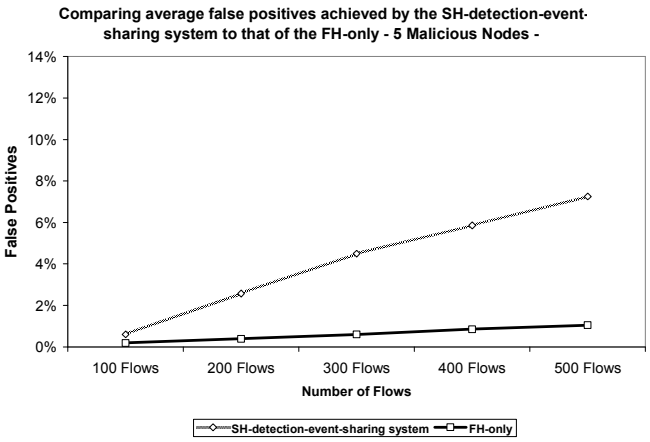

Figure 54 Average false positives for 5 malicious nodes. SH-detection-event-sharing vs. FH-only.

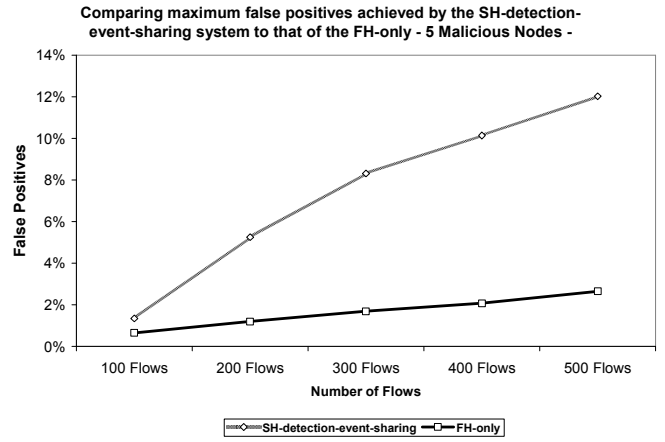

Figure 55 Max. false positives for 5 malicious nodes. SH-detection-event-sharing vs. FH-only.

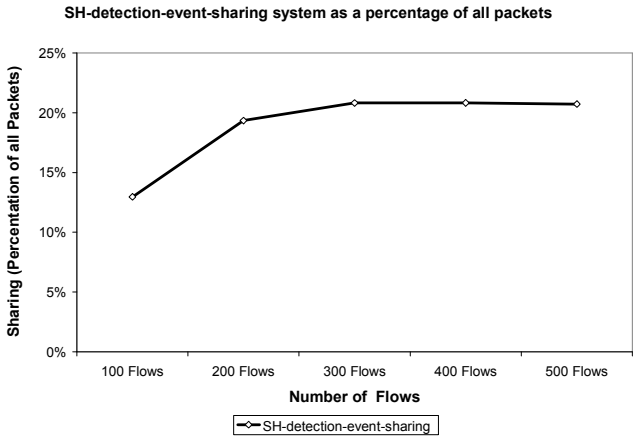

Figure 56 Overhead of the SH-detection-event-sharing system as a percentage of all packets

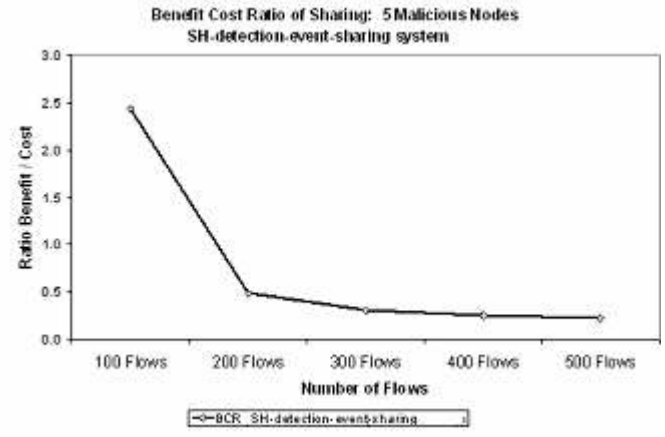

Figure 57 Benefit Cost Ratio of the SH-detection-event-sharing system.

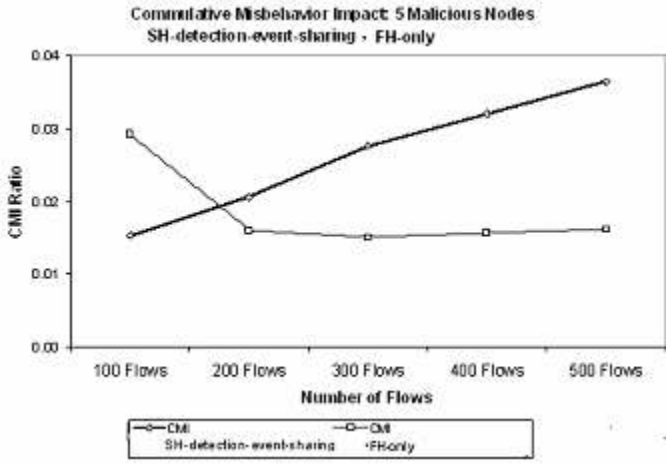

Figure 58 CMI of the SH-detection-event-sharing vs. FH-only.

5. ARMS Demonstration

The Mobile Ad hoc Networking Interoperability and Cooperation (MANIAC) Challenge is a competition that will be held in 2007 at Virginia Tech. The objective of the competition is to stimulate experimentation in wireless networking, while studying cooperation in ad hoc networks. The competition will entail a multimedia traffic relay race in which all the teams are on the mobile ad hoc network, but no team will be able to have their traffic delivered without cooperation from others. Teams will be required to bring hardware running an open source operating system. The organizers will provide software to monitor node behavior and system effectiveness, giving researchers a wealth of data on actual ad hoc network behavior in a field dominated by simulation and controlled testbed research. As a demonstration for a solution to encourage cooperation among nodes in the MANIAC challenge, a version of ARMS (auditor, behavior evaluator, behavior classifier, and a responder component) was implement using a Linux testbed. In this section, we will briefly discuss the system implemented and the evaluation results.

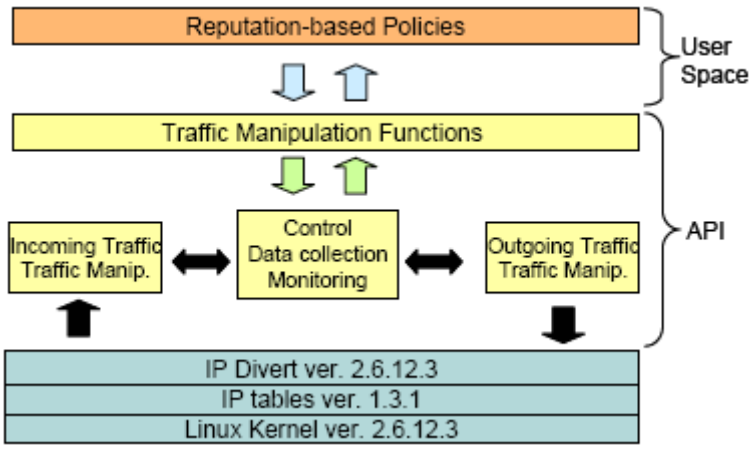

Figure 59 System Architecture

5.1. System Overview

The system architecture includes the auditor, behavior evaluator, behavior classifier, and the responder components. The system uses Red Hat Linux with IP Divert sockets integrated into the Linux kernel to intercept packets based on specific criteria and deliver them to higher layer applications. The received packets are redirected to the application program interface (API) that resides between the application and the network layer where incoming traffic and outgoing traffic are recorded for use by the auditor component (Figure 59).

5.2. System Evaluation

To demonstrate the operation of the reputation-based mechanism a video stream is generated from node MS/C1 and is sent to node MS/C2. Nodes MS/C1 and MS/C2 are not within range and will have to rely on nodes A, B, and C to forward packet sent from MS/C1 to MS/C2. Nodes A and B are cooperative while node C is selfish dropping some or all of packets routed through. The route is initially set as MS/C1-B-C- MS/C2 (Figure 60). Initially, node C acts cooperatively allowing for successful delivery of packets. This is demonstrated by the high quality of the video shown at node MS/C2 as shown in Figure 61. Once node C start to act selfishly by dropping packets forwarded through it, the quality of the video deteriorates as shown in Figure 62. Observing the lack of acknowledgments traveling in the reverse path, the reputation management system at node B identifies node C as selfish and reacts by triggering a new route to be established through node A (Figure 63). Hence, node C is bypassed ensuring successful delivery of the multimedia stream to the destination, which is demonstrated by the high quality video shown in Figure 64.

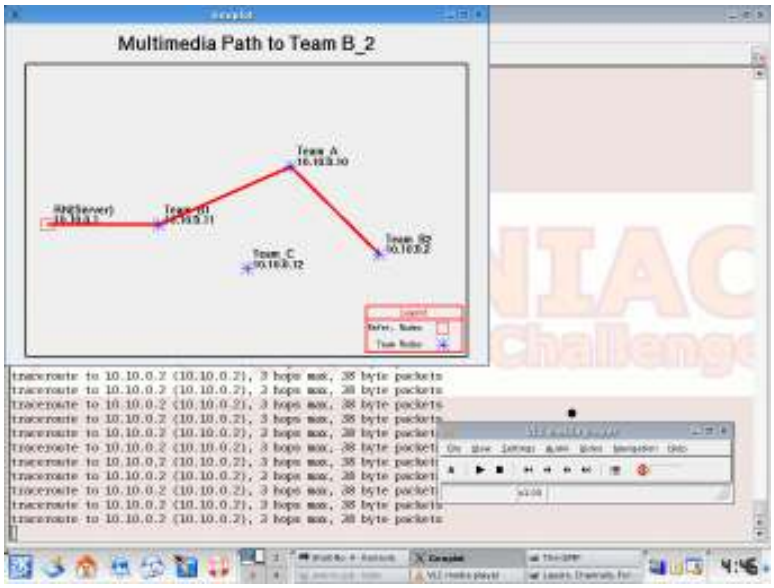

Figure 60 Initial route is set to MS/C1–B–C–MS/C2. Node C is acting cooperatively.

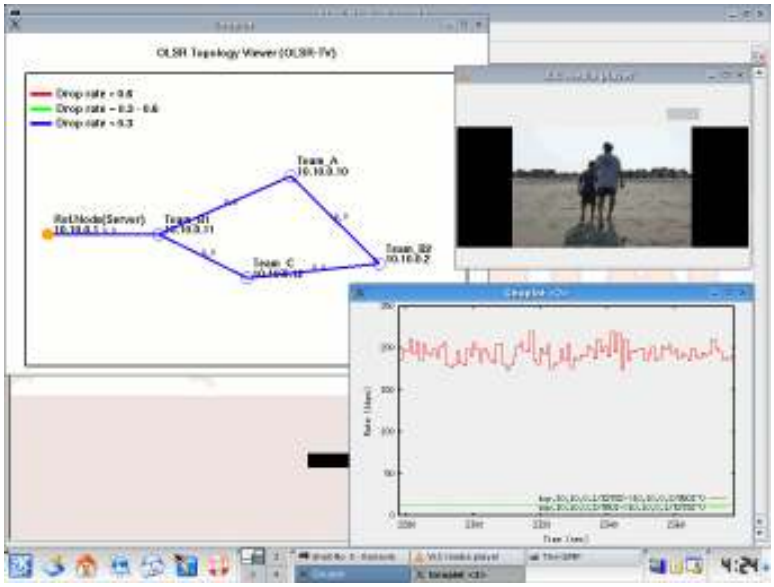

Figure 61 High quality video shown at node MS/C2 as long as node C is acting cooperatively.

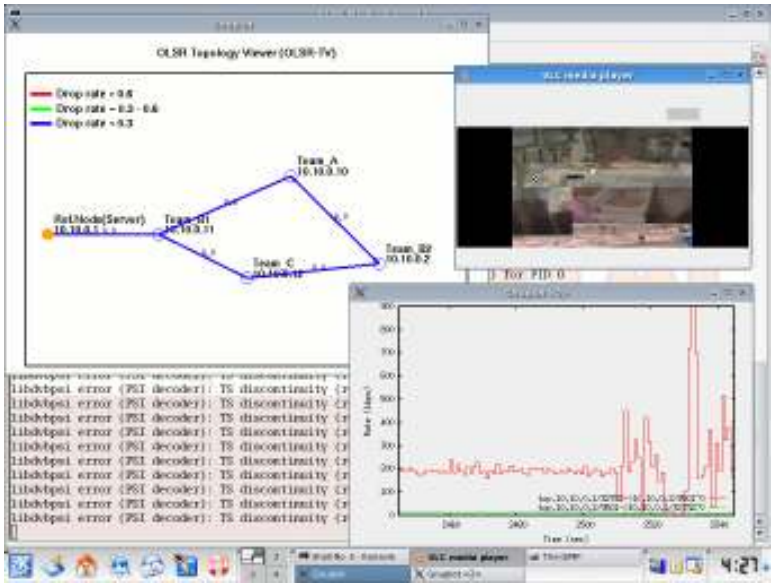

Figure 62 Video quality deteriorates when node C starts to act selfishly.

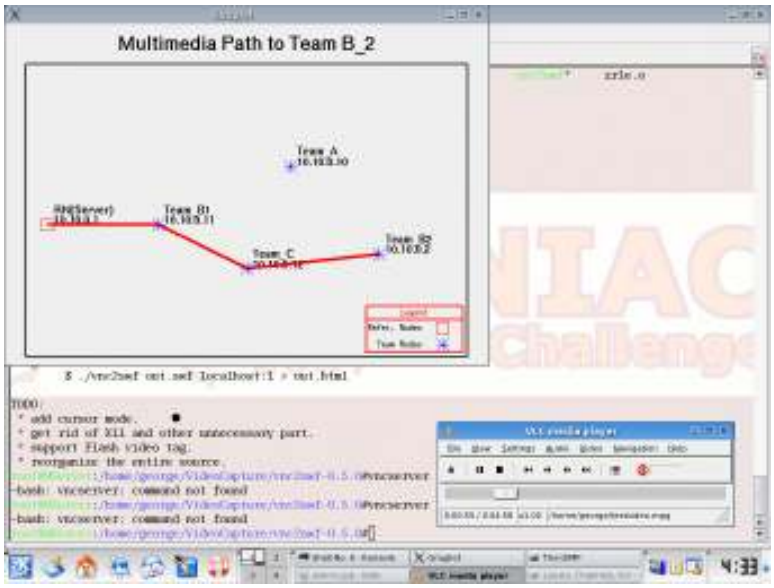

Figure 63 Node B triggers a route update and chooses to route packets to MS/C2 through node A rather than node C.

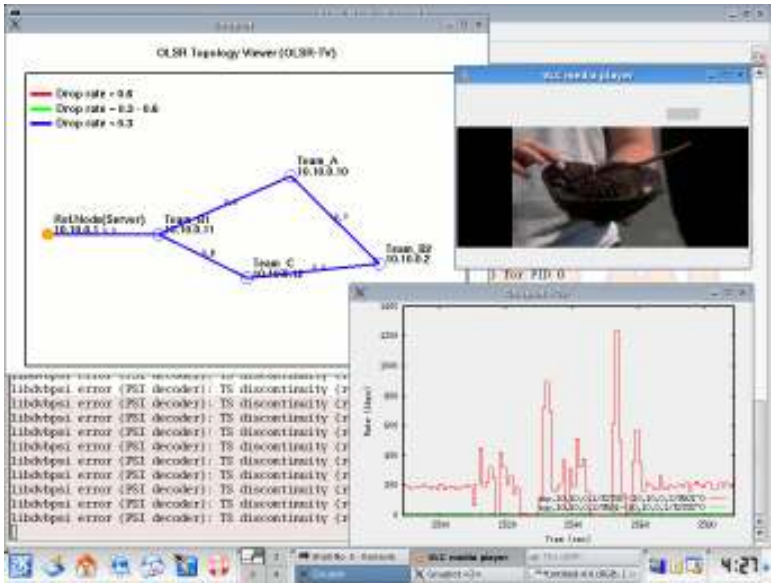

Figure 64 Video quality increases as the route changes to MS/C1–B–A–MS/C2

6. Summary

In this chapter we conducted an extensive simulation study to evaluate ARMS from performance and security perspectives. We used the quantitative evaluation metrics that we defined in Chapter III as the basis of our evaluation. We showed that ARMS limits the impact of misbehavior by detecting and isolating misbehaving nodes while incurring very low false positives and insignificant communication overhead. We also conducted a comprehensive analysis of the benefit and cost of second-hand and first-hand observations with respect to ARMS. Our results revealed that while second-hand observations could expedite the exposure of misbehavior in the network, it could also result in significant overhead. Based on a benefit-cost function that we developed, we concluded that the benefit of second-hand observations in terms of reducing the impact of misbehavior on the network may not justify the cost, which is based on the overhead exerted on the network and the increase in false positives. Finally, we discussed a demonstration of version ARMS on a Linux-based testbed, which showed that ARMS

can effectively detect and deter the negative impact of selfish behavior in ad hoc networks.

Chapter VI. Adaptive Reputation Management System for Ad hoc Networks

Typically, the functions of reputation management (evaluation, detection, and reaction) are carried out homogeneously across time and space at different nodes. The dynamic nature of ad hoc networks causes node behavior to vary spatially and temporally due to changes in the local and network-wide conditions. Reputation management functions do not adapt to such changes, which may impact the system accuracy and promptness. In this chapter, we propose an adaptive reputation management system where nodes carry out the reputation management functions heterogeneously across time and space according to the instantaneous perception by each node of its surrounding network conditions. We discuss some of the characteristics of an adaptive reputation management system such as sensitivity, adaptability, accuracy, and promptness. We propose a number of evaluation metrics from different layers of the protocol stack and discuss the impact of each metric on the accuracy and sensitivity of the evaluation function. We then propose a time-slotted approach for the evaluation function and show how the duration of an evaluation slot can adapt according to the network activity to enhance the system accuracy and promptness. We also show how the detection function can adapt to the network conditions by using the node's own behavior as a benchmark to set misbehavior detection parameters.

1. Adaptive Reputation Management Systems

The goals of a reputation management system are to: evaluate nodes' behavior based on packet forwarding (evaluation), distinguish between cooperative and

misbehaving nodes (detection), and appropriately react to misbehaving nodes (reaction). It is commonly assumed that reputation management systems at different nodes carry out these functions (evaluation, detection, and reaction) homogeneously across time and space. In other words, the evaluation criteria, detection decision factors, and reactive measures are the same at all nodes at all times. The homogeneous application of reputation decisions often results in sub-optimal performance, as it restricts adaptation [1]. Ad hoc networks operate in a dynamic environment where network conditions change from time to time and may be perceived differently from one node to another. At times the network conditions are such that a node j that chooses to act cooperatively can successfully do so by forwarding all traffic routed through it (due to moderate traffic activity, favorable channel conditions, etc.). At other times network conditions are such that node j may not be able to successfully forward traffic routed through it due to inadequate conditions (such as high congestion, channel impairments, etc.). Node behavior may vary from one node to another and from time to time due to changes in the local and network-wide conditions, affecting the ability of the reputation mechanism to distinguish between a node's willful decision not to forward packets (misbehavior) and its inability to do so due to adverse network conditions.

The accuracy and promptness of a reputation management system with respect to its misbehavior detection may deteriorate if it does not adapt to network conditions. If the perception of cooperative behavior remains the same under both adequate and inadequate network conditions, a node may be incorrectly identified as misbehaving during inadequate network conditions and inappropriate reaction may be taken against it (e.g. isolation from the network). Hence, the evaluation criteria, detection decision

factors, and reactive measures should be carried out differently at different times. We herein refer to such a reputation management system as *adaptive*.

To realize an adaptive reputation management system, its functions must be sensitive (capture changes in node behavior) and adaptable (adjust according to the network conditions) as well as accurate (able to distinguish between cooperative and misbehaving nodes unambiguously) and prompt (timely detect misbehaving nodes). We focus in this chapter on enhancing the sensitivity and adaptability of the evaluation and detection functions while maintaining the system's accuracy and promptness. Firstly, we discuss the impact of the choice of an evaluation metric on the accuracy and sensitivity of the evaluation function. We evaluate different metrics from different layers of the protocol stack and identify the ones that when used can yield accurate and prompt behavior evaluation. To enhance the sensitivity of the evaluation function, we adopt a time-slotted approach for node behavior evaluation. A node's behavior is evaluated in each slot independently from its behavior in previous slots. We illustrate how the slot duration impacts the accuracy and promptness of the system. A slot duration that is too short reduces the system's accuracy, while one that is too long reduces its promptness. We propose an approach whereby the slot duration adapts to the network activity. We then look into the detection function, proposing to enhance its adaptability by having a node use its own behavior as benchmark for making decisions about others' behavior. Cooperative nodes within close proximity usually experience similar network conditions. Thus, a cooperative node's own behavior can be an adaptable benchmark for judging the behavior of nodes within its local neighborhood. We also investigate the potential of making decisions about nodes' behavior using multiple evaluation metrics, possible at

different protocol layers, to enhance the system accuracy as opposed to making a decision using a single evaluation metric.

This chapter has three main contributions. Firstly, we identify evaluation metrics that can yield accurate and sensitive node behavior evaluation. Secondly, we introduce a slotted evaluation function and show that by adapting the slot duration according to the network activity we can maintain good system accuracy and promptness. Thirdly, we show that by using a node's own behavior as benchmark for making decisions about others' behavior the detection function can adapt to changes in network conditions. In the rest of this chapter, we adopt the notation in Table 5.

Table 5 Notation Used

N → The set of nodes in an ad hoc network.
$\mathrm{M} \subset \mathrm{N}$ →The subset of N that are misbehaving nodes.
C → The set of evaluation metrics.
$d(i, j)$ → The distance in hops between two nodes $i, j \in \mathrm{N}$.
$abs(i)$ → The absolute value of i.
$\mathrm{G_i} = \{j : j \in \mathrm{N} \setminus \mathrm{M} \wedge d(i, j) = 1\}$ → The set of non-misbehaving neighbors of node $i \in \mathrm{N}$.
p → The evaluation slot duration (in seconds).
$f(p)$ → The number of times node behavior evaluation is triggered using evaluation slot duration p.
$A_{j,i,c} = \{a^1_{j,i,c}, ..., a^{f(p)}_{j,i,c}\}$ → Set of scores assigned to node $i \in \mathrm{N}$ by neighbor $j \in \mathrm{G_i}$

using metric $c \in \mathrm{C}$ and slot duration p.
$\overline{A}_{j,i,c} = \frac{1}{f(p)} \sum_{k=1}^{f(p)} a^{k}_{j,i,c}$ → Average score assigned to node $i \in \mathrm{N}$ by neighbor $j \in \mathrm{G_i}$ using metric $c \in \mathrm{C}$ and slot duration p.
$\overline{A}_{i,c} = \frac{1}{\lvert \mathrm{G_i} \rvert} \sum_{j \in \mathrm{G_i}} \overline{A}_{j,i,c}$ → The average score assigned to node $i \in \mathrm{N}$ by the subset of its neighbors $j \in \mathrm{G_i}, j \notin M$.
$\overline{A}_{\mathrm{N \setminus M},c} = \frac{1}{\lvert \mathrm{N} \rvert - \lvert \mathrm{M} \rvert} \sum_{i \in N \setminus M} \overline{A}_{i,c}$ → Average of all scores assigned to nodes $i \in \mathrm{N \setminus M}$ by their non-misbehaving neighbors.
$\overline{A}_{\mathrm{M},c} = \frac{1}{\lvert \mathrm{M} \rvert} \sum_{i \in M} \overline{A}_{i,c}$ → The average of all scores assigned to misbehaving nodes $i \in \mathrm{M}$ by their non-misbehaving neighbors.
$\overline{A}_{\mathrm{diff},c} = abs(\overline{A}_{\mathrm{N \setminus M},c} - \overline{A}_{\mathrm{M},c})$ → Difference between the average score of misbehaving nodes and cooperative nodes.

2. Sensitivity and Adaptability of the Evaluation Function

2.1. Selection of Evaluation Metric

In this section we investigate metrics that can be used to evaluate node behavior. The sensitivity and accuracy of the system depends on the evaluation metric employed. An accurate evaluation metric is one that makes it possible to distinguish between cooperative and misbehaving nodes based on the score assigned to each. The accuracy of an evaluation metric may be impacted by the accuracy of the data it uses to assign node

scores. Passive acknowledgements, for example, which are known to yield an accurate evaluation metric, are data that can be used to determine the number of packets forwarded through a node and the number of packets routed through it to assign a score to a node. The choice of an evaluation metric as well as the accuracy of the data used by this evaluation metric are key to the accuracy of the reputation management system.

In this section, we propose evaluation metrics from the network and link layers and evaluate the accuracy and sensitivity of each. For the moment, we assume 100% data accuracy to focus on the extent of the accuracy and sensitivity of each candidate evaluation metric. For evaluation metrics that we identify as accurate and sensitive, we evaluate the impact of data inaccuracy in section 3.2.3.

To asses the accuracy of a candidate evaluation metric, we use $\overline{A}_{\text{diff},c}$. Evaluation metrics that are good indicators of node behavior will show a clear difference between $\overline{A}_{\text{N}\setminus\text{M},c}$ and $\overline{A}_{\text{M},c}$. To assess the sensitivity of a candidate evaluation metric, we consider *Node A* $\in$ N that is being evaluated by its neighbors. *Node A* alternates its behavior between cooperative and misbehaving. For the purpose of this discussion, we let *Node A* alternate its behavior three times. Let $A^1_{j,i,c} = \{a^1_{j,i,c},...,a^k_{j,i,c}\}$, $A^2_{j,i,c} = \{a^{k+1}_{j,i,c},...,a^l_{j,i,c}\}$, $A^3_{j,i,c} = \{a^{l+1}_{j,i,c},...,a^m_{j,i,c}\}$, and $A^4_{j,i,c} = \{a^{m+1}_{j,i,c},...,a^n_{j,i,c}\}$ be the sequence of scores assigned to a node $i \in \{Node\ A\}$ by one of its neighbors $j \in \text{G}_\text{i}$ based on candidate evaluation metric $c \in \text{C}$ between each two alterations of node i's behavior. We consider a candidate evaluation metric $c \in \text{C}$ sensitive if a clear difference is shown between each two behavior alterations $\overline{A}^h_{i,c}, \overline{A}^{h+1}_{i,c}, 1 \le h < 4$.

We use ns-2 simulations in our evaluation. We consider an ad hoc network of 64 nodes arranged in an 8X8 grid. The communication range is set such that each node has 4 neighbors, with the exception of edge nodes, which have 3 neighbors, and corner nodes, which have 2 neighbors. The network adopts IEEE 802.11 for medium access and AODV [55] for routing. The network has five misbehaving nodes: a misbehaving node will drop a data packet forwarded through it with probability $P_{selfish}$. We simulate the network under different traffic loads. We randomly generate sets of 20, 40, 60, 100, and 200 TCP flows. For each set of TCP flows, 10 simulations are executed, each for 600 seconds, with flow source and destination pairs selected at random for each run. Traffic is generated for each flow using a CBR (Constant Bit Rate) traffic generator that generates approximately 300 data packets per flow. All our results shown are averaged for the 10 simulation runs.

2.1.1. Network Layer Evaluation Metrics

In this section, we introduce three different evaluation metrics from the network layer and evaluate each as an indicator of node behavior. The evaluation metrics considered are packet forwarding ratio, packet drop ratio, and the arrival rate of packet forwarding requests.

2.1.1.1 Packet Forwarding Ratio

This evaluation metric monitors the packet forwarding ratio of all nodes $i \in \mathrm{N}$. The motivation to investigate the potential of this metric stems from the fact that cooperative nodes forward more packets than misbehaving nodes do.

Each neighbor $j \in \mathrm{G}_i$ of node $i \in \mathrm{N}$ keeps count of $PKT_{j,i}^{routed}$, the number of data packets routed through node $i \in \mathrm{N}$, and $PKT_{j,i}^{forward}$, the number of data packets that neighbor j believes node i forwarded. We define packet forwarding ratio of node $i \in \mathrm{N}$ as perceived by its neighbor $j \in \mathrm{G}_i$ as $PFR_{j,i} = PKT_{j,i}^{forward} / PKT_{j,i}^{routed}$, if $PKT_{j,i}^{routed} \neq 0$. At each behavior evaluation slot k where a node $j \in \mathrm{G}_i$ is evaluating the behavior of node $i \in \mathrm{N}$, $PFR_{j,i}$ is calculated, $PKT_{j,i}^{routed}$ and $PKT_{j,i}^{forward}$ are reset, and $a_{j,i,c}^{k} = PFR_{j,i}$ (evaluation metric c corresponds to packet forwarding ratio). Figure 65, Figure 66, and Figure 67 show $\overline{A}_{i,c}, \forall i \in \mathrm{N}$ under three different traffic loads with misbehaving nodes at $P_{selfish} = 75\%$. From the figures, it is clear that $\overline{A}_{i,c}, \forall i \in \mathrm{N} \setminus \mathrm{M}$ is much higher than $\overline{A}_{i,c}, \forall i \in \mathrm{M}$. In Figure 68, $\overline{A}_{\mathrm{diff},\, c}$ is shown for $P_{selfish} = 75\%$ under five different traffic loads, illustrating that the least difference in score between the average score of cooperative nodes and that of misbehaving nodes is 48 (at 200 flows). Figure 69 compares $\overline{A}_{\mathrm{diff},\, c}$ averaged over the five traffic loads for three different values of $P_{selfish}$ ($P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$), illustrating significant difference between the average score of cooperative nodes and that of misbehaving nodes for all three values of $P_{selfish}$. Accordingly, this evaluation metric is accurate as it is possible to unambiguously distinguish between cooperative and misbehaving nodes.

Figure 70 evaluates the sensitivity of this evaluation metric to change in node behavior based on the values of $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$, where $i = Node\ A$. In the Figure, $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$ for the two slots when *Node A* is misbehaving are

compared to the two slots when it is acting cooperatively. From the figure, the behavior of *Node A* at times when it acts cooperatively can be distinguished from times when it misbehaves based on the scores it is assigned using packet forwarding ratio (the same observation was made using $P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$). This holds under all traffic loads, indicating that this evaluation metric shows appropriate sensitivity to changes in behavior.

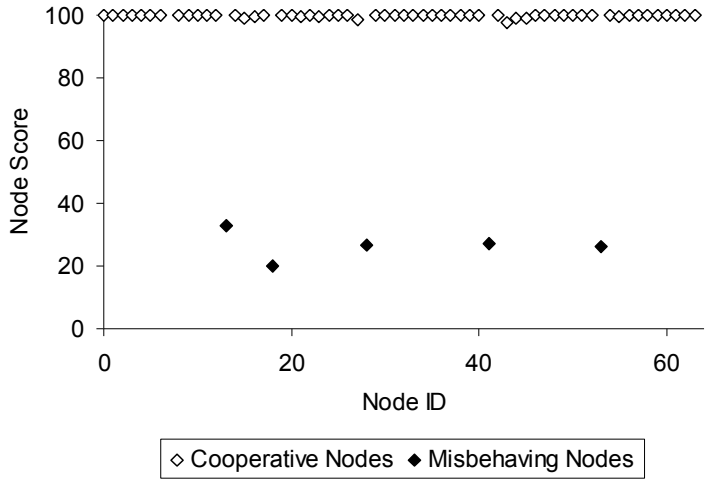

Figure 65 $\overline{A}_{i,c}$ Packet Forwarding Ratio – 20 Flows – Five Misbehaving Nodes –

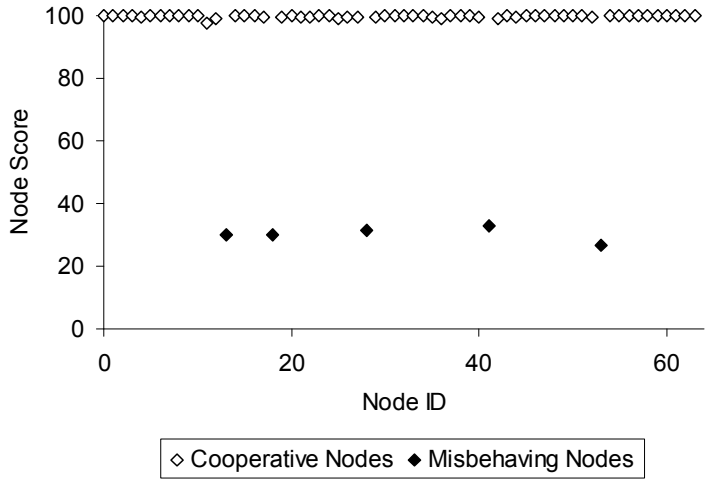

Figure 66 $\overline{A}_{i,c}$ Packet Forwarding Ratio – 60 Flows – Five Misbehaving Nodes –

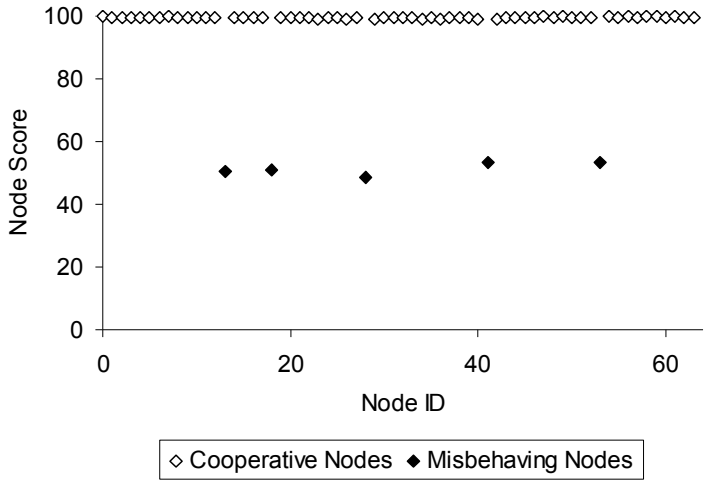

Figure 67 $\overline{A}_{i,c}$ Packet Forwarding Ratio – 200 Flows – Five Misbehaving Nodes –

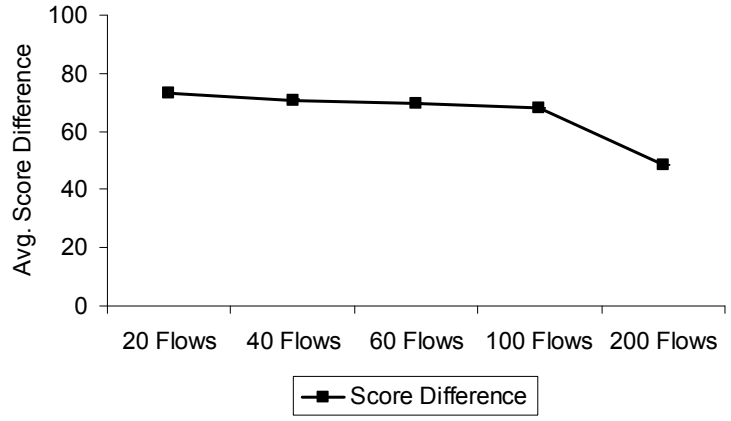

Figure 68 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Packet Forwarding Ratio under different traffic loads: ($P_{selfish} = 75\%$)

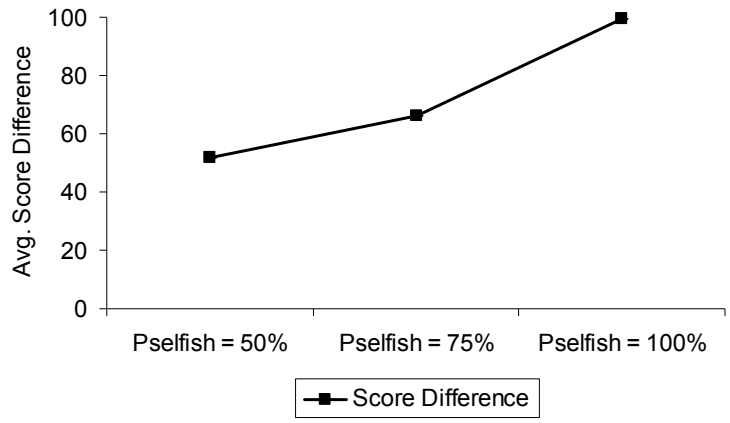

Figure 69 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Packet Forwarding Ratio under different values of $P_{selfish}$.

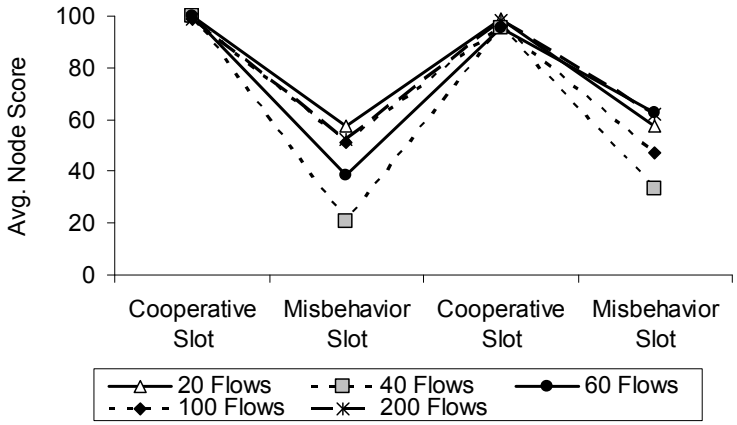

Figure 70 Sensitivity of Packet Forwarding Ratio to changes in node behavior ($P_{selfish} = 75\%$).

2.1.1.2 Packet Drop Ratio

This evaluation metric monitors the packet drop ratio of all nodes $i \in \mathrm{N}$. The motivation to investigate the potential of this metric stems from the fact that misbehaving nodes drop more packets than cooperative nodes do.

Each neighbor $j \in \mathrm{G}_i$ of $i \in \mathrm{N}$ keeps count of $PKT_{j,i}^{routed}$, the number of data packets routed through node $i \in \mathrm{N}$, and $PKT_{j,i}^{dropped}$, the number of data packets that neighbor j believes node i dropped. The count for packet drops represents packet drops at the network layer only and includes packet drops due to misbehavior (since misbehaving nodes act selfishly, dropping packets at the network layer). We define packet drop ratio of node $i \in \mathrm{N}$ as perceived by its neighbor $j \in \mathrm{G_i}$ as $PDR_{j,i} = PKT_{j,i}^{dropped} / PKT_{j,i}^{routed}$, if $PKT_{j,i}^{routed} \neq 0$. At each behavior evaluation slot k where a node $j \in \mathrm{G_i}$ is evaluating the behavior of node $i \in \mathrm{N}$, $PDR_{j,i}$ is calculated, $PKT_{j,i}^{routed}$ and $PKT_{j,i}^{dropped}$ are reset, and $a_{j,i,c}^{k} = PDR_{j,i}$ is assigned (evaluation metric c corresponds to packet drop ratio). Figure 71, Figure 72, and Figure 73 show $\overline{A}_{i,c}, \forall i \in \mathrm{N}$ under three different levels of traffic load with misbehaving nodes at $P_{selfish} = 75\%$. From the figures, it is clear that $\overline{A}_{i,c}, \forall i \in \mathrm{N} \setminus \mathrm{M}$ is much lower than $\overline{A}_{i,c}, \forall i \in \mathrm{M}$. In Figure 74, $\overline{A}_{\mathrm{diff},\,c}$ is shown for $P_{selfish} = 75\%$ under five different levels of traffic load illustrating that the least difference in score between the average score of cooperative nodes and that of misbehaving nodes is 26 (at 200 flows). Figure 75 compares $\overline{A}_{\mathrm{diff},\,c}$ averaged over the five values of traffic load for three different values of $P_{selfish}$ ($P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$), illustrating significant difference between the average score of cooperative nodes and that of misbehaving nodes for all three values of $P_{selfish}$. In summary, this evaluation metric is accurate, as it is possible to unambiguously distinguish between cooperative and misbehaving nodes.

Figure 76 evaluates the sensitivity of this evaluation metric to changes in node behavior based on the values of $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$, where $i = Node\ A$. In the Figure, $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$ for the two slots when *Node A* is misbehaving are compared to the two slots when it is acting cooperatively. From the figure, the behavior of *Node A* at times when it acts cooperatively can be distinguished from times when it misbehaves based on the scores it is assigned using packet drop ratio (we observe the same pattern for $P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$). This holds under all traffic loads indicating that this evaluation metric is appropriately sensitive.

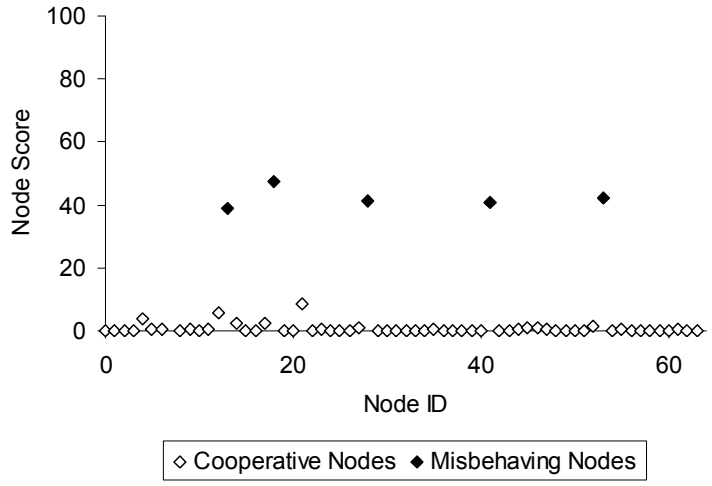

Figure 71 $\overline{A}_{i,c}$ Packet Drop Ratio – 20 Flows – Five Misbehaving Nodes –

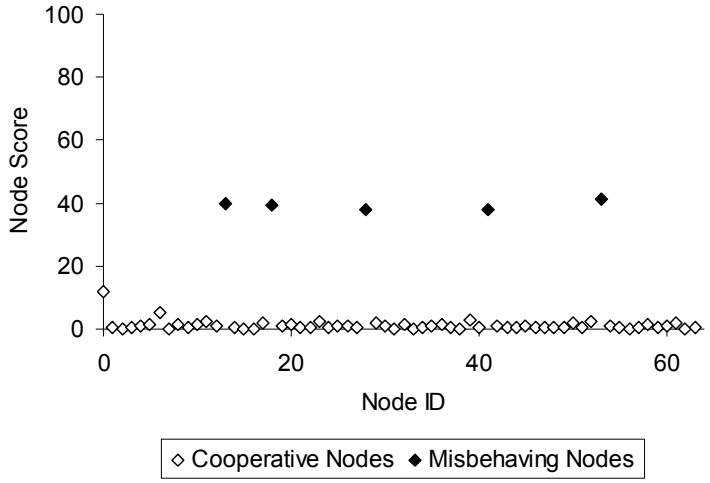

Figure 72 $\overline{A}_{i,c}$ Packet Drop Ratio – 60 Flows – Five Misbehaving Nodes –

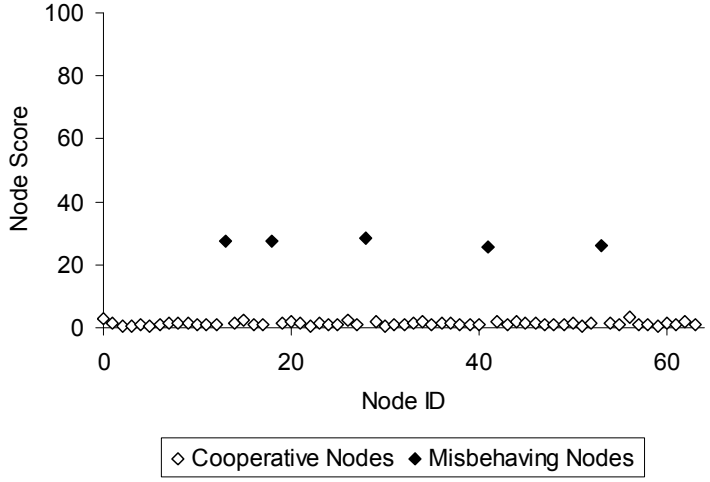

Figure 73 $\overline{A}_{i,c}$ Packet Drop Ratio – 200 Flows – Five Misbehaving Nodes –

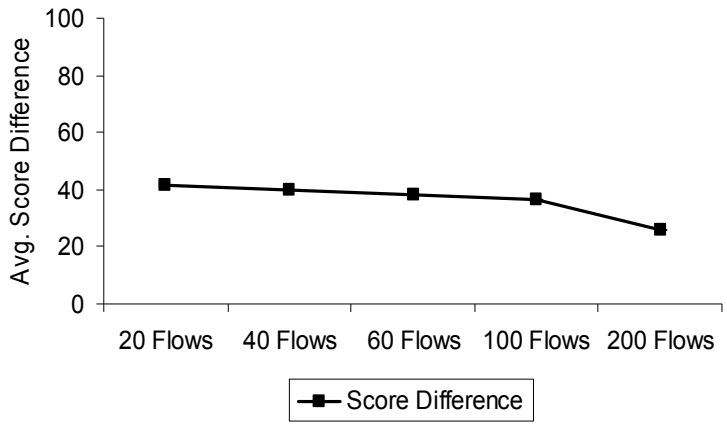

Figure 74 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Packet Drop Ratio under different traffic loads: ($P_{selfish} = 75\%$)

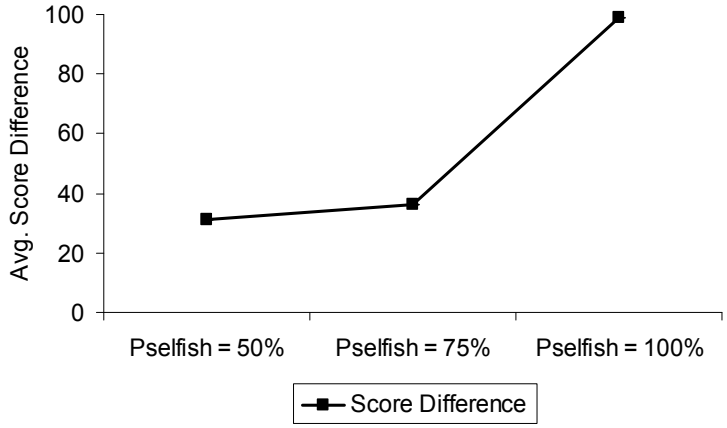

Figure 75 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Packet Drop Ratio under different values of $P_{selfish}$.

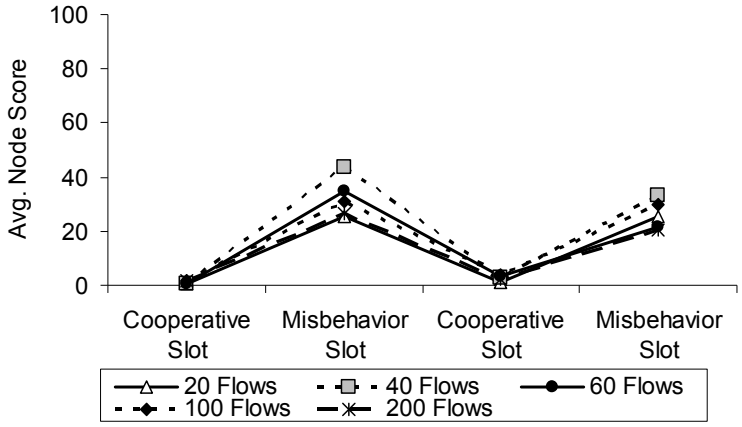

Figure 76 Sensitivity of Packet Drop Ratio to changes in node behavior ($P_{selfish} = 75\%$).

2.1.1.3 Packet Forwarding Requests Arrival Rate

This evaluation metric targets congestion aware transport layer protocols. The motivation to investigate the potential of this metric stems from the fact that misbehaving nodes drop more packets than cooperative nodes. Hence, for congestion aware transport layer protocols such as TCP, the arrival of packet forwarding requests at misbehaving nodes should be lower than at cooperative nodes due to the backoff mechanism.

Each neighbor $j \in \mathrm{G}_i$ of node $i \in \mathrm{N}$ keeps count of $PKT_{j,i}^{routed}$, the number of data packets routed through node $i \in \mathrm{N}$. At each behavior evaluation slot k where a neighbor $j \in \mathrm{G}_i$ of node $i \in \mathrm{N}$ is evaluating its behavior, the packet forwarding requests arrival rate (PFRAR) at node $i \in \mathrm{N}$ is calculated as $PFRAR_{ji} = \dfrac{PKT_{j,i}^{routed}}{\Delta T}$, where ΔT is the time elapsed since the last evaluation slot where node $j \in \mathrm{G}_i$ evaluated node $i \in \mathrm{N}$ (behavior evaluation slot duration). $PKT_{j,i}^{routed}$ is then reset and $a_{j,i,c}^{k} = PFRAR_{j,i}$ is assigned (evaluation metric c corresponds to packet forwarding requests arrival rate).

Figure 77, Figure 78, and Figure 79 show $\overline{A}_{i,c}, \forall i \in \mathrm{N}$ under three different traffic loads with misbehaving nodes at $P_{selfish} = 75\%$. From the figures, it is clear that $\overline{A}_{i,c}, \forall i \in \mathrm{N} \setminus \mathrm{M}$ is statistically indistinguishable from $\overline{A}_{i,c}, \forall i \in \mathrm{M}$. In Figure 80, $\overline{A}_{\mathrm{diff},\, c}$ is shown for $P_{selfish} = 75\%$ under five different traffic loads illustrating that the difference between the average score of cooperative nodes and that of misbehaving nodes is very small. Figure 81 compares $\overline{A}_{\mathrm{diff},\, c}$ averaged over the five traffic loads for three different values of $P_{selfish}$ ($P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$), illustrating small difference as well between the average score of cooperative nodes and that of misbehaving nodes for all three different values of $P_{selfish}$. Accordingly, this evaluation metric is inaccurate, as it is not possible to unambiguously distinguish between cooperative and misbehaving nodes.

Figure 82 evaluates the sensitivity of this evaluation metric to change in node behavior based on the values of $\overline{A}^1_{i,c}$, $\overline{A}^2_{i,c}$, $\overline{A}^3_{i,c}$, and $\overline{A}^4_{i,c}$, where $i \in \{Node\ A\}$. In the Figure, $\overline{A}^1_{i,c}$, $\overline{A}^2_{i,c}$, $\overline{A}^3_{i,c}$, and $\overline{A}^4_{i,c}$ for the two slots when *Node A* is misbehaving are compared to the two slots when it is acting cooperatively. From the figure, the behavior of *Node A* during the slots when it acts cooperatively is barely distinguishable from the slots when it misbehaves for some traffic loads and is indistinguishable for others. The sensitivity of this evaluation metric is shown to be good for low traffic loads but deteriorates as the traffic load increases (the same observation was made using $P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$). Hence, the sensitivity of this evaluation metric is a strong function of traffic load.

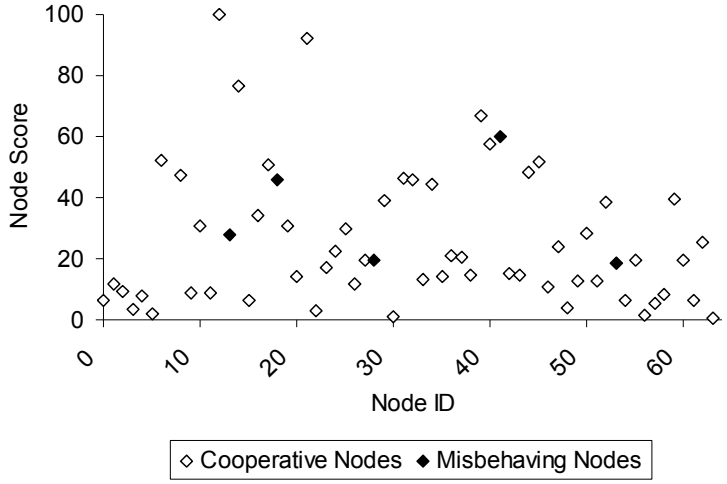

Figure 77 $\overline{A}_{i,c}$ Packet Forwarding Requests Arrival Rate – 20 Flows – Five Misbehaving Nodes –

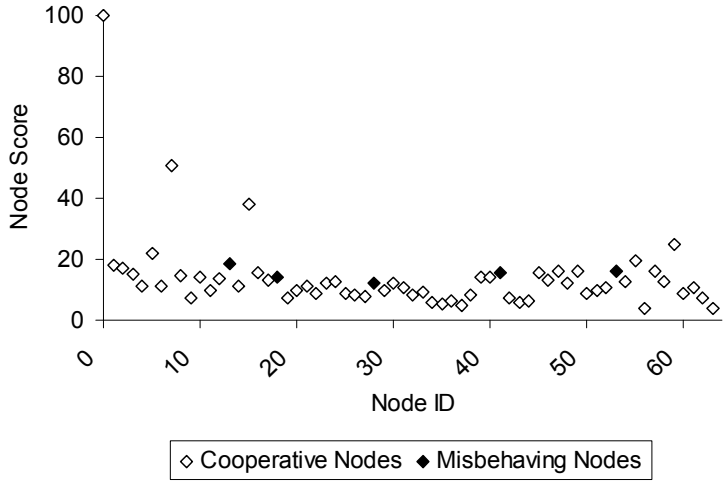

Figure 78 $\overline{A}_{i,c}$ Packet Forwarding Requests Arrival Rate – 60 Flows – Five Misbehaving Nodes –

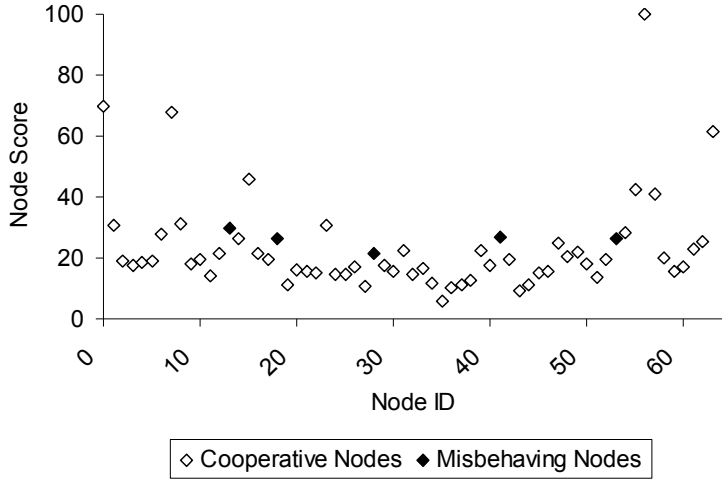

Figure 79 $\overline{A}_{i,c}$ Packet Forwarding Requests Arrival Rate – 200 Flows – Five Misbehaving Nodes –

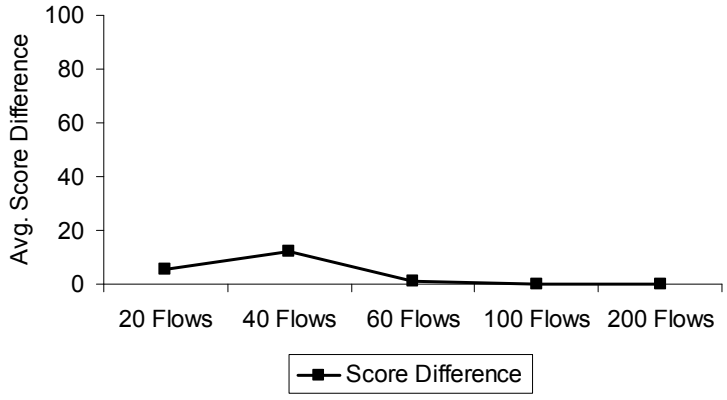

Figure 80 Accuracy ($\overline{A}_{\text{diff, } c}$) of Packet Forwarding Requests Arrival Rate under different traffic loads: ($P_{selfish} = 75\%$).

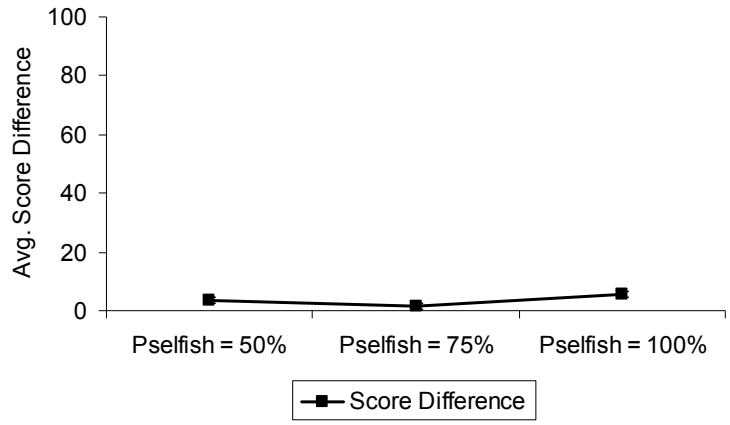

Figure 81 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Packet Forwarding Requests Arrival Rate under different values of $P_{selfish}$.

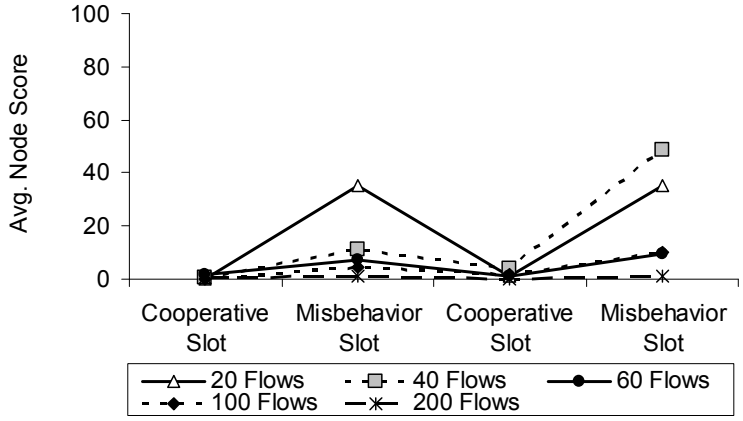

Figure 82 Sensitivity of Packet Forwarding Requests Arrival Rate to changes in node behavior ($P_{selfish} = 75\%$).

2.1.2. Link Layer Evaluation Metrics

In this section, we introduce two evaluation metrics from the data link layer and evaluate each as an indicator of node behavior. The evaluation metrics considered are medium access ratio and frame collisions.

2.1.2.1 Medium Access Ratio

This evaluation metric observes the frequency of transmitting data frames over the transmission medium compared to the frequency of receiving data frames for each node. The motivation to investigate the potential of this metric stems from the fact that misbehaving nodes drop more packets, and hence, transmit frames less frequently than they receive frames over the transmission medium.

IEEE 802.11 virtual carrier sensing makes use of Request-to-Send (RTS) and Clear-to-Send (CTS) frames preceding the transmission of a MAC Protocol Data Unit (MPDU) [63]. Each neighbor $j \in \mathrm{G}_i$ of node $i \in \mathrm{N}$ keeps count of $RTS_{j,i}$ and $CTS_{j,i}$ for node $i \in \mathrm{N}$, the number of RTS and CTS frames node $i \in \mathrm{N}$ sent. We define medium access ratio of node $i \in \mathrm{N}$ as evaluated by its neighbor node $j \in \mathrm{G}_i$ as $MAR_{j,i} = RTS_{j,i} / (RTS_{j,i} + CTS_{j,i})$, if $(RTS_{j,i} + CTS_{j,i}) \neq 0$. At each behavior evaluation slot k where a neighbor $j \in \mathrm{G_i}$ of node $i \in \mathrm{N}$ is evaluating its behavior, $MAR_{j,i}$ is calculated, $RTS_{j,i}$ and $CTS_{j,i}$ are reset, and $a^k_{j,i,c} = MAR_{j,i}$ is assigned (evaluation metric c corresponds to medium access ratio).

Figure 83, Figure 84, and Figure 85 show $\overline{A}_{i,c}, \forall i \in \mathrm{N}$ under three different traffic loads with misbehaving nodes at $P_{selfish} = 75\%$. From the figures, it is clear that $\overline{A}_{i,c}, \forall i \in \mathrm{N} \setminus \mathrm{M}$ is higher than $\overline{A}_{i,c}, \forall i \in \mathrm{M}$. In Figure 86, $\overline{A}_{\mathrm{diff},\, c}$ is shown for $P_{selfish} = 75\%$ under five different traffic loads, illustrating a difference of about 20 between the average score of cooperative nodes and that of misbehaving nodes. Figure 87 compares $\overline{A}_{\mathrm{diff},\, c}$ averaged over the five traffic loads for three different values of $P_{selfish}$

($P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$), illustrating significant difference between the average score of cooperative nodes and that of misbehaving nodes at $P_{selfish} = 100\%$ but less striking difference at $P_{selfish} = 50\%$ and $P_{selfish} = 75\%$. Accordingly, the accuracy of this evaluation metric is shown to be a function to the value of $P_{selfish}$.

Figure 88 evaluates the sensitivity of this evaluation metric to change in node behavior based on the values of $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$, where $i = Node\ A$. In the Figure, $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$ for the two slots when *Node A* is misbehaving are compared to the two slots when it is acting cooperatively. From the figure, the behavior of *Node A* at times when it acts cooperatively is not easily distinguished from times when it misbehaves (the same observation was made using $P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$). Hence, the sensitivity of this evaluation metric is not always guaranteed, as it depends heavily on the traffic load.

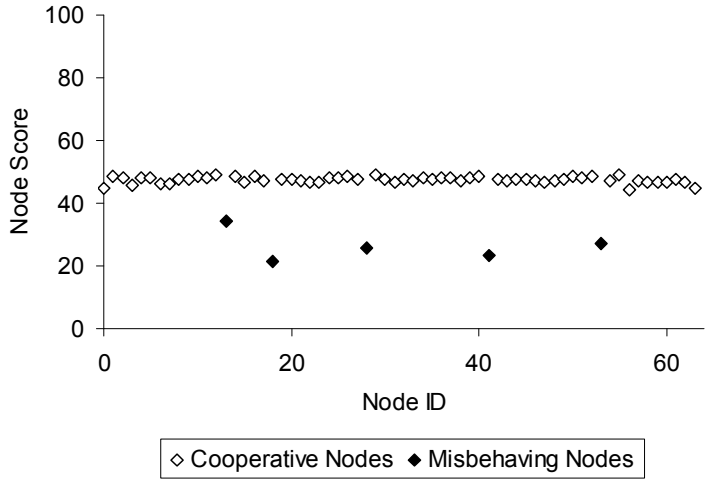

Figure 83 $\overline{A}_{i,c}$ Medium Access Ratio – 20 Flows – Five Misbehaving Nodes –

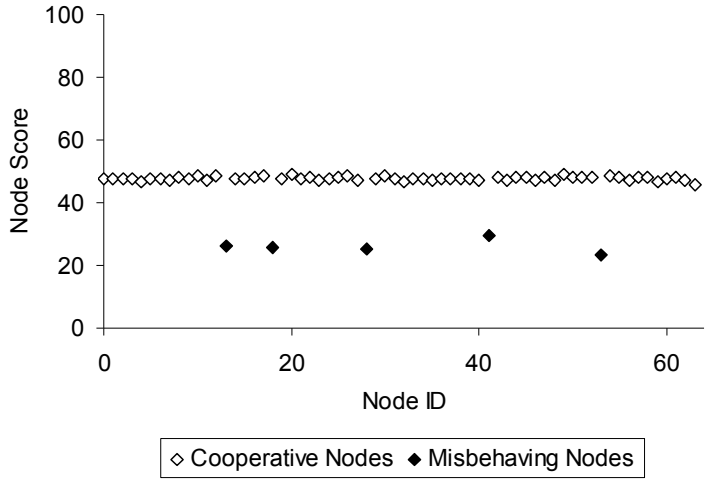

Figure 84 $\overline{A}_{i,c}$ Medium Access Ratio – 60 Flows – Five Misbehaving Nodes –

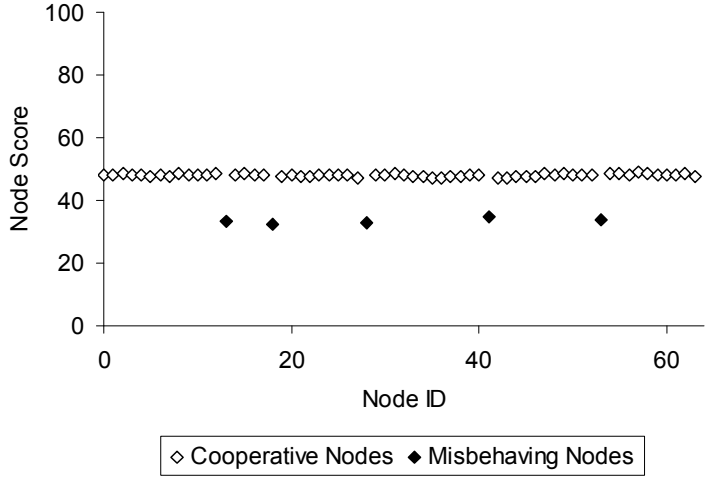

Figure 85 $\overline{A}_{i,c}$ Medium Access Ratio – 200 Flows – Five Misbehaving Nodes –

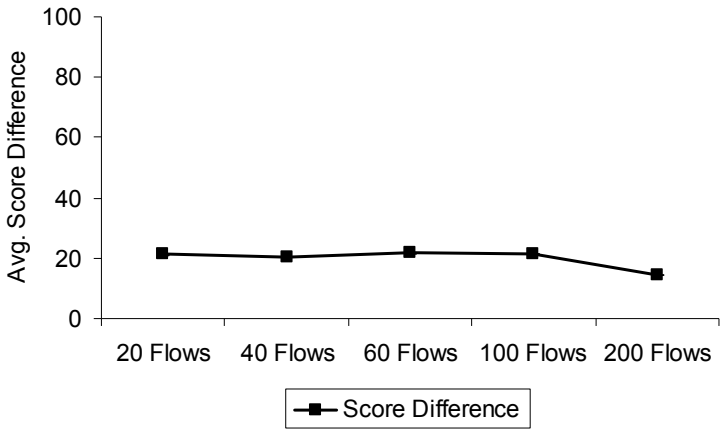

Figure 86 Accuracy ($\overline{A}_{\text{diff},\ c}$) of Medium Access Ratio under different traffic loads: ($P_{selfish} = 75\%$) .

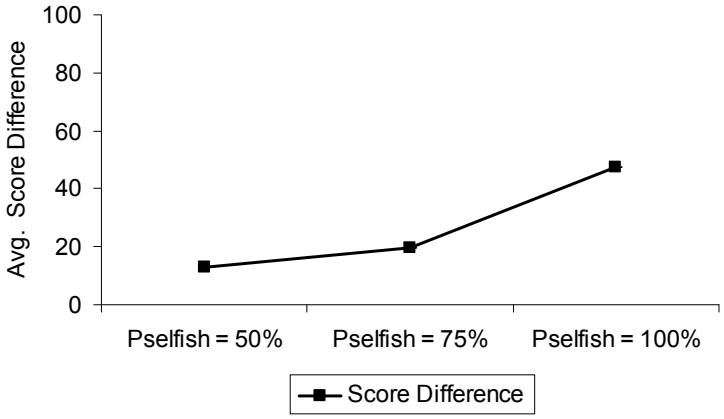

Figure 87 Accuracy ($\overline{A}_{\text{diff},\ c}$) of Medium Access Ratio under different values of $P_{selfish}$.

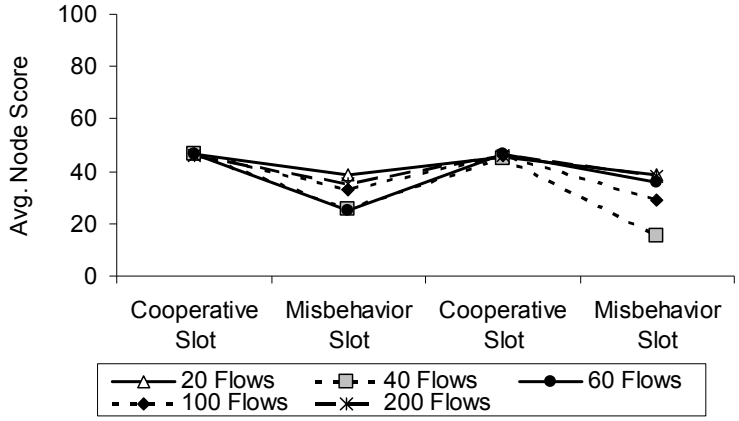

Figure 88 Sensitivity of Medium Access Ratio to changes in node behavior ($P_{selfish} = 75\%$).

2.1.2.2 Frame Collisions

The second evaluation metric we investigate at the data link layer is related to frame collisions. We observe the frequency of collisions of frames encapsulating transport layer data and compare the frequency of collisions at cooperative nodes to that at misbehaving nodes. The motivation to investigate this metric stems from the fact that misbehaving nodes forward fewer data packets, which may decrease the frame collisions they experience. Note that we chose to focus on collisions of frames that encapsulate transport layer data, as opposed to all frame collisions, since the majority of frame collisions affect control packets (e.g. network layer control packets such as RREQ, RREP, RRER, and Hello messages as well as data link layer control frames such as RTS and CTS frames)[3]. Hence, it becomes difficult to distinguish between cooperative and misbehaving nodes based on all frame collision events because misbehaving nodes

[3] Considering that broadcast messages are not protected by RTS frames in 802.11 a significant number of collision events affect network layer RREQ packets.

impact transport layer data packets only (node misbehavior does not impact control packets) which are a small proportion of frame collision events (between 1%-2%).

Each neighbor $j \in \mathrm{G}_i$ of node $i \in \mathrm{N}$ monitors the frame collisions that node $i \in \mathrm{N}$ experiences by keeping count of $COL_{j,i}$. At each behavior evaluation slot k where neighbor $j \in \mathrm{G}_i$ of node $i \in \mathrm{N}$ is evaluating its behavior, $a_{j,i,c}^{k} = COL_{j,i}$ is assigned (evaluation metric c corresponds to frame collisions) and then $COL_{j,i}$ is reset. Figure 89, Figure 90, and Figure 91 show $\overline{A}_{i,c}, \forall i \in \mathrm{N}$ under three different traffic loads with misbehaving nodes at $P_{selfish} = 75\%$. From the figures, it is clear that $\overline{A}_{i,c}, \forall i \in \mathrm{N} \setminus \mathrm{M}$ are higher than $\overline{A}_{i,c}, \forall i \in \mathrm{M}$ but within close range. In Figure 92, $\overline{A}_{\mathrm{diff},\, c}$ is shown for $P_{selfish} = 75\%$ under five different traffic loads, illustrating that the difference between the average score of cooperative nodes and that of misbehaving nodes decreases with traffic load, reaching almost zero at 200 flows (a similar observation was made using $P_{selfish} = 50\%$ and $P_{selfish} = 100\%$). This is expected, as collisions increase with the increase in traffic load. Figure 93 compares $\overline{A}_{\mathrm{diff},\, c}$ averaged over the five traffic loads for three different values of $P_{selfish}$ ($P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$), illustrating a difference of about 14 between the average score of cooperative nodes and that of misbehaving nodes for all three different values of $P_{selfish}$. In summary, the accuracy of this evaluation metric is a function of traffic load but is unaffected by the value of $P_{selfish}$.

Figure 94 evaluates the sensitivity of this evaluation metric to change in node behavior based on the values of $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$, where $i = Node\ A$. In the Figure, $\overline{A}_{i,c}^{1}$, $\overline{A}_{i,c}^{2}$, $\overline{A}_{i,c}^{3}$, and $\overline{A}_{i,c}^{4}$ for the two slots when *Node A* is misbehaving are compared to the two slots when it is acting cooperatively. From the figures, the behavior of *Node A* at times when it acts cooperatively cannot always be distinguished from times when it misbehaves based on the scores it is assigned using frame collisions (same observation was made using $P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$). Hence, the sensitivity of this evaluation metric to changes in node behavior is a strong function of traffic load.

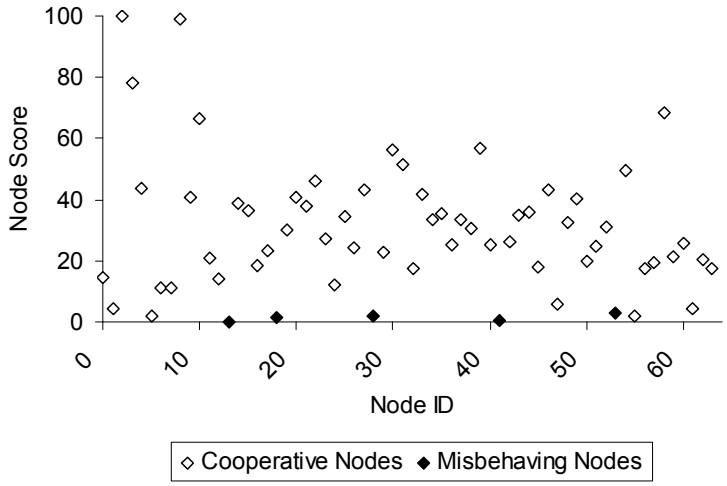

Figure 89 $\overline{A}_{i,c}$ Frame Collisions – 20 Flows – Five Misbehaving Nodes –

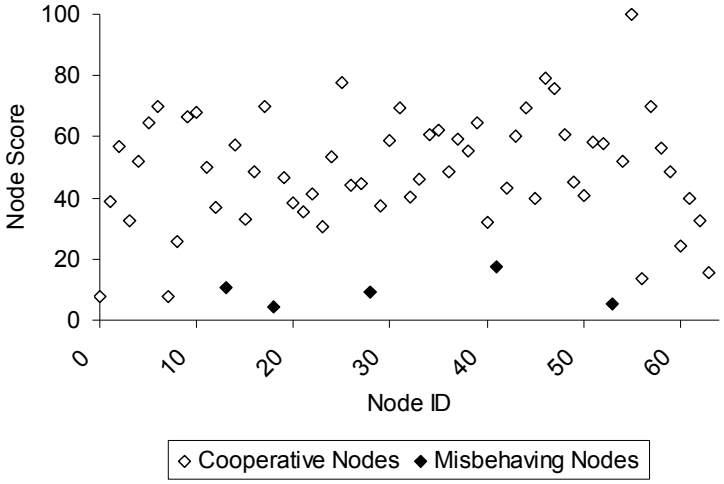

Figure 90 $\overline{A}_{i,c}$ Frame Collisions – 60 Flows – Five Misbehaving Nodes –

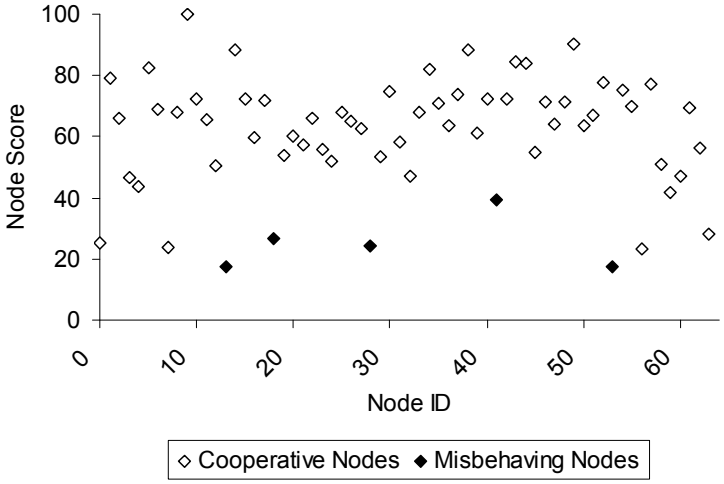

Figure 91 $\overline{A}_{i,c}$ Frame Collisions – 200 Flows – Five Misbehaving Nodes –

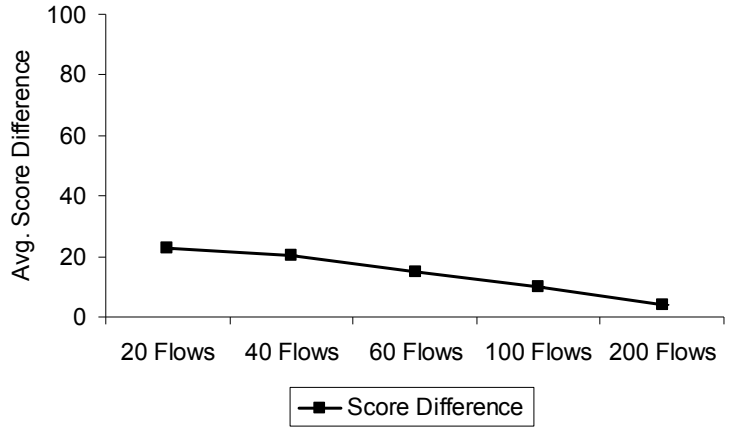

Figure 92 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Frame Collisions under different traffic loads: ($P_{selfish} = 75\%$)

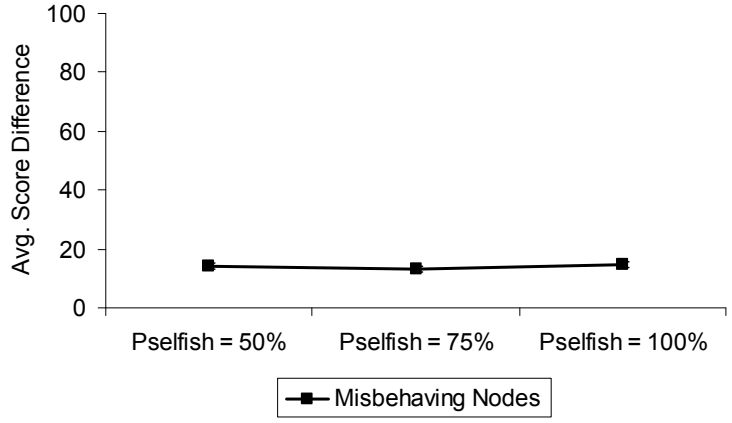

Figure 93 Accuracy ($\overline{A}_{\text{diff},\,c}$) of Frame Collisions under different values of $P_{selfish}$.

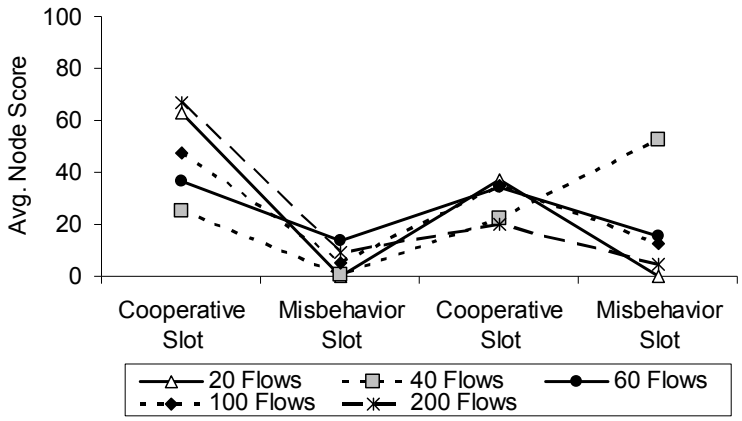

Figure 94 Sensitivity of Frame Collisions to changes in node behavior ($P_{selfish} = 75\%$).

Based on our evaluations, we consider packet forwarding ratio and packet drop ratio to be good evaluation metrics, and medium access ratio as to be an average evaluation metric. In the remainder of this chapter, we will use these evaluation metrics in our discussions.

2.1.3. Evaluation Metric Estimation

The accuracy of the data used to assign a score to a node is key to how well this score reflects node behavior. With respect to packet forwarding ratio, for example, the accuracy of the count of data packets a node forwards and the count of data packets routed through it are key to the accuracy of its behavior evaluation.

There are two commonly adopted approaches for collecting data needed by an evaluation metric: relying on local monitoring and sharing with other nodes. Nodes can rely on monitoring their local neighborhood (i.e. operate in promiscuous mode) as well as their network and MAC layer activities to collect data required for the evaluation metric. Also, nodes can share this data with one another. In this section, we will discuss possible ways a node can collect data needed to calculate packet forwarding ratio, packet drop ratio, and medium access ratio. We implement our data collection approach for medium

access ratio since it is shown, as discussed above, to provide the lowest accuracy of the three metrics under perfect information (thus providing a worst case scenario), evaluate the error in the collected data, and assess the impact of the error on the accuracy of the system.

2.1.3.1 Packet Forwarding Ratio

Packet forwarding ratio requires knowledge of the number of data packets forwarded by each node and the number of data packets routed through each. A node must have this information for each of its neighbors with adequate level of accuracy to be able to accurately evaluate each based on this evaluation metric. To estimate the number of packets routed through each of its neighbors and the number of packets forwarded by each, a node may be able to gather data by listening in promiscuous mode. Relying on destination based acknowledgement as a proof of data packets forwarded by a node is an alternate approach [64]. This data can also be gathered collaboratively through sharing, where each node shares with other nodes the number of data packets it forwarded through each of its neighbors and the number of data packets each routed through it.

2.1.3.2 Packet Drop Ratio

Packet drop ratio requires knowledge of the number of data packets dropped by and the number of data packet routed through a node. A node must have this information for each of its neighbors with adequate level of accuracy to be able to accurately evaluate each based on this evaluation metric. A node can estimate the number of packets routed through each of its neighbors as discussed before. Estimating the number of packets dropped by each neighbor is more difficult since packet drop is a silent operation.

However, a node can rely on promiscuous monitoring of its neighbors and assume that a neighbor dropped a data packet that is routed through it if it is not overheard forwarding it. Relying on destination-based acknowledgement is another option, whereby an overheard retransmission of a data packet that was previously forwarded through a neighbor can be interpreted as a drop event (i.e. the neighbor dropped a data packet, which caused a retransmission).

2.1.3.3 Medium Access Ratio

Medium access ratio requires knowledge of the number of RTS frames and the number of CTS frames sent by a node. Collecting RTS frame count is simple, since they are received and processed by all nodes within transmission range. Hence, a node can keep an RTS frame counter for each of its neighbors and increment the appropriate neighbor's RTS counter whenever it receives an RTS from that neighbor. Knowing the number of CTS frames sent by a neighbor, on the other hand, is not possible since CTS frames do not include a source address [63]. However, an estimation of this count can be obtained by assuming a node will send a CTS frame in response to an RTS frame that it receives. Hence, a node can increment the appropriate neighbor's CTS counter whenever it receives an RTS frame destined to that neighbor.

We implemented the above approach for estimating RTS and CTS frame counts using ns-2. We ran a number of simulations and collected for each the estimates $\widehat{A}_{j,i,c} = \{\widehat{a}^1_{j,i,c},...,\widehat{a}^n_{j,i,c}\}$ $\forall i \in \mathrm{N}$ and $j \in \mathrm{G}_i$ in n evaluation slots (evaluation metric c is medium access ratio). Let $A_{j,i,c} = \{a^1_{j,i,c},...,a^n_{j,i,c}\}$ be the corresponding sequence of scores assigned to node $i \in \mathrm{N}$ by its neighbor $j \in \mathrm{G_i}$ in the same n evaluation slots

assuming 100% accuracy. We calculated the *Root Mean Square Error (RMSQE)* in $\overline{\overline{A}}_{i,c}, \forall i \in \mathrm{N}$ as $RMSQE_{i,c} = \sqrt{\frac{1}{|G_i|}\sum_{j \in G_i}(\overline{A}_{i,c} - \overline{\overline{A}}_{i,c})^2}$. Figure 95 shows $\frac{1}{|\mathrm{N}|}\sum_{i \in \mathrm{N}} RMSQE_{i,c}$, the average $RMSQE_{i,c}, \forall i \in \mathrm{N}$, as well as $\frac{\frac{1}{|\mathrm{N}|}\sum_{i \in \mathrm{N}} RMSQE_{i,c}}{\frac{1}{|\mathrm{N}|}\sum_{i \in \mathrm{N}} \overline{A}_{i,c}}$, the ratio of the average $RMSQE_{i,c}, \forall i \in \mathrm{N}$ to the average value of $\overline{A}_{i,c}, \forall i \in \mathrm{N}$ under different traffic loads. From the Figure, it is clear that the *RMSQE* is about 20% and that the ratio to the average node score is about 45%. Note that the approach we implemented for estimating medium access ratio is very simple and can further be improved to reduce the error.

Table 6 Estimation Error

Evaluation Metric	Cooperative	Misbehaving
Packet Forwarding Ratio	$\overline{\overline{A}}_{i,c} = \overline{A}_{i,c} - (e * \overline{A}_{i,c}), \forall i \in N \setminus M$	$\overline{\overline{A}}_{i,c} = \overline{A}_{i,c} + (e * \overline{A}_{i,c}), \forall i \in M$
Packet Drop Ratio	$\overline{\overline{A}}_{i,c} = \overline{A}_{i,c} + (e * \overline{A}_{i,c}), \forall i \in N \setminus M$	$\overline{\overline{A}}_{i,c} = \overline{A}_{i,c} - (e * \overline{A}_{i,c}), \forall i \in M$
Medium Access Ratio	$\overline{\overline{A}}_{i,c} = \overline{A}_{i,c} - (e * \overline{A}_{i,c}), \forall i \in N \setminus M$	$\overline{\overline{A}}_{i,c} = \overline{A}_{i,c} + (e * \overline{A}_{i,c}), \forall i \in M$

In the remainder of this section, we assume that all evaluation metrics exhibit similar estimation error as medium access ratio and assess the impact of such error on the accuracy of packet forwarding ratio and packet drop ratio (note that medium access ratio

had the worst accuracy among the three evaluation metrics). Given $\overline{A}_{i,c}$, the average score of a node $i \in \mathrm{N}$ by a packet forward ratio or packet drop ratio, we apply a random error e with mean 50% to the node's score as an estimation error. Accordingly, the average node score is calculated after applying the error e, as summarized in Table 6.

We then re-plot Figure 69 and Figure 75 (shown in Figure 96 and Figure 97) to assess the effect of the error on the accuracy achieved by packet forwarding ratio and packet drop ratio. Figure 96 and Figure 97 compare $\overline{\widehat{A}}_{\mathrm{N}\setminus\mathrm{M},\,c} - \overline{\widehat{A}}_{\mathrm{M},\,c}$, the difference between the average score of cooperative nodes and that of misbehaving nodes over all traffic loads recalculated using the values $\overline{\widehat{A}}_{i,c}, \forall i \in N$ obtained as described in Table 2. We compare these values for $P_{selfish} = 50\%$, $P_{selfish} = 75\%$, and $P_{selfish} = 100\%$ showing that a significant difference still exists between the average score of cooperative nodes and that of misbehaving nodes. Hence, even with an estimation error of 20%, both evaluation metrics are still able to maintain good accuracy.

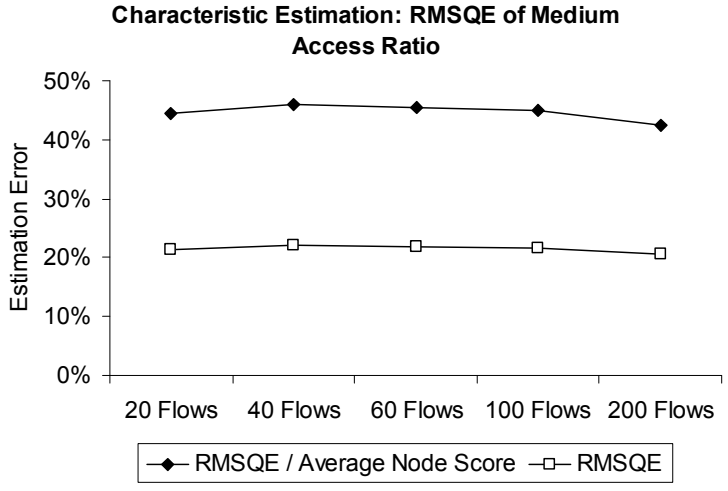

Figure 95 Estimating Error in Medium Access Ratio

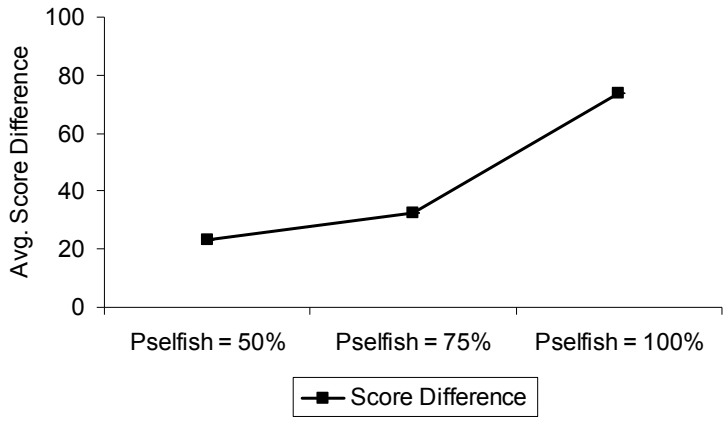

Figure 96 Packet Forwarding Ratio: 50% estimation error

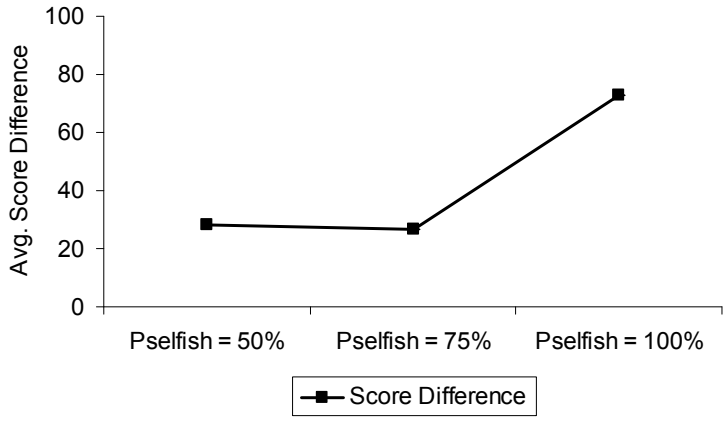

Figure 97 Packet Drop Ratio: 50% estimation error

2.2. Slotted Evaluation Function

In this section we discuss how to enhance the sensitivity and adaptability of the evaluation function. Sensitivity of the evaluation function reflects its ability to capture changes in node behavior. We found that an approach where node behavior evaluation is slotted in time can significantly enhance the sensitivity of the evaluation function. In this approach, a node's behavior is evaluated and assigned a score based on its behavior within each time slot, irrespective of its past behavior. Given a slot duration p , a node $i \in \mathrm{N}$ is assigned a score $a_{j,i,c}^{k}$ by a node $j \in \mathrm{G_i}$ based on its behavior within evaluation

slot k (between times $p(k-1)$ and pk , $k>0$) using evaluation metric $c \in \mathrm{C}$. The instantaneous change in node i 's behavior can be assessed by comparing $a_{j,i,c}^{k-1}$ to $a_{j,i,c}^{k}$.

In a time slotted behavior evaluation function, the value of the slot duration p affects the accuracy and the promptness of the reputation management system. Consider a static network or one with low traffic activity. Selecting a short slot duration for such a scenario reduces the system's accuracy. A short slot may not comprise enough packet forwarding activity to be representative of node behavior, which may result in false positives (cooperative nodes incorrectly classified as misbehaving) or false negatives (misbehaving nodes incorrectly classified as cooperative). On the other hand, consider a highly mobile network scenario or one with high traffic activity. Since node behavior evaluation is triggered at the end of each slot, unnecessarily increasing the slot duration may impact the system's promptness. Accordingly, the slot duration must adapt according to the network activity (traffic activity, node mobility, etc.). As the network activity increases the slot duration should decrease. Note that there is always a tradeoff between slot duration and overhead. Decreasing the slot duration to achieve better accuracy and promptness results in increased computational and possibly communication overhead.

To assess this relationship between network activity and slot duration, we conducted a sensitivity analysis using a network of $|\mathrm{N}|=64$ nodes $|\mathrm{M}|=5$ misbehaving nodes utilizing packet forwarding ratio as the evaluation metric (herein denoted as $\hat{c}$). We tested values of slot duration, in seconds, in $P=\{1,2,4,8,18,36,75,150,300,600\}$. For each slot duration $p \in P$ we observe the system's accuracy and promptness. We

measure accuracy based on $\overline{A}_{\text{diff},\hat{c}}$. For each $p \in P$ we calculate $\overline{A}_{\text{diff},\hat{c}}$, which we denote as $\overline{A}_{\text{diff},\hat{c},p}$. Note that $0 \le \overline{A}_{\text{diff},\hat{c},p} \le 100$ and that as the value of $\overline{A}_{\text{diff},\hat{c},p}$ decreases it becomes more difficult to distinguish between cooperative and misbehaving nodes, which increases false positives as well as false negatives (i.e. decreases accuracy). To model network activity, we use a fixed network topology with two different traffic activities, high (200 Flows) and low (20 Flows)[4]. Figure 98 shows the value of $\overline{A}_{\text{diff},\hat{c},p}$ for all slot durations $p \in P$ under high and low traffic activities indicating that as the slot duration increases the value of $\overline{A}_{\text{diff},\hat{c},p}$ increases.

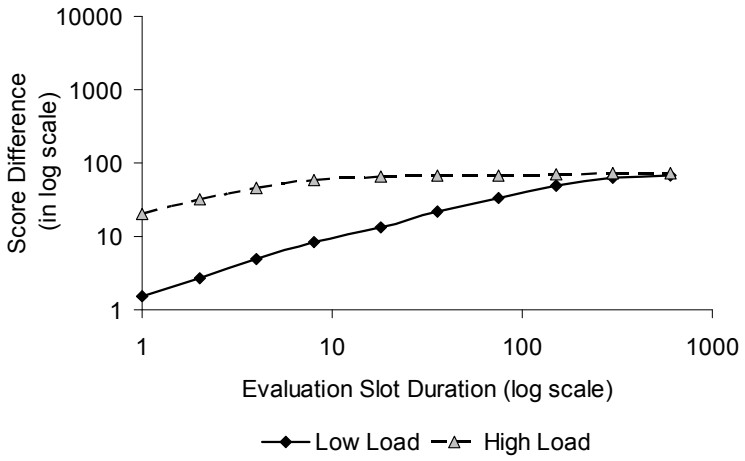

Figure 98 System Accuracy: comparing $\overline{A}_{\text{diff},c,p}$ for high and low traffic activities

$\forall p \in P$

[4] Alternatively, network activity can also be modeled by increasing node mobility under the same traffic activity.

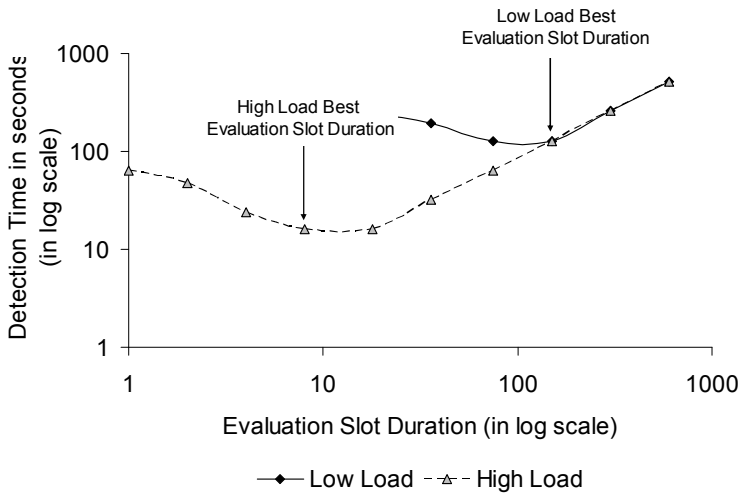

Figure 99 System Promptness: average misbehavior detection time for high and low traffic activities $\forall p \in P$

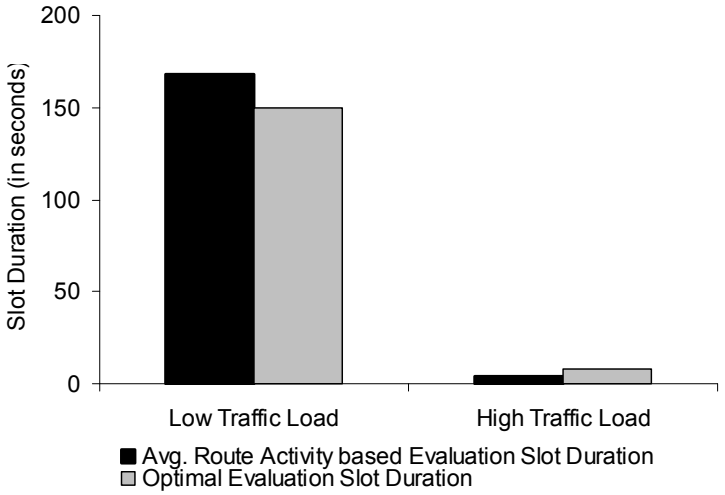

Figure 100 Average slot duration for high/low traffic activities

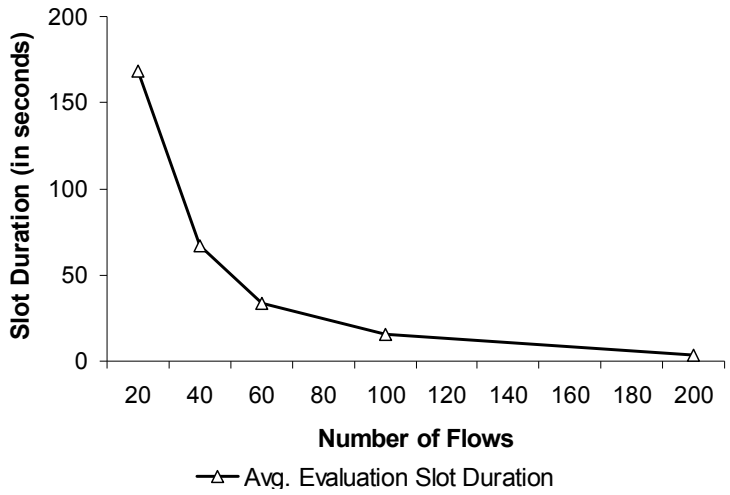

Figure 101 Average evaluation period for 5 traffic activities

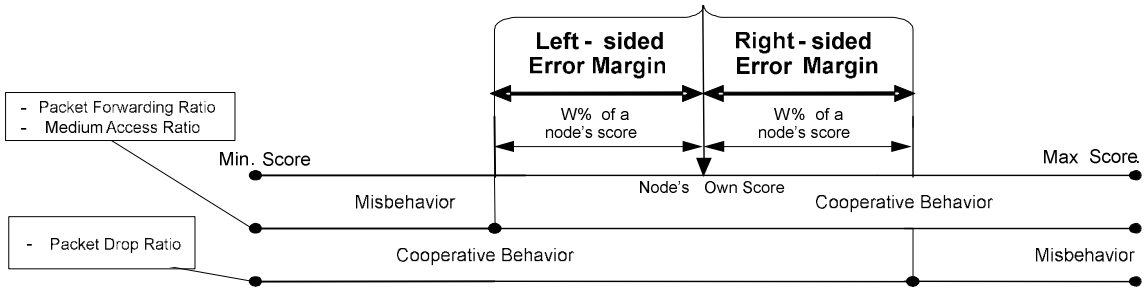

Figure 102 Adaptive approach: Left-sided error margin for packet forwarding ratio and medium access ratio. Right-sided for packet drop ratio.

Increasing the slot duration allows for more packet forwarding activity to take place within the slot so as to contain sufficient representative node behavior. As a result, the evaluation function has enough packet forwarding activity to accurately evaluate node behavior. Note the clear difference between low and high traffic activities. At a high traffic activity, high values of $\overline{A}_{\text{diff},\hat{c},p}$ occur much faster as the number of interactions within a slot increases. If the system for instance requires $\overline{A}_{\text{diff},\hat{c},p} \geq 25$ to achieve a

target accuracy level, this corresponds to slot duration $p \geq 2$ seconds for the high traffic activity and $p \geq 75$ seconds for the low traffic activity.

The tradeoff to the increase in accuracy as a result of increasing slot duration is the decrease in promptness. Promptness is the earliest time the scores of cooperative nodes can be clearly distinguished from those of misbehaving nodes. Figure 99 plots the promptness of the system with the required accuracy level used earlier ($\overline{A}_{\text{diff},\hat{c},p} \geq 25$) denoting promptness as the time of the first slot where $\overline{A}_{\text{diff},\hat{c},p} \geq 25$. The figure shows that increasing the slot duration may decrease promptness. It also shows that there is a particular slot duration where the required accuracy level is reached fastest. This is the ideal slot duration where enough packet forwarding activity is present at the earliest time to distinguish between the behavior of cooperative nodes and that of misbehaving nodes at the required accuracy level. Lower slot durations do not take into consideration enough packet forwarding activity, while higher slot durations unnecessarily delay node evaluation, both resulting is lower promptness. From the figure, the slot duration in which misbehaving nodes are detected fastest at the required accuracy level is $p = 8$ seconds for the high traffic activity case and $p = 150$ seconds for the low traffic activity case.

We propose an adaptive approach whereby the slot duration is adapted according to the network activity. We consider route activity as a measure of network activity. Frequency of route establishment, route update, and route expiration events increases as network activity increases. Whenever a route establishment, route update, or a route expiration event occurs, each node that becomes aware of the update (the route event affects one or more fields in the node's routing table) triggers the behavior evaluation

function to evaluate its neighbors. Hence, the slot duration differs from one node to another and from time to time according to the network activity each node is witnessing. At high network activity, routes are likely to change frequently, resulting in a short slot duration. At low network activity, routes are likely to be more stable, resulting in a long slot duration. To assess the effectiveness of this approach we compare in Figure 100 the average slot duration for periods of high and low traffic activities when this approach is employed compared to the ideal values from Figure 99. Figure 101 shows the average slot duration of cooperative nodes at five different traffic activity levels, illustrating how the increase in traffic activity decreases the slot duration.

3. Adaptability of the Detection Function

In this section we discuss the adaptability of the detection function. Typically, the *threshold score* used by the detection function is a fixed value that is identically set for all nodes in the network. Its value is usually assigned to reflect acceptable node behavior under a particular network condition (e.g. moderate traffic load). Consider the packet forwarding ratio evaluation metric, for example: if the score of most cooperative nodes under moderate traffic activity is above 80%, the *threshold score* can be set to 70% to allow for a margin of error. However, if the network conditions change (significant increase or decrease in traffic activity, for example), the *threshold score* selected may no longer be accurate, resulting in inaccuracy in the detection function. Accordingly, the value of *threshold score* must be adapted by each node to reflect its surrounding network conditions.

To enhance the adaptability of the detection function, we propose an approach whereby cooperative nodes use their own behavior as a benchmark for making decisions

about others' behavior. Consider a cooperative node $j \in \mathrm{N} \setminus \mathrm{M}$ that is evaluating the behavior of its neighbor $i \in \mathrm{G_j}$ according to evaluation metric $c \in \mathrm{C}$. At each evaluation slot k, node j evaluates its own behavior based on the same evaluation metric c and assigns itself a score $a_{j,c}^{k}$. Since cooperative nodes within close proximity usually experience similar network conditions, a node's evaluation of its own behavior can provide a good estimate of the expected score of a cooperative node under the current local conditions surrounding the node. Note that this value will continuously adapt as the node's local network conditions change. Node j can then use $a_{j,c}^{k}$ as the *threshold score* for the evaluation instance k while allowing for a margin of error W. Node j then evaluates the behavior of each of its neighbors $i \in \mathrm{G_j}$ and based on evaluation metric c it assigns each a score $a_{j,i,c}^{k}$. The score $a_{j,i,c}^{k}$ is then compared to $a_{j,c}^{k}$ and depending on the relative values of $a_{j,c}^{k}$ and $a_{j,i,c}^{k}$ and the value of W, node j can identify each node $i \in \mathrm{G_j}$ as either cooperative or misbehaving. If packet forwarding ratio or medium access ratio is the evaluation metric employed, node j would consider a node $i \in \mathrm{G}_j$ misbehaving if $a_{j,i,c}^{k} < (a_{j,c}^{k} - W * a_{j,c}^{k})$. Other nodes $i \in \mathrm{G}_j$ where $a_{j,i,c}^{k} \geq (a_{j,c}^{k} - W * a_{j,c}^{k})$ are considered cooperative. If packet drop ratio is used, node j would consider a node $i \in \mathrm{G}_j$ misbehaving if $a_{j,i,c}^{k} > (a_{j,c}^{k} + W * a_{j,c}^{k})$. Nodes $i \in \mathrm{G}_j$ where $a_{j,i,c}^{k} \leq (a_{j,c}^{k} + W * a_{j,c}^{k})$ are considered cooperative (Figure 102). We conducted a preliminary comparison between the accuracy of two systems, one with an adaptive detection function and another with a non-adaptive detection function, in a fixed ad hoc

network with a dynamically changing traffic load (traffic load fluctuates between high and low). Our results showed that the adaptive system outperforms the non-adaptive system.

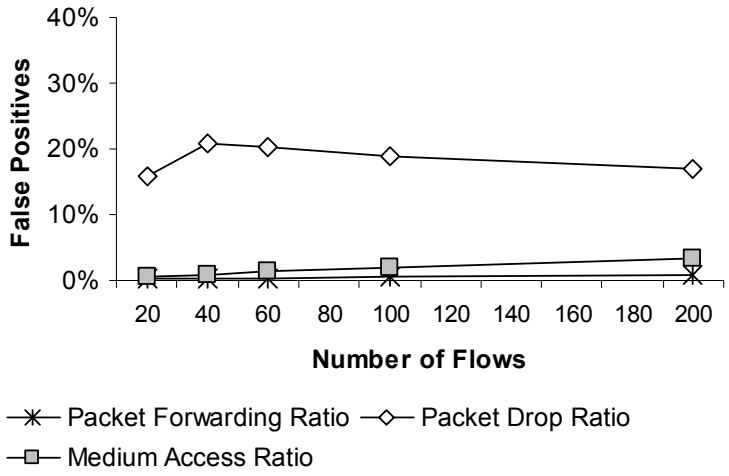

Figure 103 False positives of the 3 evaluation metrics

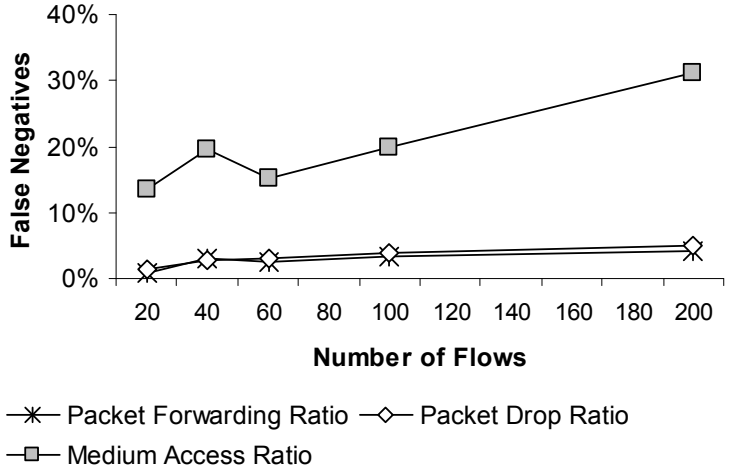

Figure 104 False negatives of the 3 evaluation metrics

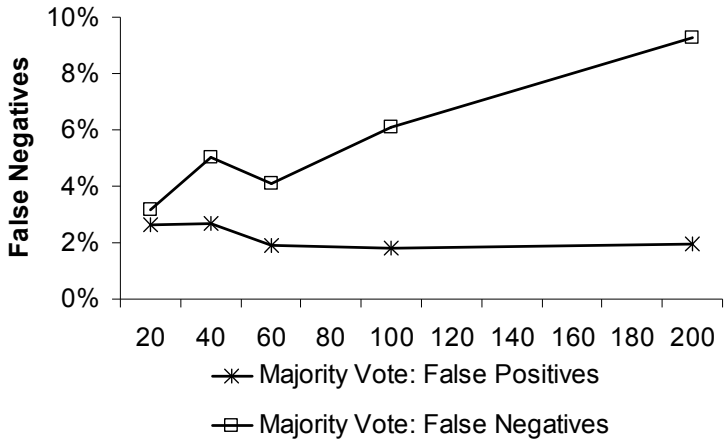

Figure 105 Metric combination

3.1. Combining Metrics

In this section, we investigate whether the system accuracy can be enhanced by simultaneously utilizing multiple evaluation metrics from different layers of the protocol stack to distinguishing between cooperative and misbehaving nodes. Given two nodes, $i \in \mathrm{N}$ and $j \in \mathrm{G}_i$, node j evaluates the behavior of node i based on each evaluation metric $c \in \mathrm{C}$ independently and the outcome of each is fed into the detection function. The detection function at node j then makes a decision about node i's behavior (as either cooperative or misbehaving) based on each individual metric. Consider $d_{j,i,c} = \begin{cases} 1, & \text{if node } i \text{ is misbehaving} \\ 0, & \text{if node } i \text{ is cooperative} \end{cases}$ to be the decision about node i's behavior by node j according to evaluation metric $c \in \mathrm{C}$. Node j then considers all decisions $d_{j,i,c}, \forall c \in C$ to reach a final decision $d_{j,i}$. The final decision $d_{j,i}$ could be

$d_{j,i} = \prod_{\bar{c} \in \bar{C}} d_{j,i,c}$, which represents unanimous agreement, or

$$d_{j,i} = \begin{cases} 1 \text{ (node } i \text{ is misbehaving) if } \sum_{\bar{c} \in \bar{C}} d_{j,i,c} > \frac{|C|}{2} \\ 0 \text{ (node } i \text{ is cooperative) if } \sum_{\bar{c} \in \bar{C}} d_{j,i,c} \leq \frac{|C|}{2} \end{cases}$$, which represents majority vote.

Using $W = 25\%$ [5], we evaluated the system accuracy when using a single evaluation metric compared to combining metrics based on unanimous agreement and majority voting approaches. Figure 103 and Figure 104 show the accuracy of the system (in terms of false positives and false negatives) when using only one evaluation metric. The system's accuracy when using packet forwarding ratio yields the lowest false positives (bounded by 1%) and false negatives (bounded by 5%) under all traffic loads. Packet drop ratio shows false negatives comparable to that of packet forwarding ratio under all traffic loads, however false positives are much higher, reaching about 21% under some traffic loads. Medium access ratio, on the other hand, shows low false positives (bounded by 4%) but false negatives are high (reaching 31% under 200 flows).

Combining metrics and making decisions based on unanimous agreement resulted in negligible false positives but very high false negatives. This is expected since a node can only be identified as misbehaving if the decisions by all metrics simultaneously identify the node as such. Majority voting on the other hand results in low false positives (bounded by 3%) and false negatives (bounded by 10%) as shown in Figure 105. This indicates better system accuracy compared to using packet drop ratio only or medium access ratio only. However, the accuracy of the system based on packet forwarding ratio

[5] The value of W impacts the system's accuracy. We conducted a sensitivity analysis which showed that using $W = 25\%$ results in reasonable system accuracy. Due to space constraints, we omit the detailed discussion of this evaluation.

only is better than majority voting. The inaccuracies of packet drop ratio and medium access ratio (i.e. high false positives and high false negatives respectively) significantly impact the accuracy of the combined metrics.

While our results may suggest that a highly accurate evaluation metric such as packet forwarding ratio can be better than a combination of metrics with varying accuracies, it should not discount the potential of combining metrics to enhance system accuracy. Considering a system that can only use packet drop ratio and medium access ratio, both metrics can potentially be combined in such a way that they complement each other's inaccuracies (i.e. high false positives in packet drop ratio and high false negatives in medium access ratio).

3.2. Margin of Error

In this section we evaluate the impact of the margin of error on the system's accuracy. In our evaluation, a negative value of W means that a node accepts a neighbor as cooperative only if the neighbor has a higher score than its own. Figure 106 and Figure 107 show the false positives and false negatives averaged over all traffic loads using different values of W. From the figures, the system's accuracy is shown to be sensitive to the choice of W when using packet forwarding ratio or medium access ratio. There is a tradeoff between low margin of error, which results in high false positives, and high margin of error, which results in high false negatives. At $W = 0.25\%$, false positives and false negatives are reasonably low for packet forwarding ratio (0.47% and 2.68% respectively) and for medium access ratio (1.67% and 19.80% respectively).

The system's accuracy is shown to be unaffected by the value of W when packet drop ratio is employed. This is because the scores assigned to a cooperative node based

on packet drop ratio are usually small since cooperative nodes usually forward a lot more packets than they drop. The value of $a_{j,c}^k + W * a_{j,c}^k$ is very close to $a_{j,c}^k$, and hence, this evaluation metric is not very sensitive to the range of W values we used.

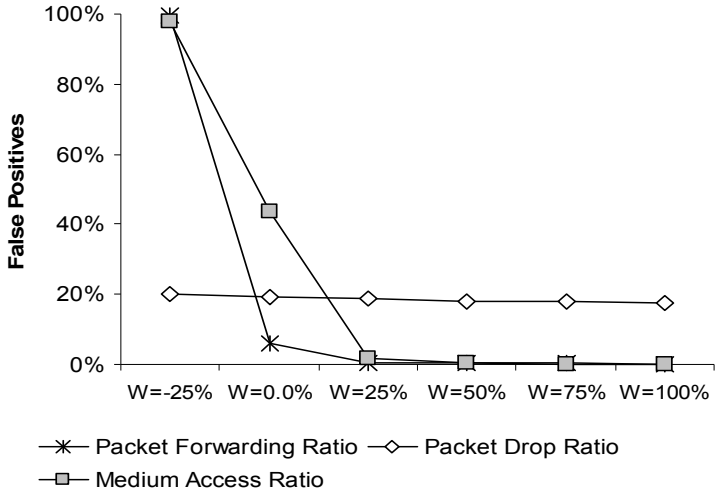

Figure 106 Impact of margin of error on the system's accuracy: false positives

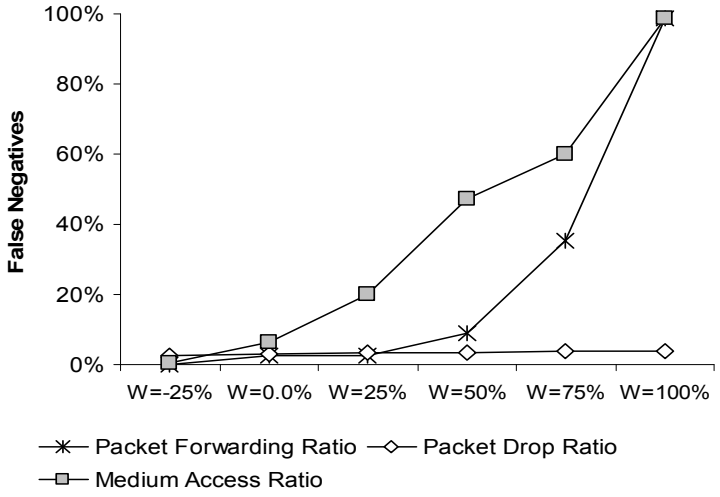

Figure 107 Impact of margin of error on the system's accuracy: false negatives

4. Summary

In this chapter, we highlighted the importance of adaptation in reputation management systems in dynamic network environments such as ad hoc networks. We

identified some of the requirements of an adaptive reputation management system such as sensitivity and adaptation. Focusing on the evaluation function, we proposed a number of evaluation metrics from different layers of the protocol stack and discussed the impact of each metric on the accuracy and sensitivity of the evaluation function. We also introduced a time-slotted approach to enhance the sensitivity of the evaluation function and showed how to adapt the duration of the evaluation slot according to the network activity to enhance the system accuracy and promptness. We also showed how the detection function can adapt to the network conditions by using the node's own behavior as a benchmark to set misbehavior detection parameters.

Chapter VII. Contribution and Future Work

1. Contribution

The contributions of this research include the following:

- The work developed a formal framework for reputation management systems. This framework identifies the necessary components of a reputation management system, the functions of each, and how these components interact with one another. It can serve as a basis for the design and classification of reputation management systems and for comparison between reputation management systems to highlight their similarities and differences.
- There are no standard performance metrics used to evaluate the performance of reputation management systems. This work defines and describes a number of security and performance metrics that can be used for quantitative evaluation and comparison of reputation management systems.
- This research showed that by relying on localized information only it is possible to distinguish between cooperative and misbehaving nodes with high accuracy, low false positives, and low overhead. We introduced the design, implementation, and evaluation of an autonomous reputation management system (ARMS). In ARMS, nodes evaluate each other's behavior using localized information only. The developed reputation management system is particularly applicable to commercial deployments of ad hoc networks, where the environment is decentralized and unstructured and nodes are autonomously managed and controlled.

- This research showed that a sequential probability ratio test can be devised to clearly distinguish between infected routes and clean routes. Sequential analysis works well with the nature of ad hoc networks since it does not require a predetermined number of observations to terminate a test. This suits well the nature of ad hoc networks where the number of observations needed to make a decision about a node's behavior (i.e. identify it as cooperative or misbehaving) is not known apriori.
- In this research, we realized an adaptive reputation management system as one whose functions (i.e. evaluation, detection, and reaction) are sensitive (capture changes in node behavior) and adaptable (adjust according to the network conditions) as well as accurate (able to distinguish between cooperative and misbehaving nodes unambiguously) and prompt (timely detect misbehaving nodes).
- We showed in this research how the sensitivity and accuracy of a reputation management system depends on the evaluation metric employed. We evaluated the accuracy and sensitivity of node behavior using a number of metrics from the network and medium access layers. Packet forwarding ratio was shown to be one of the accurate and sensitive evaluation metrics.
- To enhance the sensitivity of the evaluation function, we introduced a slotted evaluation function and showed that by adapting the slot duration according to the network activity we can adapt the behavior evaluation function to changes in network conditions while maintaining good system accuracy and promptness.
- We showed that by using a node's own behavior as benchmark for making decisions about others' behavior the detection function can adapt to changes in network conditions.

2. Future Work

Extensions of this work should focus on two major research points.

- Perform a comprehensive comparison between an adaptive reputation management system and a non-adaptive reputation management system. The main challenge of this work is to model a realistic dynamic environment where the performance of reputation management systems may deteriorate if they do not adapt accordingly.
- In this work, we investigated how a reputation management system can adapt to the dynamic change in node behavior. Another approach to adaptation is for a node to adapt its behavior according to the environment it is operating in. One example would be adaptation of a node's behavior in the presence of congestion. A node that senses congestion in the network may decrease its level of participation in the routing functionality (but maintain being cooperative in the packet forwarding functionality) to avoid being penalized (since it may unintentionally drop packets).

References

[1] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, "Cognitive Networks: Adaptation and Learning to Achieve End-to-end Performance Objectives," *IEEE Communications Magazine*, December, 2006.

[2] D. Djenouri, L. Khelladi, and N. Badache, "A Survey of Security Issues in Mobile Ad Hoc and Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 7, pp. 2-28, 2005.

[3] P.-W. Yau and C. J. Mitchell, "Security Vulnerabilities in Ad hoc Networks," *Proceedings of the 7th International Symposium on Communication Theory and Applications (ISCTA)*, pp. 99–104, 2003.

[4] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," *IEEE Computer Magazine*, vol. 35, pp. 54-62, 2002.

[5] D. Chen, J. Deng, and P. K. Varshney, "Protecting Wireless Networks against a Denial of Service Attack Based on Virtual Jamming (research poster)," *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2003.

[6] Y. Rong, S. K. Lee, and H.-A. Choi, "Detecting Stations Cheating on Backoff Rules in 802.11 Networks Using Sequential Analysis," *Proceedings of the 25th Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*, 2006.

[7] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure on-demand routing protocol for ad hoc networks," *Proceedings of the 8th International Conference on Mobile Computing and Networking (MobiCom)*, pp. 12-23, 2002.

[8] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole Detection in Wireless ad hoc Networks," *Rice University Department of Computer Science Technical Report TR01-384*, 2002.

[9] Y. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," *Proceedings of the ACM Workshop on Wireless Security (WiSE)*, pp. 30-40, 2003.

[10] I. Aad, J. Hubaux, and E. Knightly, "Denial of Service Resilience in Ad Hoc Networks," *Proceedings of the 10th International Conference on Mobile Computing and Networking (MobiCom)*, pp. 202 - 215, 2004.

[11] L. Zhou and Z. Haas, "Securing Ad hoc Networks," *IEEE Network Magazine*, vol. 13, 1999

[12] Y. Zhang and W. Lee, "Intrusion Detection in Wireless Ad hoc Networks," *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, pp. 275-283, 2000.

[13] J. P. Hubaux, L. Buttyan, and S. Capkun, "The Quest for Security in Mobile Ad hoc Networks," *Proceedings of the 2nd ACM International Symposium on Mobile ad hoc Networking & Computing (MobiHoc)*, pp. 146–155, 2001.

[14] Y.-C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2002.

[15] P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad hoc Networks," *Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, pp. 27-31, 2002.

[16] S. Yi, P. Naldurg, and R. Kravets, "Security-Aware Routing Protocol for Wireless Ad Hoc Networks," *The 6th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2002.

[17] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA Broadcast Authentication Protocol," *Proceedings of the IEEE Symposium on Security and Privacy*, 2000.

[18] "http://en.wikipedia.org/wiki/Byzantine_fault_tolerance."

[19] A. Haeberlen, P. Kouznetsov, and P. Druschel, "The Case for Byzantine Fault Detection," *Tech. Report, Max Planck Institute for Software Systems*, 2006.

[20] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 3, pp. 382-401, 1982.

[21] L. Buttyan and J.-P. Hubaux, "Enforcing Service Availability in Mobile Ad-hoc WANS," *Proceedings of the 1st ACM International Symposium on Mobile Ad hoc Networking & Computing (MobiHoc)*, pp. 87-96, 2000.

[22] S. Zhong, Y. Yang, and J. Chen., "Sprite: A Simple, Cheat-proof, Credit-based System for Mobile Ad hoc Networks," *Proceedings of the 22nd Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*, vol. 3, pp. 1987- 1997, 2003.

[23] V. Srinivasan, P. Nuggehalli, and R. R. Rao, "Cooperation in Ad Hoc Networks," *Proceedings of the 22nd Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*, 2003.

[24] L. Buttyan and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad hoc Networks," *Proceedings of the ACM/Kluwer Mobile Networks and Applications Journal*, pp. 579–592, 2003.

[25] V. Srivastava, J. Neel, A. MacKenzie, R. Menon, L. A. DaSilva, J. Hicks, J. H. Reed, and R. Gilles, "Using Game Theory to Analyze Wireless Ad Hoc Networks," *IEEE Communications Surveys and Tutorials*, 2005.

[26] R. Axelrod, "The Evolution of Cooperation," *Basic Books, New York*, 1984.

[27] L. A. DaSilva and V. Srivastava, "Node Participation in Ad-hoc and Peer-to-peer Networks: A Game-theoretic Formulation," *Workshop on Games and Emergent Behavior in Distributed Computing Environments*, 2004.

[28] L. Anderegg and S. Eidenbenz, "Ad hoc-VCG: A Truthful and Cost-efficient Routing Protocol for Mobile Ad hoc Networks with Selfish Agents," *Proceedings of the 9th International Conference on Mobile Computing and Networking (MobiCom)*, pp. 245-259, 2003.

[29] S. Zhong, L. E. Li, Y. Liu, and Y. R. Yang, "On Designing Incentive-Compatible Routing and Forwarding Protocols in Wireless Ad-Hoc Networks," *Proceedings of the 11th ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[30] K. Balakrishnan, Deng, and P. Varshney, "TWOACK: Preventing Selfishness in Mobile Ad Hoc Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 4, pp. 2137-2142, 2005.

[31] S. Buchegger and J.-Y. L. Boudec, "A Robust Reputation System for Mobile Ad hoc Networks," *EPFL Technical report No. IC/2003/50*, 2003.

[32] S. Buchegger and J.-Y. L. Boudec, "The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks," *Proceedings of Modeling and Optimization in Mobile Ad Hoc and Wireless Networks (WiOpt)*, 2003.

[33] S. Buchegger and J. Y. LeBoudec, "Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks)," *Proceedings of the 2nd ACM International Symposium on Mobile ad hoc Networking & Computing (MobiHoc)*, pp. 226-236, 2002.

[34] S. Buchegger, C. Tissieres, and J.-Y. L. Boudec, "A Test-Bed for Misbehavior Detection in Mobile Ad-hoc Networks — How Much Can Watchdogs Really Do?," *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 102-111, 2004.

[35] M. Conti, E. Gregori, and G. Maselli, "Towards Reliable Forwarding for Ad Hoc Networks," *Proceedings of Personal Wireless Communications (PWC)*, pp. 790-804, 2003.

[36] P. Dewan, P. Dasgupta, and A. Bhattacharya, "On Using Reputations in Ad hoc Networks to Counter Malicious Nodes," *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 665- 672, 2004.

[37] Q. He, D. Wu, and P. Khosla, "SORI: A Secure and Objective Reputation-based Incentive Scheme for Ad hoc Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 2, pp. 825-830, 2004.

[38] J. Liu and V. Issarny, "Enhanced Reputation Mechanism for Mobile Ad hoc Networks," *Proceedings of the 2nd International Conference on Trust Management (iTrust)*, 2004.

[39] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad hoc Networks," *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, pp. 255-265, 2000.

[40] M. T. Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy, "A Reputation-based Mechanism for Isolating Selfish Nodes in Ad Hoc Networks," *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*, pp. 3-11, 2005.

[41] P. Michiardi and R. Molva, "CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad hoc Networks," *Proceedings of the IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security*, pp. 107-121, 2002.

[42] P.-W. Yau and C. J. Mitchell, "Reputation Methods for Routing Security for Mobile Ad Hoc Networks," *Proceedings of the Joint 1st Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC)*, pp. 130-137, 2003.

[43] K. Aberer, Z. Despotovic, W. Galuba, and W. Kellerer, "The Complex Facets of Reputation and Trust," *Computational Intelligence, Theory and Application, chapter. The complex facets of reputation and trust. Springer*, to appear 2006.

[44] J. Sabater and C. Sierra, "Review on Computational Trust and Reputation Models," *Artificial Intelligence Review*, vol. 24, pp. 33–60, 2005.

[45] E. Chang, F. K. Hussein, and D. Tharam, "Towards Defining an Ontology for Reputation," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2601-2607, 2005.

[46] S. Casare and J. Sichman, "Towards a Functional Ontology of Reputation," *Proceedings of the 4th ACM International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2005.

[47] W. Yu and K. J. R. Liu, "Attack-Resistant Cooperation Stimulation in Autonomous Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 2260-2271, 2005.

[48] H. Yang, J. Shu, X. Meng, and S. Lu, "SCAN: Self-Organized Network-Layer Security in Mobile Ad Hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. Vol. 24, NO. 2, pp. 261-273, 2006.

[49] S. R. J. S. Baras and I. Koutsopoulos, "A Framework for MAC Protocol Misbehavior Detection in Wireless Networks," *Proceedings of the 4th ACM Workshop on Wireless Security (WiSE)*, pp. 33-42, 2005.

[50] S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for mobile ad-hoc networks," *EPFL Technical report No. IC/2003/50*, 2003.

[51] J. Mundinger and J.-Y. L. Boudec, "Reputation in Self-Organized Communication Systems and Beyond," *Proceedings of the Workshop on Interdisciplinary Systems Approach in Performance Evaluation and Design of Computer & Communication Systems (Inter-Perf)*, 2006.

[52] W. Sears, Z. Yu, and Y. Guan, "An Adaptive Reputation-based Trust Framework for Peer-to-Peer Applications," *Proceedings of the 4th IEEE International Symposium on Network Computing and Applications*, pp. 13-20, 2005.

[53] Imad Aad, Jean-Pierre Hubaux, and E. W. Knightly, "Denial of Service Resilience in Ad Hoc Networks," *Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 202 - 215 2004.

[54] A. Wald, "Sequential Analysis, J. Wiley & Sons, New York," 1947.

[55] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-demand Distance Vector (AODV) routing," *IETF RFC 3561*, 2003.

[56] J. Broch, D. B. Johnson, and D. A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad hoc Networks," *Internet-Draft Version 03, IETF*, 1999.

[57] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *Tech. Report RFC3626, IETF*, 2003.

[58] "IEEE Standard for Wireless LAN MEdium Access Control (MAC) and Physical Layer (PHY) Specifications, P802.11."

[59] P.-W. Yau and C. J. Mitchell, "Reputation Methods for Routing Security for Mobile Ad Hoc Networks," *Proceedings of SympoTIC '03 Joint IST Workshop on Mobile Future and Symposium on Trends in Communications*, pp. 130-137, 2003.

[60] S. Buchegger and J.-Y. L. Boudec, "The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks," *Proceedings of WiOpt `03: Modeling and Optimization in Mobile Ad Hoc and Wireless Networks*, 2003.

[61] M. T. Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy, "A Reputation-based mechanism for Isolating Selfish Nodes in Ad Hoc networks," *Proceedings of the IEEE Mobiquitous 2005*, pp. 3-11, 2005.

[62] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure on-demand routing protocol for ad hoc networks," *The 10th Annual International Conference on Mobile Computing and Networking (MobiCom '02)*, pp. 12-23, 2002.

[63] "IEEE Standard for Wireless LAN MEdium Access Control (MAC) and Physical Layer (PHY) Specifications, P802.11,"

[64] M. T. Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy, "A Reputation-based mechanism for Isolating Selfish Nodes in Ad Hoc networks," *In Proc. of IEEE Mobiquitous '05*, pp. 3-11, 2005.

Vita

Mohamed Tamer Refaei was born in Cairo Egypt. He received his B.S in Computer Science form the University of Maryland College Park in 2000 and his M.S in Computer Science from the George Washington University in 2003. He has been the recipient of the NSF IGERT Fellowship 09/2003-07/2006. His research interests are in security, wireless networks, and cognitive radio networks. The following is his current list of publications:

- M. Tamer Refaei, L. Dasilva, M. Eltoweissy, "Reputation Management Systems in Ad hoc Networks," Book Chapter, submitted 2006.
- M. Tamer Refaei, Y. Rong, L. DaSilva, and H. Choi "Detecting Node Misbehavior in Ad hoc Networks," *Proceedings of the IEEE International Conference on Communications (ICC 2007)*, Glasgow, Scotland, to appear.
- J. Hwang, M. Tamer Refaei, H. Choi, J. Kim, J. Sohn, and H. I. Choi "Policy-Based QoS -Aware Packet Scheduling for CDMA 1xEV-DO," *Proceedings of the IEEE International Conference on Communications (ICC 2007)*, Glasgow, Scotland, to appear.
- G. Hadjichristofi, L. DaSilva, A. MacKenzie, S. Midkiff, M. Tamer Refaei, "Monitoring Cooperation: A Reputation-based Mechanism for Isolating Selfish Nodes in Mobile Ad hoc Networks," MOBICOM 2006 demo.
- R. Chen, M. Snow, J. Park, M. Tamer Refaei, M. Eltoweissy, "Defending against Routing Disruption Attacks in Mobile Ad hoc Networks," *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2006),* San Francisco, CA.

- J. Agre, W. Chen, M. Tamer Refaei, A. Sonalker, C. Zhu and X. Yuan, "Secure NOmadic Wireless Mesh (SnowMesh) 802.11 TGs ESS Mesh Networking Proposal," IEEE 802.11, FLA-PCR-TM-23, June, 2005
- N. Aboudagga, M. Tamer Refaei, M. Eltoweissy, L. DaSilva, and J. Quisquater, "Authentication Protocols for Ad hoc Networks: Taxonomy and Research Issues," *Proceedings of the 1st ACM International Workshop on Quality of Service & Security in Wireless and Mobile Networks*, 2005, Montreal Canada, pp. 96 – 104.
- M. Tamer Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy "A Reputation-based mechanism for Isolating Selfish Nodes in Ad hoc networks," *Proceedings of the IEEE Mobiquitous 2005*, San Diego, CA, pp 3 – 11.
- M. Tamer Refaei, V. Srivastava, L. DaSilva, "DEBRA: Delivery Based Reputation mechanism for Ad-hoc networks," *Research Poster, IEEE Sensor and Ad hoc Communications and Networks Conference*, October 2004.