# MOBIWORP: Mitigation of the Wormhole Attack in Mobile Multihop Wireless Networks

**Issa Khalil, Saurabh Bagchi, Ness B. Shroff**
**Dependable Computing Systems Lab (DCSL) & Center for Wireless Systems and Applications (CWSA)**
**School of Electrical & Computer Engineering, Purdue University**
**Email: {ikhalil, sbagchi, shroff}@purdue.edu**

**Corresponding author: Issa Khalil, ikhalil@purdue.edu**

**Contact Info for Corresponding Author: 465 Northwestern Avenue, West Lafayette, Indiana 47907. USA.**

**Phone: 765-494-3362 Fax: 765-494-2706**

## Abstract

In multihop wireless systems, the need for cooperation among nodes to relay each other's packets exposes them to a wide range of security attacks. A particularly devastating attack is the wormhole attack, where a malicious node records control traffic at one location and tunnels it to a colluding node, possibly far away, which replays it locally. This can have an adverse effect on route establishment by preventing nodes from discovering legitimate routes that are more than two hops away. Previous works on tolerating wormhole attacks have focused only on detection and used specialized hardware, such as directional antennas or extremely accurate clocks. More recent work has addressed the problem of locally isolating the malicious nodes. However, all of this work has been done in the context of static networks due to the difficulty of secure neighbor discovery with mobile nodes. The existing work on secure neighbor discovery has limitations in accuracy, resource requirements, and applicability to ad hoc and sensor networks. In this paper, we present a countermeasure for the wormhole attack, called MOBIWORP, which alleviates these drawbacks and efficiently mitigates the wormhole attack in mobile networks. MOBIWORP uses a secure central authority (CA) for global tracking of node positions. Local monitoring is used to detect and isolate malicious nodes locally. Additionally, when sufficient suspicion builds up at the CA, it enforces a global isolation of the malicious node from the whole network. The effect of MOBIWORP on the data traffic and the fidelity of detection is brought out through extensive simulation using ns-2. The results show that as time progresses, the data packet drop ratio goes to zero with MOBIWORP due the capability of MOBIWORP to detect, diagnose and isolate malicious nodes. With an appropriate choice of design parameters, MOBIWORP is shown to completely eliminate framing of a legitimate node by malicious nodes, at the cost of a slight increase in the drop ratio. The results also show that increasing mobility of the nodes degrades the performance of MOBIWORP.

**Keywords**: Mobile ad hoc networks, neighbor watch, wormhole attack, secure neighbor discovery, node isolation.

## 1 Introduction

There is significant interest in the research and development of ad hoc and sensor wireless networks for a variety of emerging applications. These multi-hop wireless networks are especially suited for scenarios where it is infeasible or expensive to deploy significant networking infrastructure. However, the open nature of the wireless communication channels, the lack of infrastructure, and the hostile environments where they may be deployed, make them vulnerable to a wide range of security attacks. These attacks could involve eavesdropping, message tampering, or identity spoofing, which have been addressed by customized cryptographic primitives. Many attacks are targeted directly at the data traffic by dropping all data packets (blackhole attack), selectively dropping data packets (grayhole attack), and performing statistical analysis on the data packets to obtain critical information, such as the location of primary entities in the network. For an attacker to be able to launch damaging data attacks, one option is to have a large number of powerful adversary nodes distributed over the network and possessing cryptographic keys. Alternately, the attacker can achieve such attacks by having a few powerful adversary nodes that need not authenticate themselves to the network (i.e., external nodes). The attacker can achieve this by targeting specific *control traffic* in the network. Typical examples of control traffic are routing, monitoring liveness of a node, topology discovery, and distributed location determination. A particularly severe control attack on the routing functionality of wireless networks, called the *wormhole attack*, has been introduced in the context of ad hoc networks [11][13]-[15]. During the attack, a malicious node captures packets from one location in the network, and "tunnels" them to another malicious node at a distant point, which replays them locally. The tunnel can be established in many different ways, such as through an out-of-band hidden channel (e.g., a wired link), packet encapsulation, or high powered transmission. This tunnel makes the tunneled packet arrive either sooner or with lesser number of hops compared to the packets transmitted over normal multihop routes. This creates the illusion that the two end points of the tunnel are very close to each other. A wormhole tunnel can actually be useful if used for forwarding all the packets. However, in its malicious incarnation, it can be used by the two malicious end points of the tunnel to pass routing traffic to attract routes through them. The malicious end points can then launch a variety of attacks against the data traffic flowing on the wormhole, such as the grayhole attack or statistical flow analysis of the traffic. Also the wormhole attack can affect route establishment by preventing any two nodes in the network that are greater than two hops away from discovering routes to each other. The wormhole attack affects many applications and utilities in ad hoc networks such as, network routing, data aggregation and clustering protocols, and location-based wireless security systems [2]-[12][17][18]. Finally, the wormhole attack is considered particularly insidious since it can be launched without having access to any cryptographic keys or compromising any legitimate node in the network.

Our primary goal in this paper is to provide primitives that mitigate the wormhole attack in *mobile* ad hoc networks. Mitigation involves detection of the attack, diagnosis of the adversary nodes, and nullifying their capability for further damage. Previous approaches to handling the wormhole attack have concentrated on detection using specialized hardware [14], highly accurate time measurement [20], specialized trusted nodes [32] and clock synchronization [13]. However, these may not be feasible for many large scale ad hoc or sensor networks due to the hardware complexity or cost. Also importantly, all of these approaches focus only on detecting and avoiding the attack but do not identify and neutralize malicious nodes. More recent work in a protocol called LITEWORP [15] has provided both detection and local isolation of wormhole nodes. *However, it breaks down in mobile scenarios.* The limitation arises from the inability to securely determine neighbors at arbitrary points in the lifetime of the network. Existing work on secure neighbor discovery cannot be applied to the problem because it hinges on one or more of the following features: (i) the requirement of extremely accurate clocks, (ii) the assumption of no delay in the network apart from propagation delay [16], and (iii) the requirement of directional antennas and measurement of exact angle of reception [14]. The large volume of work on location determination relies on inaccurate measures, such as received signal strength, and is distinct from the problem of *location verification* of a possibly malicious node. A second challenge arises from the possibility of a mobile adversary that may perform malicious actions at one location and move. The LITEWORP protocol only performs local isolation of the adversary and leaves the network open to unbounded amount of damage through the mobile adversary.

In this paper, we make the following contributions:

- We provide a primitive that prevents a node from claiming to exist at more than one position in the network. This primitive can be used in detecting several different attacks such as the Sybil attack ([42] [43]).
- We develop a protocol called MOBIWORP that can detect and diagnose wormhole attacks in mobile networks.
- We provide a technique in MOBIWORP to isolate malicious nodes from the network, thereby removing their ability to cause future damage.

- We analyze the detection latency and overhead of our solution and provide extensive simulations to study the efficacy of our approach.

MOBIWORP uses local monitoring of neighborhood communication by each node as a primitive. It does not require specialized hardware at the network nodes, but instead relies on a secure central authority (*CA*) for position tracking of the mobile nodes and keeping track of adversarial behavior by a mobile node. The use of *CA* appears to fly in the face of the holy design grail of completely distributed protocols. However, the *CA* is contacted only in the event of motion and the protocol can continue to operate through periods when the *CA* is unreachable. To improve scalability and availability, the architecture can accommodate a hierarchical *CA* structure with each *CA* responsible for part of the network.

The detection in MOBIWORP is of two types – *local detection* and *global detection*. In the former, the adversarial node is detected by the guards in its current neighborhood in a distributed fashion. In the latter, the adversary is detected on a global network scale by the *CA* aggregating reports from guards at multiple locations. The first protocol proposed under MOBIWORP is called the *Selfish Move protocol* (SMP). In SMP, the mobile node can generate, send, and receive its own traffic but cannot forward any traffic. This design arises from the insight that a node can only launch a wormhole attack if it can forward packets. However, SMP may cause the network to be disconnected if a large fraction of the nodes are mobile at the same time. This scenario is expected to occur in only the most mobile networks.

To address this case, we develop a second protocol called Connectivity Aided Protocol with Constant Velocity (CAP-CV). This protocol eliminates the aforementioned lack of connectivity problem by allowing the mobile node to also forward packets. However, this protocol comes with some requirement: the node has to file an "approximate flight plan" with the *CA* giving the average velocity between the current and the new position. Note that in the SMP, the node does not need to determine a priori its trajectory from the source to the destination while in the CAP_CV, it does.

MOBIWORP provides a technique that isolates the malicious nodes from the network thereby removing their ability to cause future damage. The isolation is achieved in two phases – locally, whereby the malicious node is removed from the current neighborhood and globally using global information at the *CA* so that a peripatetic mobile node cannot cause unbounded damage in the network. The detection and the isolation process are done judiciously to minimize the possibility of victimizing innocent nodes due to false alarms caused by natural collisions in the wireless medium or deliberate framing by malicious nodes. The simulation results show that, for the network densities we simulate, MOBIWORP can achieve more than 90% local and global isolation of malicious nodes. Moreover, the data packet drop ratio goes to zero with time due to the capability of MOBIWORP to isolate malicious nodes that are involved in packet dropping. For an appropriate choice of design parameters, MOBIWORP can completely eliminate local framing at the cost of slight increase in the data packet drop ratio.

The rest of the paper is organized as follows. Section 2 presents related work in the field of wormhole detection and mitigation. Section 3 lays out the design foundations while 4 describes the protocols for secure location estimation. Section 5 gives the simulation experiments and the results. Section 6.1 presents the analysis for resource overhead, detection latency, and possibility of framing of good node. Section 7 concludes the paper.

## 2 Related Work

The wormhole attack in wireless networks was independently introduced by Dahill [1], Papadimitratos [4], and Hu [13]. Hu *et al.* [13] introduced the concept of geographical and temporal packet leashes for detecting wormholes. The solution requires either that each node has accurate location information and loose clock synchronization (geographical leash) or accurate clock synchronization (temporal leash). An implicit assumption in the approach is that packet processing, sending, and receiving delays are negligible. Both geographical and temporal leashes need to add authentication data to each packet to protect the leash, use a large amount of storage for the Merkle hash tree based authentication scheme [22], and do not isolate malicious nodes. Capkun *et al.* [20] present SECTOR, which can detect wormhole attacks without requiring any clock synchronization but using special hardware for a challenge request-response and for accurate time measurements. Hu and Evans [14] use directional antennas to *prevent* a sub-class of wormhole attacks. They provide a method for secure neighbor discovery using the directionality of the antennas and under the assumption that all the nodes are aligned. The requirement of directional antennas on all nodes may be infeasible for some deployments. Another approach is sending acknowledgement to packets to discover wormholes in the path [19]. This approach introduces overhead of control messages and does not isolate the malicious nodes. Wang *et al.* [31] present a method for graphically visualizing the occurrence of wormholes in *static* sensor networks by reconstructing the lay-out of the sensors using multi-dimensional scaling.

The approach that we propose for detection of wormhole nodes is *local monitoring* whereby nodes oversee part of the traffic going in and out of its neighbor nodes. The idea of overhearing traffic in the vicinity in wireless networks is not new (e.g. [21][24][25][26]). Our novelty lies in applying and extending it for detecting wormhole attacks in *mobile* networks.

A fundamental building block for detecting the wormhole attack in mobile networks is a protocol for *secure* neighbor discovery. Neighbor discovery can be looked upon as a subset of the problem of location determination under the condition that the location of a node can be determined by *other* nodes. Several physical properties of the received signal are used for one hop location estimation – signal strength, time of flight, and angle of arrival [27]. The time of flight approach is similar to the temporal leash and suffers from the same drawbacks. Typically the location determination protocols have an explicit localization phase when beacon messages are exchanged after which each node determines its relative location with respect to its neighbors. However, this is not secure since a powerful adversary can increase its transmission power for just this phase. The plethora of existing protocols for a node to determine its own location (e.g. [28]-[30]), sometimes in the presence of malicious beacon nodes [36], are asymmetric to our problem where the determination has to be done securely by the neighbors of a node.

There are few solutions proposed in the literature for secure neighbor discovery. The approach by Evans [14] uses directional antennas on each node with precise alignment of the nodes. The approach by Perrig [16] is presented in the context of designing a route discovery component that is secure to the rushing attack. The approach relies on the time of flight and thus assumes very accurate time measurement and disregards all sources of delay other than the propagation delay. The MAC delay in networks of even moderate density can make this assumption dubious. Many schemes use beacons sent by powerful nodes to enable location determination by other nodes. Sastry *et al.* [41] tackle the problem of a node securely verifying the location of possibly malicious beacon nodes that send spurious information about their own location. This problem definition is similar to ours, except that we want to verify location of any arbitrary node. Their approach uses a very fast (e.g., radio frequency) and a relatively slow (e.g., ultrasound) signal to derive distance from the time delay. While this kind of capability can be mounted on a limited set of beacon nodes, it is infeasible to do this on all the nodes in the network.

## 3 Design Foundations

### 3.1 System Model and Assumptions

The system comprises a mix of static and mobile nodes with a single level of transmission power and bi-directional links. Each mobile node is capable of determining its destination location before moving and knows its current location. Such location information may be obtained using the Global Positioning System (GPS) [35] or through location discovery algorithms that depend on beacon nodes such as [33][34][40][41]. The network is assumed to be very loosely time-synchronized, in the range of tens of milliseconds. The nodes may or may not be resource constrained, however, MOBIWORP attempts to be parsimonious in its own resource consumption. The network has a trusted central authority (*CA*) and each node has a shared key with the *CA*. The *CA* does not have any resource constraint. Each node in the network can have a symmetric shared key with each other node and is capable of verifying public key certificates issued by the *CA*. Such a shared key may be obtained through one of several possible key management schemes existing in ad hoc wireless networks (e.g., [38]).

The adversary node may be external or internal (i.e. possessing the cryptographic keys) and it may be more resource-rich than a regular node, such as having unlimited energy source, high speed motion, and high powered transmission capability. Recall that the wormhole attack can be launched by external or internal adversary nodes since the adversary does not need to possess any cryptographic key to successfully launch the attack. Multiple adversary nodes may collude. Each node has to authenticate itself to the CA and other nodes that it may communicate with using the shared keys. Therefore, a node cannot assume multiple identities (for details refer to [42]). We assume that there is a maximum limit ($M_{max}$) on the number of internal nodes that an attacker can capture. Such assumptions are commonly made in sensor networks as in [37][38]. The wormhole attack can be launched in one of four modes according to the classification in [15], such as high powered transmission and packet encapsulation. Without loss of generality, the mode that is simulated is the out-of-band high bandwidth channel between the malicious nodes.

### 3.2 Local Monitoring and Node Locations

A collaborative detection strategy is used where a node monitors the control traffic going in and out of its neighbors. This strategy was introduced in [15] for *static sensor* networks and here we give the background needed to follow the protocols presented in that paper for mobile ad hoc and sensor networks.

For a node, say $\alpha$, to be able to watch a node, say $\beta$, $\alpha$ must be a neighbor of both $\beta$ and the previous hop from $\beta$, say $\delta$. Then we call $\alpha$ the guard node for the link from $\delta$ to $\beta$. For example, in Figure 1, nodes *M*, *N*, and *X* are the guard nodes of *A* over the link from *X* to *A*. Information from each packet sent from *X* to *A* is saved in a *watch buffer* at each guard. The guards expect that *A* will forward the packet towards the ultimate destination, unless *A* is itself the destination. Each entry in the watch buffer is time stamped with a time threshold, $\tau$, by which *A* must forward the packet. Each packet forwarded by *A* with *X* as a previous hop is checked for the corresponding information in the watch buffer. The check can be to verify if the packet is fabricated or duplicated (no corresponding entry in the buffer), corrupted (matching hash of the payload), dropped or delayed (entry is not matched within $\tau$).

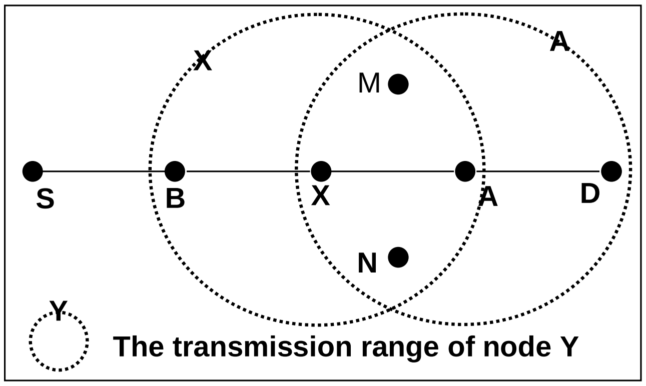

**Figure 1: X, M, N are guards of the link from X to A**

A malicious counter (*MalC(i,j)*) is maintained at each guard node, *i*, for a node, *j*, at the receiving end of each link that *i* is monitoring over a sliding window of length $T_{win}$. *MalC(i,j)* is incremented for any malicious activity of *j* detected by *i*. The increment to *MalC* depends on the nature of the malicious activity, being higher for more severe infractions. When the growth in the counter value maintained by a guard node $\alpha$ for node *A* ($MalC(\alpha,A)$) crosses a threshold rate ($MalC_{th}$) over $T_{win}$, node $\alpha$ revokes *A* from its neighbor list, and sends to each neighbor of *A*, an authenticated (using the shared key) alert message indicating *A* is a suspected malicious node. When a neighbor $d_i$ gets the alert, it verifies the authenticity of the alert message, that $\alpha$ is a first-hop neighbor of *A*, and that *A* is $d_i$'s neighbor. It then stores the identity of $\alpha$ in an alert buffer associated with *A*. When $d_i$ gets enough alert messages about *A*, it marks the status of *A* as revoked in the neighbor list. The notion of enough number of alerts is quantified by the *detection confidence index* $\gamma$. Each node maintains memory of nodes that it has revoked through a local *blacklist* so that a malicious node cannot come back to its neighborhood and claim to be blameless. Each entry in the blacklist consists of two fields—the identity of the malicious node and a one-bit flag to indicate whether this malicious node has been detected directly or through the reception of $\gamma$ or more alerts from other nodes. This constitutes *local isolation* of a malicious node by its current neighbors. Note, however, that a node that has less than $\gamma$ neighbors (such as nodes at corners or in sparse areas) may adapt $\gamma$ locally to be the number of neighbors that it has.

*Framing* is the process by which an innocent node is proved to be malicious by a quorum of malicious nodes. A small value for $\gamma$ increases the chance of successful framing of some good nodes (by collusion among $\gamma$ or more malicious nodes), while a large value of $\gamma$ may increase the rate of harm a malicious node causes in the network before being locally detected. If we set $\gamma$ to be infinity it means that a node only trusts itself in revoking a suspicious node and thus the local framing probability goes to zero. Note that the number of alerts is cumulative over time. A malicious node can not claim more than one identity due to authentication and therefore even with a finite $\gamma$, a single node cannot frame another node.

The *physical location* of the node is the location where the node physically exists. The *logical location* of the node is the location that the node announces to the *CA*. A node $\alpha$ is considered *integrated* at a position (*X*, *Y*) if there exists at least one node within one transmission range of (*X*, *Y*) which considers $\alpha$ to be its first-hop neighbor. If no node at all exists in the vicinity of (*X*, *Y*), then the condition of integration is trivially satisfied. The property guaranteed by MOBIWORP is that a node $\alpha$ can only be integrated in its *logical location*. The physical location and the logical location of a good node are the same but may not be for a malicious node. If a node is integrated at a location, it can send, receive, and forward packets from its neighbors in that location. If a node is not integrated at a location it cannot do that irrespective of its physical location. In this paper, we use location to mean the logical location, unless explicitly stated otherwise.

The determination of first- and second-hop neighbors plays a crucial role in the detection of the wormhole attack using local monitoring. A node does not accept or send packets to a node that is not recognized as a first-hop neighbor. Also, a node acts as a guard depending on its knowledge of one hop neighbors. The second-hop neighbor information is required to detect when a node falsifies information about the immediate sender. In a static scenario, the neighbor list is built once at the time of deployment when the network is assumed adversary-free as in [15]. However, in a mobile scenario, the neighborhood may change during the lifetime of the network and therefore dynamic secure neighbor discovery is required. The problem of neighbor determination is a subset of the problem of verifying the location of each node that lies within two transmission ranges. Hence, verifying the location of a node is the core of our solution and forms the topic of the discussion in the next section.

### 3.3 Wormhole Attack Scenario Mitigation

Two colluding malicious nodes may launch a wormhole attack to involve themselves in a route by simply giving the false illusion that the route through them is the shortest. Consider the scenario in Figure 2. Two colluding nodes, $M_1$ and $M_2$, use an out-of-band channel or packet encapsulation to tunnel routing information between them. When $M_1$ receives the *Route Request (REQ)* initiated by *S*, it tunnels the *REQ* to $M_2$. Node $M_2$, then, broadcasts the *REQ* in its neighborhood. To mitigate the attack, we require as mentioned in Section 3.2, that each node knows its first-hop and second-hop neighbors and that the packet forwarder announces the node from which it has received the packet. Assuming for now that these requirements are satisfied, MOBIWORP can detect the wormhole attack. When $M_2$ receives the *REQ* tunneled by $M_1$, it has two choices for the previous hop — either to append the identity of $M_1$, or to append the identity of one of $M_2$'s neighbors, say *X*. In the first choice all the neighbors of $M_2$ reject the *REQ* because they all know that $M_1$ is not a neighbor of $M_2$. In the second case, all the guards of the link from *X* to $M_2$ (*X*, *N*, and *L*) detect $M_2$ as fabricating the route request since they do not have the information for the corresponding packet from *X* in their watch buffer. Similarly, when $M_1$ receives the *REP* tunneled from $M_2$ it has the same choices as $M_2$ and a similar scheme is used by the guards of the incoming link to $M_1$.

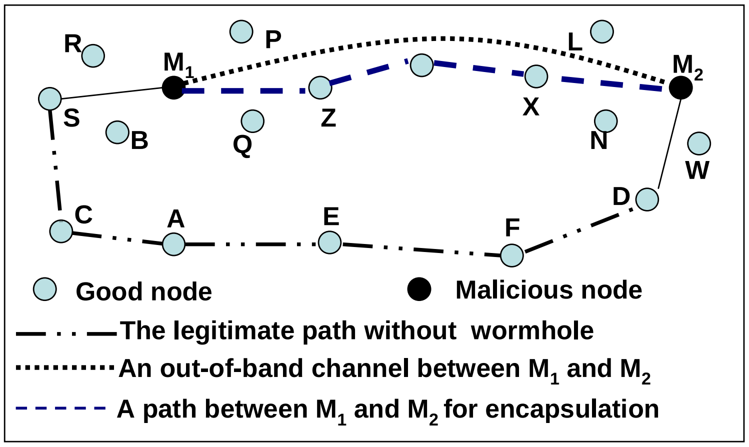
R
P
L
M1
M2
S
Z
X
B
Q
N
W
D
C
A
E
F
Good node
Malicious node
The legitimate path without wormhole
An out-of-band channel between M1 and M2
A path between M1 and M2 for encapsulation

**Figure 2: A wormhole attack scenario**

## 4 Secure Node Integration Protocols

In this section we describe node integration within the network. Node integration includes secure neighbor determination and the determination of the role that a node is allowed to play after being integrated.

### 4.1 Fundamental Structures for Neighbor Determination Protocols

The integration of a node in the network is preceded by an exchange of control messages between the mobile node and the *CA*, called the node-to-*CA* handshake. We introduce the concept of the *Authentication Neighbor Update Message* (**ANUM**), which is akin to a certificate given by the *CA* to a node. The node uses this ANUM to convince other nodes of its *logical* location. The ANUM is signed with the private key of the *CA* and thus can be verified by each node. It carries an expiry time with it, which is the maximum time for which the node can remain integrated in the given location with the current ANUM.

Every node in the network has a structure called *neighbor list*, which is a list of nodes that are within two transmission range distances and the location of each node. The neighbor list is updated as the node moves through the network or new nodes move to its neighborhood. A *monitoring round* of guard node $\alpha$ for the monitored node *i* is defined as the period which starts when they become first-hop neighbors and ends when they no longer remain first-hop neighbors, may be due to the mobility of either $\alpha$ or $i$. The *MalC* counter value at node $\alpha$ for node $i$ is not remembered across monitoring rounds. A node can be revoked from the network either locally (Section 3.2) or globally, when its suspicion goes beyond application defined thresholds. *Local revocation* of a node $\alpha$ means that all the first-hop neighbors of $\alpha$ stop interacting with it. *Global revocation* of $\alpha$ means that $\alpha$ is revoked at the *CA* and therefore it can not perform any network function in any part of the network.

The *CA* maintains a global suspicion table ($ST_{glob}$) which is an $(N+1)\times N$ matrix, where $N$ is the number of nodes in the network. The entry $(i, j)$ has $MalC(i,j)$ and a status field ($S_f$) indicating if node $i$ has locally revoked node $j$. The $(N+1)^{th}$ row has the global opinion of the *CA* about a given node. Thus entry $ST_{glob}[N+1,i]$ has a counter field (*Cntr*) for how many nodes have flagged node $i$ to be malicious and a status field ($S_f$) set to one if $Cntr > M_{max}$. This serves as the trigger for the *CA* to globally revoke node $i$. The *CA* aggregates the *MalC* values of node $\alpha$ about $i$ over multiple monitoring rounds. Therefore, even if $MalC(\alpha,i)$ does not cross the $MalC_{th}$ during any single monitoring round, $MalC(\alpha,i)$ may cross the threshold if aggregated at the *CA* over more than one round.

## 4.2 Selfish Move Protocol (SMP)

In this section, we present SMP in the following stages–how does a node handshake with the *CA*, how it behaves when in motion, and how the node gets integrated with the network in the new position. The fundamental insight that is leveraged here is that a node cannot launch a wormhole if it is not allowed to forward any traffic and therefore, if a node's credentials are unsure, it is safe to allow it only to send and receive its own packets. The overall process flow for SMP is shown in Figure 6.

### 4.2.1 Node-to-CA Handshake

A node $\beta$ at position $(X_0, Y_0)$ tries to obtain an ANUM for position $(X_1, Y_1)$, which may be the same as $(X_0, Y_0)$ using the Node-to-*CA* handshake algorithm presented in Figure 3.

1. When the current ANUM of $\beta$ expires, it sends a message to the *CA* with the time till which $\beta$ expects to stay at the new location $T_{pause}(X_1, Y_1)$. This message is called ANUM Request and it is sent by $\beta$ to the *CA* encrypted using the shared symmetric key.
2. The *CA* checks its database for $\beta$ to verify that $\beta$ has no previous valid ANUM and that $\beta$ has not been previously revoked. If $\beta$ has a previous valid ANUM, the *CA* drops the ANUM Request and the handshaking stops at this point. If $\beta$ has been revoked, the *CA* sends an ANUM Reject signed by the private key of the *CA* back to $\beta$.
3. If the checks in the previous step are negative, the *CA* prepares an ANUM Reply that contains the identity of $\beta$, the expiration time of the ANUM, which is equal to the time when the *CA* replies to the ANUM Request plus $T_{pause}(X_1, Y_1)$, and the new location of $\beta(X_1, Y_1)$. This message is signed by the *CA* and sent back to $\beta$.
4. When $\beta$ receives the ANUM Reply, $\beta$ verifies its integrity through the public key of the *CA*.
5. If the *CA* sends an ANUM Reject to $\beta$, then every node that overhears or forwards the ANUM Reject along its path from the *CA* to $\beta$ adds $\beta$ to its local blacklist after verifying the ANUM Reject.
6. If $\beta$ does not receive any reply within a timeout period, it retries the handshaking for three times. If none of these attempts succeeds, $\beta$ selects a backoff time after which it repeats the process until it succeeds.

**Figure 3: SMP handshake between $\beta$ and the *CA***

Two questions arise: What if $\beta$ cannot renew the ANUM due to unavailability or disconnectedness from the *CA*? How does $\beta$ communicate while moving from one location to another?

The fundamental requirement in both cases is to prevent the node from launching a wormhole. In the SMP, we allow a moving node to send and receive its own traffic but not forward any other traffic. However, we want to limit the time from the expiry of a node's ANUM for which it can even do this. This requirement gives rise to the concept of a *grace period* ($t_{grace}$) from the expiry time of the ANUM. The rationale behind the grace period is to give the *CA* the ability to prevent a malicious node from performing any function in the network permanently. This is guaranteed by requiring the node to go back to the *CA* after the expiration of the grace period to renew its ANUM at which point the *CA* can reject the request.

Based on ANUM status, a node can be in one of the four states presented in Figure 4. Recollect that an ANUM has an associated position and expiry time. Figure 5 shows the state transition diagram between these states. It is important for the neighbors of a node $\alpha$ ($NB_\alpha$) to determine its state, so that each member of $NB_\alpha$ can make decisions about the packets to forward to or from $\alpha$. A member of $NB_\alpha$ can determine the valid and incorrect states of $\alpha$ unambiguously but cannot generally differentiate between invalid and revoked states. However, if a member of $NB_\alpha$ hears the ANUM Reject for $\alpha$, it concludes that $\alpha$ is revoked.

***Valid:*** The current position is the same as the one mentioned in the ANUM and the ANUM is not expired. In this state, the node can send, receive, and forward packets, i.e. full network functionality.
***Incorrect:*** The current position is different from the one mentioned in the ANUM (*Incorrect Remote*), the ANUM is expired but within the grace period (*Incorrect Expired*), or both. In this state, the node can only send and receive its own packets.
***Invalid:*** The ANUM is expired beyond the grace period. In this state, the node cannot send, receive, or forward any packet except the handshaking packets with the *CA*.
***Revoked:*** The node has been globally revoked from the network. In this state, the node is completely cut off from the network.

**Figure 4: Node states based on the ANUM status**

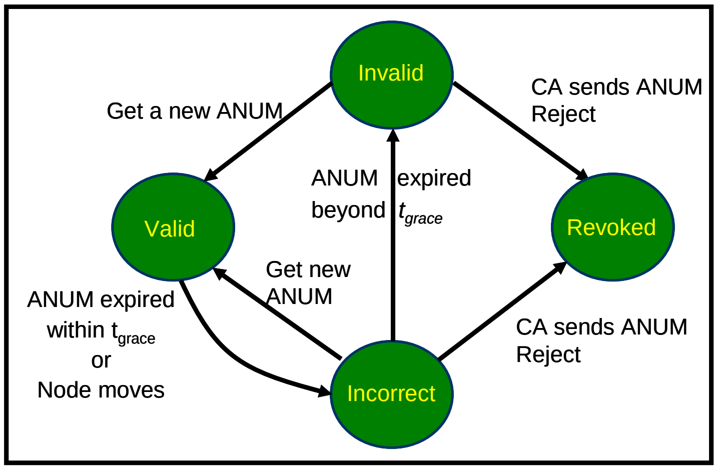

**Figure 5: State transition diagram of node's states**

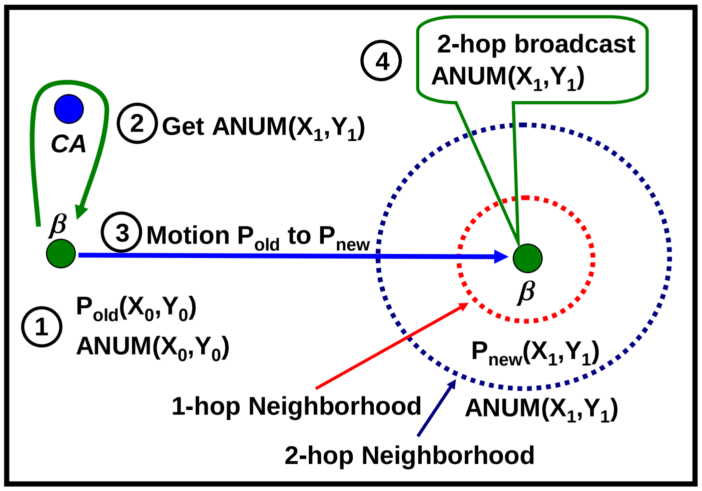

**Figure 6: Schematic of SMP for movement of node $\beta$**

### 4.2.2 Secure Neighbor Discovery and Node Integration Algorithm

After getting, and verifying the ANUM, $\beta$ comes to the *valid state* and uses the ANUM to get integrated at the location associated with ANUM through the algorithm presented in Figure 7. A node $\beta$ in the *incorrect state* carrying an ANUM with position ($X_0$, $Y_0$), that is currently at ($X_1$, $Y_1$) likely due to the fact that $\beta$ is moving to ($X_0$, $Y_0$), integrates with the network using the same algorithm presented in Figure 7 with two changes. In the first step of the algorithm, $\beta$ attaches its current location ($X_1$, $Y_1$) with the ANUM broadcast and in the third step, a neighbor $\alpha$ marks in its neighbor list entry that $\beta$ can only send and receive its own traffic. A node in the *invalid state* can not integrate in the network until it gets an ANUM through the node-to-*CA* handshake algorithm, Section 4.2.1. Finally, a node in the *revoked state* cannot get an ANUM nor can it integrate in any part of the network.

1. Node $\beta$ sends a two-hop broadcast of its ANUM, ANUM Discover, seeking to discover neighboring nodes.
2. A neighbor $\alpha$ that receives the ANUM Discover, verifies the signature of the *CA* and if its expiry time is in the future. Recollect that the clocks of the different nodes are loosely synchronized.
3. Node $\alpha$ computes the distance to $\beta$, adds $\beta$ to its first-hop or second-hop neighbor list based on the computed distance between the position in the ANUM and its own position. Then it stores $\beta$'s location and the expiration time of its ANUM.
4. Node $\alpha$ then sends its own ANUM to $\beta$. Along with this, node $\alpha$ sends its local blacklist to $\beta$ authenticated using the shared key.
5. Node $\beta$ verifies the ANUM of node $\alpha$ using the signature of the *CA* and its expiry time, updates its neighbor list to include node $\alpha$ based on the computed distance between the position in the ANUM of $\alpha$ and its own position, and stores the blacklist of $\alpha$.
6. After $\beta$ discovers its first-hop neighbors, it sends an authenticated one-hop broadcast of its blacklist to them. This broadcast is authenticated individually using the shared key between $\beta$ and each first-hop neighbor.
7. Each malicious node in the blacklist of $\beta$ (similarly, $\alpha$) that is directly detected by $\beta$ (similarly, $\alpha$) serves as an alert of malicious detection to the first-hop neighbors of $\beta$ (similarly, to $\beta$).
8. When the ANUM of $\beta$ expires, node $\alpha$ removes $\beta$ from its neighbor list, and vice-versa.

**Figure 7: Node integration by $\beta$ in valid state**

## 4.3 Connectivity Aided Protocol with Constant Velocity (CAP-CV)

SMP suffers from two shortcomings. In network scenarios with a high number of concurrently moving nodes, a large fraction of the nodes is disallowed from forwarding packets, thereby disconnecting the network. A second problem is that SMP prevents a node which needs communication while moving, from doing so after the grace period. Therefore, we provide in this section a protocol called CAP-CV that preserves the same connectivity conditions of the mobile network and allows the moving nodes to travel any distance. However, we require the mobile nodes to declare to the *CA* the average velocity with which it will move to ($X_1$, $Y_1$).

1. Node $\beta$ sends an ANUM Request to the *CA* with ($X_0$, $Y_0$), ($X_1$, $Y_1$), the start time of motion $T_{move}$, and an anticipated velocity *v*.
2. Identical to step 2 of the SMP node-to-CA handshake protocol.
3. If the checks in the previous step are negative, the *CA* sends a signed ANUM Reply to $\beta$ that contains the identity of $\beta$, ($X_0$, $Y_0$), ($X_1$, $Y_1$), the moving start time $T_{move}$, *v,* and the expiration time of the ANUM which is equal to the anticipated arrival time of $\beta$ at ($X_1$, $Y_1$). This message is signed using the private key of the *CA*.
4. When $\beta$ receives the ANUM Reply, it verifies its integrity. Then $\beta$ can use the ANUM to discover the neighbors and prove its existence while moving to ($X_1$, $Y_1$).
5. & 6. Identical to steps 5 & 6 of the SMP node-to-*CA* handshake algorithm, Figure 3, where the *CA* generates an ANUM Reject.

**Figure 8: CAP-CV handshake between $\beta$ and the *CA***

The node is allowed to vary its promised velocity ($v$), as long as the difference between the actual position and the expected position is less than a threshold value $D_{th}$. The value $D_{th}$ should be high enough to account for the inaccuracy of location determination systems such as GPS. However, the tradeoff is that a high value of $D_{th}$ reduces the possibility of detecting a malicious node that intentionally lies about its physical location (for detection procedure see Section 0). Alternately, the node may declare to the CA its entire trajectory from the source to the destination. The node to *CA* handshake that happens in CAP-CV is presented in Figure 8.

The protocol to integrate node $\beta$ at location ($X_i$, $Y_i$) on the moving path is the same as the one described in Figure 3 of the SMP with two important changes. In step one, before $\beta$ can broadcast its ANUM to discover the neighborhood, it checks whether the anticipated position (computed using velocity $v$) and its actual position are different by more than $D_{th}$. If it is, $\beta$ refrains from communication and does not proceed in the integration because it may be accused as malicious by some other nodes. Otherwise, $\beta$ proceeds in the integration process. Also, in step three, when a node $\alpha$ determines $\beta$ to be its neighbor, it assigns an expiry timer to $\beta$'s entry which depends on when the distance between them gets larger than twice the transmission range. This in turn depends the velocities of $\alpha$ and $\beta$, as given in their ANUMs.

## 4.4 Two Specific Attacks

**False location information**. MOBIWORP enables a new malicious behavior called *location deviation* in which a malicious node lies about its location by presenting a logical location that differs from its physical location. This kind of malicious behavior cannot help the malicious nodes to establish wormholes since our protocols for node integration guarantee that any node can be integrated in the network with forwarding capability while in the valid state only. This can only happen at exactly one location at any time. However, this malicious activity can be detected without incurring any additional overhead. Recall that in node integration, Section 4.2.2, a node *M* broadcasts its ANUM two hops away and the ANUM carries the location of *M*. A node $\alpha$ that receives the ANUM of *M*, computes the distance between itself and *M*. If the distance is greater than the transmission range by more than $D_{th}$, $\alpha$ concludes that *M* is malicious – either transmitting at a higher transmission power or has a physical location different from its logical location.

**DoS against MOBIWORP**. MOBIWORP is a self-healing protocol in that if an intermediate node tries to launch a denial of service attack by dropping ANUM packets, it can be detected by local monitoring since the traffic is part of control traffic. A node cannot exhaust resources of a neighbor by sending false ANUM broadcasts or ANUM Requests since they can be detected respectively by a neighbor and the *CA*. This reasoning relies on the assumption that the node cannot assume multiple identities, which is provided by any protocol that mitigates the Sybil attack [42].

## 4.5 Isolating a Malicious Node

When a node is determined to be malicious, it is important to take some action to neutralize the ability of the node to cause further damage. This aspect is not addressed by any of the previous work on wormhole detection except LITEWORP for static networks. The process of local revocation described in Section 3.2 is quick and lightweight, and has the desired effect of removing the potential for mischief of static malicious nodes. However, a mobile malicious node can move to a new location and perform some malicious activities before it is detected. Hence, MOBIWORP uses the *CA*'s capability to limit the potential for damage by a mobile adversary node. When a guard directly detects a malicious node, it sends an alert packet to inform the *CA* of the identity of the malicious node. The *CA* collects alerts for a node from all the guards that can detect the malicious behavior of the monitored node. When the number of alerts for a certain node exceeds a threshold, $M_{max}$, the *CA* globally revokes the node by preventing it from getting any ANUM in the future. This eventually results in isolating the malicious node from the whole network. The global isolation protocol is shown in Figure 9.

## 5 Simulation Results

We use the *ns-2* simulation environment [23] to simulate a random any-to-any data exchange protocol, in the baseline case without any protection and with MOBIWORP. We initially distribute a given number of nodes randomly over a square field of constant dimensions, 1500 m $\times$ 1500 m. Thus the density increases with the number of nodes. The mobile nodes move according to the random waypoint model with velocity chosen from the uniform distribution ($v_{min}$, $v_{max}$).The *CA* is placed randomly at a certain location in the deployment field and it may be disconnected from some nodes at certain times during the network operation due to mobility.

We use a generic on-demand shortest path routing protocol that floods route requests and unicasts route replies in the reverse direction. A route, once established, is not used forever but is evicted from the cache after a timeout period expires ($TOut_{Route}$). A wormhole is established through an out-of-band channel simulated by allowing the malicious nodes to exchange control packets among themselves instantaneously. After a wormhole is established, the malicious nodes at each end of the wormhole drop all the packets forwarded to them. Each node acts as a data source and generates data using an exponential random distribution with inter-arrival rate of $\mu$. The destination is chosen at random and is changed using an exponential random distribution with rate $\xi$. The input parameters with the experimental values are given in Table 1. As in the protocol description, $m$ is the number of malicious nodes, $M_{max}$ the maximum number of malicious nodes in the network, $\gamma$ the detection confidence, and $N$ the total number of nodes. The simulation accounts for losses due to natural collisions, unreachable destinations, and route breaks due to mobility. The output parameters that we present here are obtained by averaging over 30 runs. For each run, the malicious nodes are chosen randomly, introduced at a random time from the start of the simulation picked from a uniform random distribution (0s, 100s). The total simulation time is 1500s and unless otherwise specified, each output parameter is measured at the end of the simulation time.

1. When node $\alpha$ detects node *M* to be malicious through local monitoring, it sends an alert message to the *CA* with the identity of node *M* signed using the shared symmetric key.
2. When the *CA* receives the alert message from $\alpha$, it updates the data structure described in Section 4.1 to reflect that node $\alpha$ has revoked node *M*, i.e., it sets the entry $ST_{glob}$ *[*$\alpha$*,M].Sf* to one. Node $\alpha$ can inform the *CA* of its *MalC* value for node *M* when the monitoring round of $\alpha$ for *M* ends. Node $\alpha$ piggybacks the counter values it has for its neighbors with its ANUM Request. The *CA* performs aggregation of *MalC(*$\alpha$*, M)* across monitoring rounds and if it determines *M* to be malicious, it sets the entry $ST_{glob}$ *[*$\alpha$*,M).Sf* to one.
3. If any counter value, say for node *M*, crosses the threshold $MalC_{th}$, the *CA* increments $ST_{glob}$*[ N+1,M].Cntr* by one. If $ST_{glob}$*[N+1,M].Cntr* exceeds $M_{max}$, the *CA* globally revokes *M* by setting $ST_{glob}$*[N+1,M].Sf* to one. This means that node *M* can never receive a valid ANUM from the *CA* in the future.

**Figure 9: Global isolation algorithm**

**Table 1: Simulation's Input parameter values**

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Tx Range (*r*) | 250 m | $TOut_{Route}$ | 50 s |
| Avg. # of neighbors | 4-9 | # of nodes (*N*) | 50-100 |
| Channel BW | 2Mbps | $\mu$ | 0.2 s |
| $(v_{min}, v_{max})$ | (10,30) | $\xi$ | 0.02 s |

**Temporal behavior of drop ratio**. In this experiment, we calculate the percentage of data packets dropped with simulation time for both the baseline and the MOBIWORP case. The drop ratio is calculated as (# data packets received at the destination-# data packets sent from the source)/# data packets sent from the source. From Figure 10, it is seen that the drop ratio is lower with MOBIWORP and that the values tend to zero with increasing time, while with the baseline a steady state is reached and the percentage stabilizes. With MOBIWORP the malicious nodes are identified and isolated, however, some cached routes through these malicious nodes continue to be used and hence the percentage of dropped packets does not immediately go to zero on isolation of all wormhole nodes. The higher the number of nodes, the smaller is the fraction of malicious nodes and therefore the lower the percentage of dropped packets.

We also compare the percentage of drop ratio as a function of time for two different values of $\gamma$. The results (figure not shown) show the same trend as in Figure 10, with drop ratio increasing slightly for $\gamma=\infty$. This indicates that for the particular network density, all the guards see nearly the same view of the monitored node and therefore, the difference in time between a guard detecting the event itself and being told by other guards is small. Importantly, the benefit of eliminating all framing ($\gamma=\infty$) comes at a relatively low cost of increase in drop ratio.

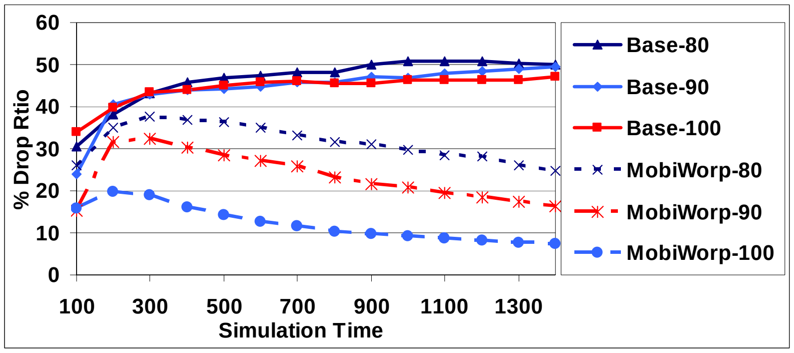

**Figure 10: % data drop ratio ($\gamma$=$M_{max}$=3, m=4)**

**Effect of $\gamma$ on local properties**. The detection confidence ($\gamma$) is varied. Percentage of local isolation is the number of malicious nodes locally isolated to the total number of malicious nodes, while percentage of local false isolation is the number of nodes falsely isolated locally by the total number of good nodes. False detection happens when a good node is mistakenly flagged as malicious due to natural collisions. Consider any two randomly selected neighbor nodes, *S* and *D*, as shown in Figure 11(a). As shown in Figure 11(b), a guard *G* will not detect a fabricated packet sent by *D*, claiming it was received from *S*, if *G* experienced a collision at the time that *D* transmits. A false alarm occurs when *D* receives a packet sent from *S*, while *G* does not receive that packet, and later, *G* receives the corresponding packet forwarded by *D*.

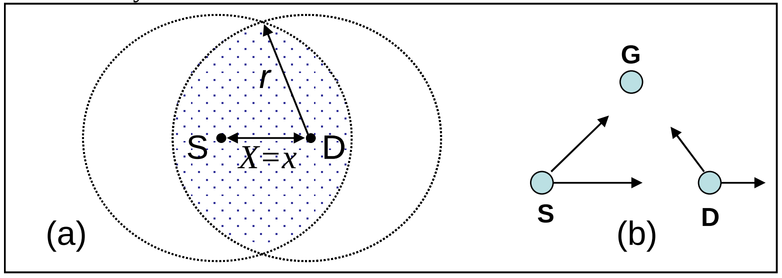

**Figure 11: (a) Area from which a node can guard the link S–D (b) Illustration for detection accuracy**

In Figure 12, we see that with increasing $\gamma$, the percentage of local isolation becomes lower since it becomes more difficult to get agreement on malicious behavior from at least $\gamma$ guards. However, the percentage of local false isolation also decreases since it becomes less likely that $\gamma$ nodes will incorrectly assume malicious behavior due to collisions.

Figure 14 shows the local isolation time, which is the time interval between when a malicious node starts attack at a neighborhood to when it is locally revoked by all its first-hop neighbors. Expectedly, with increasing $\gamma$, the isolation time increases because it takes longer to get an agreement of $\gamma$ the guard nodes.

**Effect of $\gamma$ and $M_{max}$ on global properties.** In this experiment we evaluate the effect of changing $\gamma$ and $M_{max}$ on the global isolation coverage and global isolation time. For a fixed high value of $M_{max}$ (25% of *N*), the global isolation (Figure 13) is very low for low values of $\gamma$. This is because only the guards that directly detect the malicious node report to the *CA*. With a low $\gamma$, most nodes take the opinion of the few who have detected the malicious node through their own observation. Thus, the contribution of each neighborhood in the global isolation is small and the malicious node has to move and be detected at many neighborhoods before being globally isolated. As $\gamma$ increases the global isolation percentage increases since fewer neighborhoods are enough to reach $M_{max}$. The global false isolation is always zero since it is highly unlikely that greater than $M_{max}$ nodes mistakenly accuse a good node due to natural collisions.

Figure 14 shows that the global isolation latency decreases with increasing $\gamma$. Even though the local isolation latency increases as $\gamma$ increases, the global latency decreases due to more number of alerts from each neighborhood and the latter effect dominates.

Figure 15 shows the trend of global isolation coverage as $M_{max}$ increases with infinite $\gamma$. As $M_{max}$ increases, it becomes harder to get an agreement from $M_{max}$ guards about any node which decreases the global isolation and the global false isolation. The figure also shows that the global parameters are insensitive to network density as the results for the 60-node and the 100-node network are close.

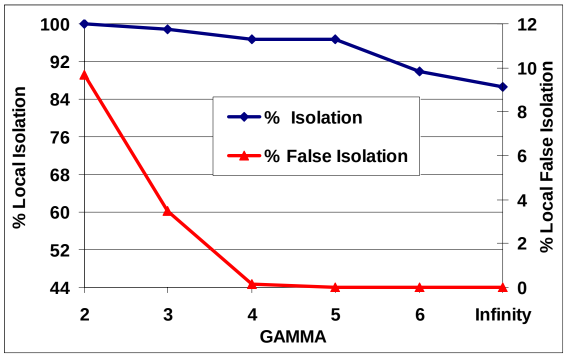

**Figure 12: Local & false isolation** (*m*=4, *N*=60)

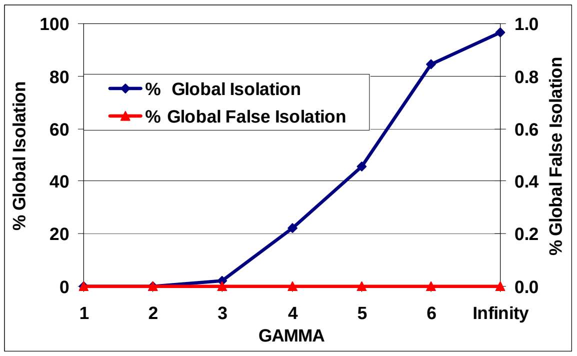

**Figure 13: Global isolation** (m=4, N=60,$M_{max}$=15)

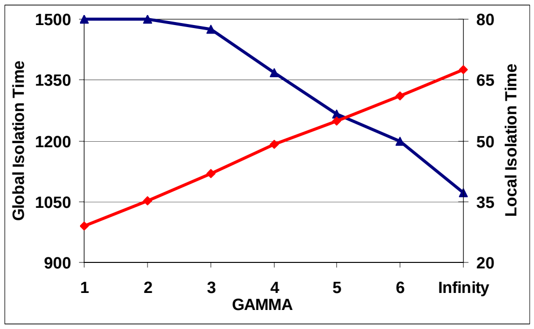

**Figure 14: Isolation time** (m=4, N=60,$M_{max}$=15)

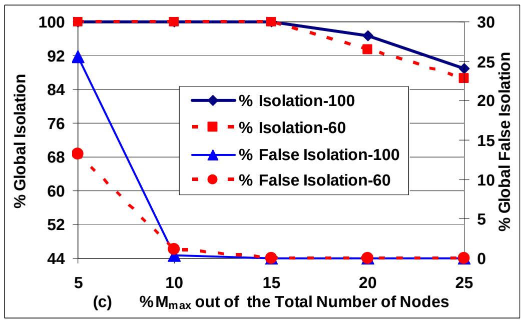

**Figure 15: Global & false isolation** (m=4,$\gamma$=∞)

**Scalability of MOBIWORP**. The next set of experiments brings out the scalability of MOBIWORP with increasing number of nodes. As the number of nodes increases, the density in the network increases leading to increased collisions and thus increasing false isolation (for the same value of $\gamma$ and $M_{max}$, the global and local parameters are almost the same), Figure 16. The percentage of isolation, however, increases due to an increase in the number of guard nodes. The increase in isolation percentage is not high since the minimum neighbor density is greater than $\gamma$, therefore, there is always sufficient number of guards (in average) in all scenarios. However, if we continue increasing *N*, we expect the isolation probability to eventually decrease due to collisions.

**Effect of variation of the number of malicious nodes**. In this set of experiments we bring out the effect of changing the number of malicious nodes on the baseline and the MOBIWORP cases. Figure 17 shows that the percentage of isolation increases with increasing the number of nodes reaffirming the conclusions from Figure 16. The isolation percentage is high even with 6 malicious nodes in the network with perfect capability for collusion, 90% for 80 nodes. The figure also shows relatively constant trend with the number of malicious nodes due to the uniform distribution of the malicious nodes in the simulation.

The trend in false isolation is found to be almost constant with *m* (figure not shown), which is a desirable trend. The trend of isolation time with the number of malicious nodes (figure not shown) is relatively constant since the malicious nodes are likely far apart in the network and the isolation process for the multiple nodes is independent.

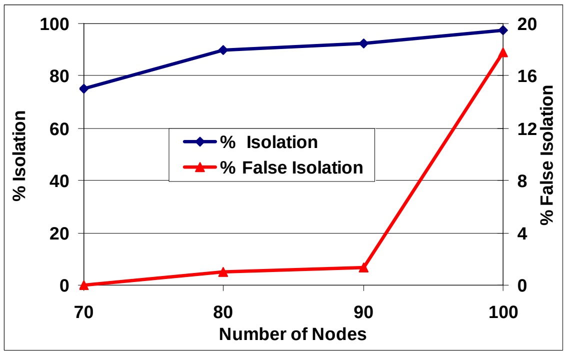

**Figure 16: Scalability of MOBIWORP** ($\gamma$=$M_{max}$=3, m=4)

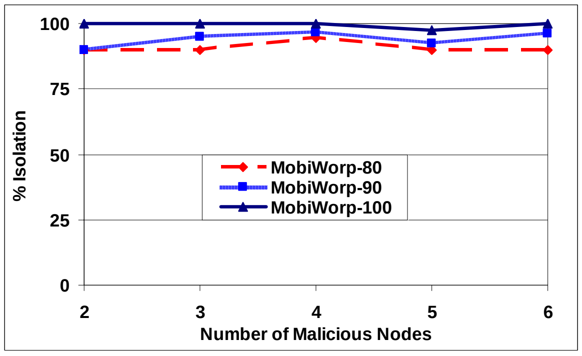

**Figure 17: % Isolation of MOBIWORP** ($\gamma$=$M_{max}$=3)

**Effect of motion**. The duty cycle of motion is defined as the ratio between the time a node spends moving to the total simulation time and is varied by varying the pause time. From Figure 18, it is seen that the percentage of isolation decreases with the increase in the frequency of motion. When a node moves frequently, it often moves before $T_{win}$, i.e. the *MalC* value at a guard is not checked. The *CA* does not aggregate across different guards, i.e. guards at the old location and those at the new location if there is no overlap between them. This causes the isolation coverage to decrease as well as the drop ratio to increase. The percentage of false isolation also decreases because $MalC_{th}$ is not crossed by the time the node moves. In Figure 19, the decrease in the drop ratio in the baseline case is due to the fact that frequent motion causes the wormhole routes to get broken.

## 6 MOBIWORP Analysis

In this section we analysis the ANUM communication overhead, the resource consumption overhead, the detection latency, and the possibility of framing in MOBIWORP. The analysis show that MOBIWORP can operate in resource constrained settings. Also, the analysis of the probability of a good node being framed locally can be set to zero by setting γ to infinity and the possibility of a good node being framed globally can be set close to zero by choosing increasing the value of $M_{max}$.

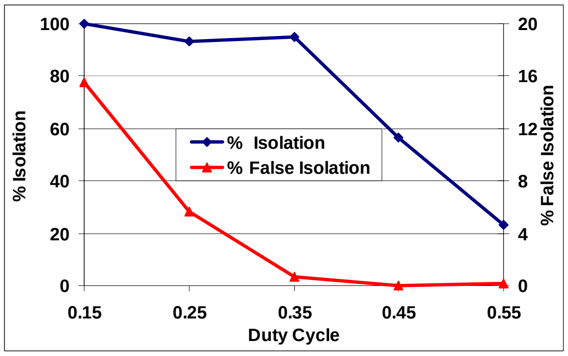

**Figure 18: % of Isolation** ($\gamma=M_{max}$=3, m=4, N=60)

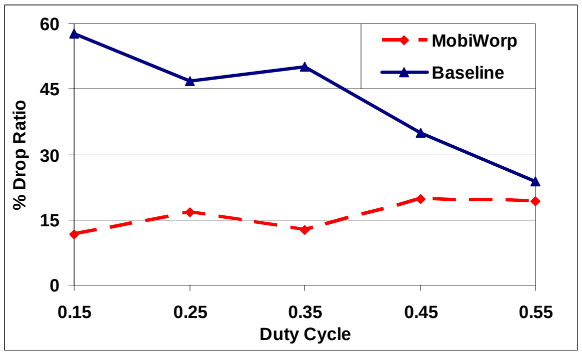

**Figure 19: Comparative performance of baseline and MOBIWORP** ($\gamma=M_{max}$=3, m=4, N=60)

### 6.1 Overhead of ANUM Broadcast

Here we derive an upper bound on the number of ANUM broadcasts if a node ($\alpha$) needs continuous communication while it is moving from its current location $P_0$ to a new location $P_1$ using SMP protocol. Assume that the traveled distance is *X* and one node is enough for $\alpha$ to be connected to the network. Assuming the nodes that are static while $\alpha$ is moving, are uniformly distributed with density *d*.

The shaded area, *Area(X)*, represents the area of common neighbors of $\alpha$ at $P_0$ and $P_1$. If the number of neighbors in *Area(X)* is greater than zero, then $\alpha$ does not need to rebroadcast the ANUM at $P_1$. We need to calculate the value of the maximum traveled distance *X* (call it *x*), such that the probability that there is at least one node in *Area(X)* is greater than some threshold $R_{th}$. Due to our assumption of uniform distribution of nodes in the sensor field, the number of nodes in the shaded area follows a Poisson distribution with rate *Area(X).d*, where

$$Area(X) = 2r^2 \cos^{-1}\left(\frac{X}{2r}\right) - X\sqrt{r^2 - \frac{X^2}{4}} \tag{1}$$

The number of neighbors of a node is $N_b = \pi r^2 \cdot d$ . Therefore, $\alpha$ needs to rebroadcast its ANUM every *R*/*x* distance, where *x* is the maximum value of *X* that satisfies the following inequality,

$$1 - e^{-Area(X)\cdot d} \geq R_{th} \Rightarrow Area(X) \leq -\frac{1}{d}\ln(1 - R_{th}) \tag{2}$$

The upper bound on the traveled distance ($x_0$) as a function of the number of neighbors ($N_b$) is shown in Figure 21. The figure shows that the maximum distance before a required ANUM broadcast, to maintain connectivity using SMP while moving, increases with the network density but the increase slows down. It shows that with densities of 20 neighbors and above, the traveled distance is more than the transmission range.

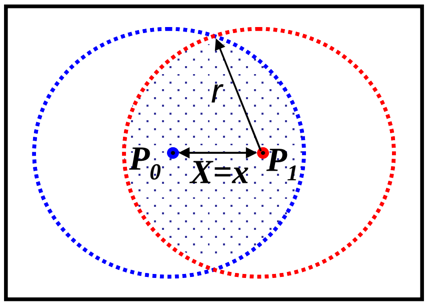

**Figure 20: A node travels from $P_0$ to $P_1$**

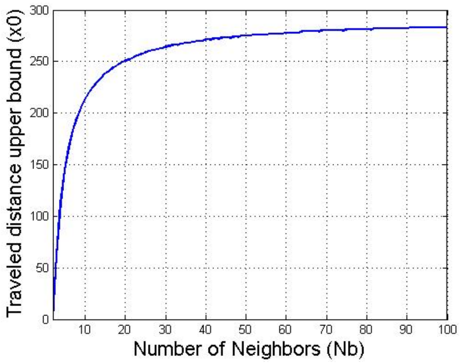

**Figure 21: Traveled distance upper bound before ANUM broadcast in SMP**

## 6.2 Latency Analysis of MobiWorp

Here, we analyze the amount of time it takes to detect a malicious node. Assume the traffic distribution and the bandwidth capacity allows a maximum of $\mu$ packets to be forwarded by a malicious node $M$ within a time window $T_{win}$. Assume that $M$ selectively fabricates (to evade detection) packets with probability $P_{fab}$. Let $g$ be the guard node of $M$ over the link $X \rightarrow M$ that collects and keeps a malicious counter (*MalC(g,M)*) for $M$ over a window of length $T_{win}$ which slides by $\delta$ units. Assume the *MalC* threshold (*MalC$_{th}$*) over $T_{win}$ is $\beta$ and each malicious activity increases the *MalC* by one. Let $T_{win}/\delta = \eta$.

**Independent case**

When $\eta$=1, the sliding windows are non-overlapping and therefore, the events detected in any two windows are independent. The probability that $g$ detects $M$ during a certain time window ($P_{gdM}$) equals the probability that $M$ fabricates at least $\beta$ packets within $T_{win}$, which is given by

$$P_{gdM} = \sum_{i=\beta}^{\mu} \binom{\mu}{i} (1 - P_{fab})^{\mu - i} P_{fab}{}^{i} \tag{3}$$

The expected time of detection is calculated from the number of $T_{win}$ time slots ($N_{ts}$) that pass before the guard $g$ detects the malicious node $M$. The probability that $N_{ts} = k$ is

$$P(N_{ts} = k) = P_{gdM} (1 - P_{gdM})^{k-1} \tag{4}$$

Using Bernoulli trials, the expected value for $N_{ts}$ is given by $E[N_{ts}] = 1/P_{gdM}$. The expected number of time slots ($E[N_{ts}]$) before a single guard detects a malicious node is plotted in Figure 22. The plot shows that the latency decreases very fast with increasing probability of malicious behavior.

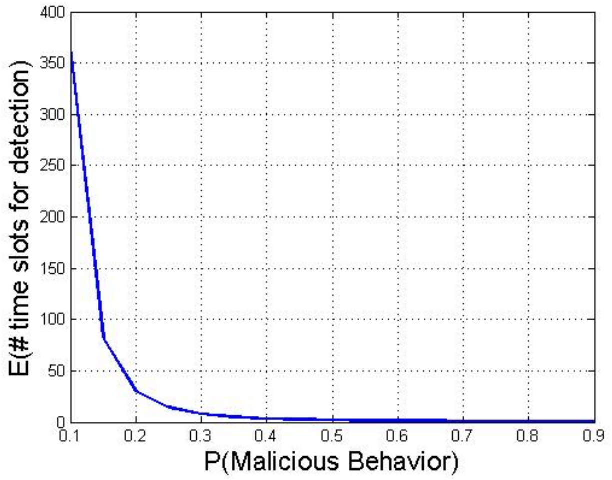

**Figure 22: Expected number of time slots $E[N_{ts}]$ before a single guard detects a malicious node**

**Non-independent case**

For the case with overlapping sliding windows (η>1), the analysis becomes more difficult and we use Martingale Theory [39] to obtain bounds on the delay. Here, we assume a rate based detection, i.e., a node is determined to be malicious if the rate of malicious activities goes above a threshold $\alpha$. We present this analysis for $\gamma = \infty$ since it eliminates framing and is shown to give reasonable detection rates as shown through the simulations (Section 5).

Let $X_i$ be an i.i.d. Bernoulli random variable tracking the number of malicious actions by a node, such that $X_i$=1 (malicious activity) with probability $\lambda$ and zero, otherwise. Thus, $E[X_i] = \lambda$. Consider that the guard observes the node for $N_{act}$ activities (packet forwarding actions).

If we define $Z_{N_{act}}$ as

$$Z_{N_{act}} = \sum_{i=1}^{N_{act}} X_i - N_{act}\lambda \tag{5}$$

Then it can be easily shown that $Z_{N_{act}}$ is a martingale with mean zero. Similarly, $Y_{N_{act}}$ as defined below, is also a martingale with mean zero

$$Y_{N_{act}} = \sum_{i=2}^{N_{act}} X_i - (N_{act}-1)\lambda \tag{6}$$

Now let $N_0$ be the number of activities at which the guard detects the node to be malicious. Then, we define $N_0$ as

$$N_0 = \min_{N_{act}} \sum_{i=1}^{N_{act}} X_i \geq \alpha N_{act} \tag{7}$$

Our goal is to find $E[N_0]$. We can write the following recursion

$$E[N_0] \ = \ E[N_0 \mid N_0 = 1]P(N_0 = 1) \ + \ E[N_0 \mid N_0 > 1]P(N_0 > 1) \tag{8}$$

Note that,

$$E[N_0 \mid N_0 = 1]P(N_0 = 1) \ = \ 1 \times \lambda \ \ = \lambda \tag{9}$$

Also

$$P(N_0 > 1) \ = \ P(X_1 = 0) \ = \ 1\text{-}\lambda \tag{10}$$

Next we find $E[N_0|N_0>1]$. From the Optional Stopping Theorem [39], $E[Y_{N_0}] = E[Y_2] = 0$. Also note that given $N_0$>1,

$$N_0 = \min_{N_{act}} \sum_{i=2}^{N_{act}} X_i \geq \alpha N_{act} \tag{11}$$

This means that given $N_0$>1,

$$N_0 = \min_{N_{act}} Y_{N_{act}} + (N_{act}-1)\lambda \geq \alpha N_{act} \tag{12}$$

In other words, $Y_{N_0} \geq (\alpha\text{-}\lambda)N_0 + \lambda$. Taking expectations on both sides:

$$E[Y_{N_0} \mid N_0 > 1] \ \geq (\alpha\text{-}\lambda)\cdot E[N0] + \lambda \Rightarrow 0 \geq (\alpha\text{-}\lambda)\cdot E[N_0] + \lambda \Rightarrow E[N_0 \mid N_0 > 1] \geq \lambda/(\lambda - \alpha) \tag{13}$$

Thus, de-conditioning, we get the lower bound as

$$E[N_0] = \lambda + \frac{(1-\lambda)\cdot\lambda}{\lambda - \alpha} \tag{14}$$

For the upper bound we can repeat the arguments. Therefore, define

$$Z_{N_0} + N_0 \cdot \lambda \ < \ \ \alpha \cdot N_0 \ + \ 1 \tag{15}$$

The last term is because $X_i \leq 1$. Now, choosing $\alpha$ such that, $\lambda < \alpha$ (i.e., the rate of malicious activity is less than the detection threshold) and taking expectations

$$0 + E[N_0]\cdot(\lambda - \alpha) < 1 \Rightarrow E[N_0] < 1/(\lambda - \alpha) \tag{16}$$

Therefore, the bounds for the expected number of activities after which the guard will detect the node as malicious is

$$\lambda + \frac{\lambda\cdot(1-\lambda)}{\lambda - \alpha} < E[N_0] < \frac{1}{\lambda - \alpha} \tag{17}$$

We plot this equation in Figure 23 and find that the bounds asymptotically converge and exist only for $\lambda > \alpha$.

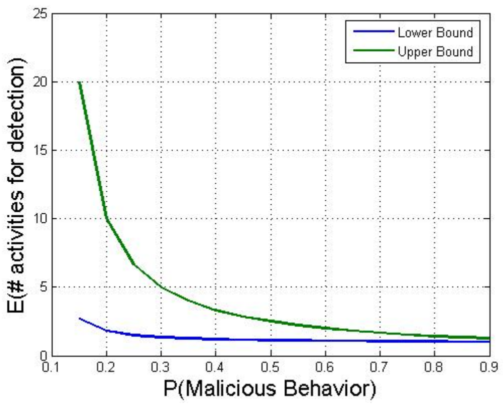

**Figure 23: Lower and upper bound for expected # of activities before a malicious node is detected by a guard**

## 6.3 Overhead Analysis of MOBIWORP

In this subsection, we analyze the memory, the computation, and the bandwidth overheads of MOBIWORP in order to estimate the resource needs it puts on the mobile network. This can lead to the determination of the suitability of the protocol to resource constrained networks, such as sensor networks. An important metric to analyze is the coverage – probability of isolation and probability of false isolation. Since the basic detection mechanism in MOBIWORP is identical to that in LITEWORP, the overhead for static node follows the one in [15] Section 5.1 and is not repeated here. Also the *CA* is considered powerful enough and for the sake of space we are not presenting its overhead analysis here.

**Memory Overhead**: Every node in the network needs to store the first and the second hop neighbor list, the watch buffer [15], the alert buffer, and the black list. If the identity of a node in the network is 2 Bytes, the size of neighbor list is $NB_L = \pi(2r)^2 d$ entries, where $r$ is the communication range and $d$ is the average node density. Each entry in the $NB_L$ requires 9 Bytes; 2 for identity of the neighbor, 1 for the *MalC* associated with that neighbor, 4 for the x-y coordinate of the neighbor, and 2 for the expiration time of the entry. So the total $NB_L$ storage, $NB_{LS} = 9\pi(2r)^2 d$. For example, for an average of 10 neighbors per node, $NB_L$ is 40 and $NB_{LS}$ is less than half a kilobyte. The alert buffer has at most $\gamma$ number of 2 byte entries.

From [15], the number of nodes involved in monitoring a route reply is $N_{REP} = 2r^2(h+1)d$ . Thus, given $N$ as the total number of nodes in the network, each node is involved in watching $(N_{REP}/N)f$ route replies per unit time. If the time delay for packet forwarding is $T_{FWRD}$ and using Little's law the length of the watch buffer $L_W = T_{FWRD}(N_{REP}/N)f$ . $T_{FWRD}$ depends on the processing time at the intermediate node and the MAC-layer contention delay. The processing time is negligible for route reply forwarding since replies are not hop-by-hop authenticated and negligible processing is required at an intermediate node. The MAC-layer delay for the binary exponential backoff for light to moderate loads has the mean $T_{MAC} = Gn^2$ ([44]-[46]), where $G$ is the proportionality constant that depends on the network load, and $n$ is the number of nodes contending for transmission which is equal to the number of first hop neighbors ($\pi r^2 d$) here. According to [44][45], $G$=0.01. Therefore, $L_W = G\, n^2(N_{REP}/N)f$. Each entry in the watch buffer consists of 10 bytes combined for the identity of the source, the destination, the intermediate source, the intermediate destination, and the packet sequence number. For example, if $N$ = 100 nodes, $h$ = 4 hops, and $f$ = 100 routes every one time unit, then $N_{REP}$ = 17, and each node watches only 17 route replies every one time unit. Therefore, $L_W = 0.01 \times 100 \times 17 = 17$ entries, and the total size is 170 bytes.

Each entry in the black list consists of 2 Bytes and the size of the list depends on the number of malicious nodes that has been detected. The maximum size of the buffer equals $M_{max}$ + the number of nodes that could be falsely isolated.

**Computation Overhead**: The main computational overhead is in computing the signature over the ANUM by the *CA* and verifying the signature by the rest of the nodes. If RSA is used for ANUM signing, then the cost of generating a $b$-bit signature is $O(b^3)$ and the cost of verifying the signature is $O(b^2)$. The signature generation is only done by the *CA* when the node moves. The signature verification is done during the neighbor discovery by the moving node and its first-hop and second-hop neighbors. Also during the neighbor discovery a node has to compute the distance to the neighbor using the position information, which is a simple constant time operation. The other part of computation overhead is in maintaining the neighbor list and the watch buffers by inserting, deleting and searching the buffers. These buffers, as we saw in the storage overhead computation, are relatively small data structures, so if we use single link list implementation, then insertion can be done in constant time at the head of the list, and deleting an old entry involves searching and manipulation of two pointers. The searching overhead is linear in the size of the buffer.

**Bandwidth Overhead**: The bandwidth overhead is incurred by three sources. The first is the ANUM handshake with the *CA*, which consists of an ANUM-Request by the node and an ANUM-Reply or Reject by the *CA*. This is incurred only once every time a node moves. Second, the neighbor discovery in which the moving node sends a two-hop broadcast of its ANUM and receive a one-hop unicast from its first-hop neighbors and a two-hop unicast from its second-hop neighbors. Thus the total number of one-hop ANUM transmissions is $(1+\pi r^2 d)$ for broadcasting (one by the original source and the remaining by the first hop neighbors) and $(\pi (2r)^2 d)$ for unicasting by each node within the two-hop transmission radius. The last ingredient in the communication overhead comes from the alert propagation by a guard node upon detection of a malicious node. The guard sends an alert message to the *CA* through multihop routes, broadcasts one-hop alert to inform the common neighbors of the guard and the malicious node, and several two-hop unicasts to inform the nodes that are first-hop neighbors to the malicious node and second-hop neighbors of the guard. This overhead is incurred only upon malicious node detection and can thus be considered negligible when amortized over extended periods of failure free operation.

## 6.4 Possibility of Framing

Let $N$ be total number of nodes in the network, $N_m$ be the number of malicious nodes, $P_m = N_m/N$ be the probability that a node gets compromised, $d$ be the density of nodes in the network, $R$ be the range of communication, and $N_b = \pi R^2 d$ be the number of neighbors of a node.

The probability that a good node $X$ is locally framed equals the probability that there are at least $\gamma$ malicious nodes among $X$'s neighbors which is given by,

$$P_{frame}(\gamma) = \sum_{i=\gamma}^{N_b} \binom{N_b}{i} P_m^i (1-P_m)^{N_b-i} \tag{18}$$

Note that the probability of global framing is zero based on the assumption of less than $M_{max}$ compromised nodes. The probability of node framing ($P_{frame}$) as a function of the probability of node compromise for $\gamma = 5$ and $N_b = 7$ is plotted on Figure 24. From the figure, we see that the probability of framing increases exponentially with the probability of node compromise but up to the upper end of the range, it is still less than 0.03.

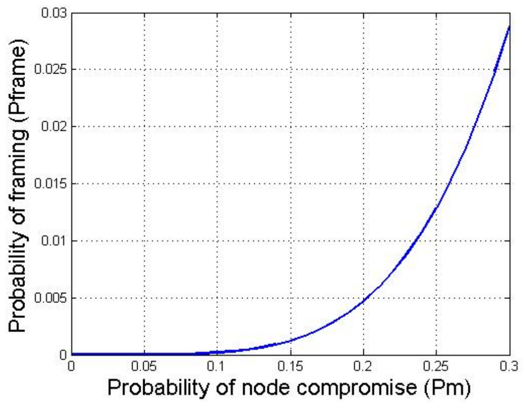

**Figure 24: Probability of node framing ($\gamma$=5, $N_b$=6)**

## 7 Conclusion

In this paper we proposed a protocol called MOBIWORP for mitigating the wormhole attack in mobile multihop ad hoc and sensor networks. MOBIWORP incorporates two protocols SMP and CAP-CV for differing degrees of functionality afforded to a mobile node. We also proposed local and global isolation protocols that will neutralize the capability of the malicious nodes from launching further attacks after detection, whether at the current location or at a new location. We demonstrated the effect of MOBIWORP under different network conditions and mobility patterns using simulations.

In the near future, we are experimenting with more accurate and yet computationally tractable ways of accumulating suspicion information from multiple guard nodes. Our current work is looking at adapting local monitoring to systems to work with sleep-awake protocols. In addition, we are investigating protocols that will choose guard nodes based on their location, eliminating the causes of loss in detection coverage or false detection.

## 8 References

[1] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Royer, A secure routing protocol for ad hoc networks, in the Proceedings of the 10th IEEE International Conference on Network Protocols, pp. 78-87, 2002.

[2] Y.-C. Hu, D. B. Johnson, and A. Perrig, SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks, in Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002), pp. 3-13, 2002.

[3] Y.-C. Hu, A. Perrig, and D. B. Johnson, Ariadne: A secure on-demand routing protocol for ad hoc networks, in ACM MobiCom'02, pp. 12-23, 2002.

[4] P. Papadimitratos and Z. Haas, Secure routing for mobile ad hoc networks, in Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002).

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in ACM MobiCom'00), pp. 65-67, 2000.

[6] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, Highly-resilient, energy-efficient multipath routing in wireless sensor networks, in ACM SIGMOBILE Mobile Computing and Communications Review, vol. 4, no. 5, pp. 11-25, 2001.

[7] F. Ye, A. Chen, S. Lu, and L. Zhang, A scalable solution to minimum cost forwarding in large sensor networks, in the 10th International Conference on Computer Communications and Networks (ICCCN), pp. 304-309, 2001.

[8] D. Braginsky and D. Estrin, Rumor routing algorithm for sensor networks, in the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), pp. 22-31, 2002.

[9] C. E. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications, pp. 234-244, 1994.

[10] C. Karlof and Y. Li, J. Polastre, ARRIVE: Algorithm for Robust Routing in Volatile Environments, Technical Report UCB/CSD-03-1233, March 2003.

[11] C. Karlof and D. Wagner, Secure Routing in Sensor Networks: Attacks and Countermeasures, in the 1st IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[12] S. Marti, T. J. Giuli, K. Lai, and M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in ACM MobiCom'00, pp. 255-265, 2000.

[13] Y. C. Hu, A. Perrig, and D.B. Johnson, Packet leashes: a defense against wormhole attacks in wireless networks, in IEEE INFOCOM, pp. 1976-1986, 2003.

[14] L. Hu and D. Evans, Using Directional Antennas to Prevent Wormhole attacks, in Network and Distributed System Security Symposium, pp. 131-141, 2004.

[15] I. Khalil, S. Bagchi, and N. B. Shroff, LITEWORP: A Lightweight Countermeasure for the Wormhole Attack in Multihop Wireless Networks, in the International Conference on Dependable Systems and Networks (DSN), pp. 612-621, 2005.

[16] Y. C. Hu, A. Perrig, and D. Johnson, Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols, in ACM Workshop on Wireless Security (WiSe'03), pp. 30-40, 2003.

[17] A. Qayyum, L. Viennot, and A. Laouiti, Multipoint Relaying: An Efficent Technique for Flooding in Mobile Wireless Networks, Technical Report Research Report RR-3898, project HIPEERCOM, INRIA, February 2000.

[18] B. Bellur and R. G. Ogier, A Reliable, Efficient Topology Broadcast for Dynamic Networks, in IEEE INFOCOM'99, pp. 178-186, 1999.

[19] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H Rubens, Mitigating Byzantine Attacks in Ad Hoc Wireless Networks, Department of Computer Science, Johns Hopkins University, Tech. Rep. Version 1, March 2004.

[20] S. Capkun, L. Buttyán, and J.-P. Hubaux, SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks, in Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN 03), pp.21-32, 2003.

[21] S. Buchegger and J.-Y. Le Boudec, Performance analysis of the CONFIDANT protocol, in ACM MobiHoc, pp. 226-236, 2002.

[22] Ralph C. Merkle, Protocols for Public Key Cryptosystems, in Proceedings of the IEEE Symposium on Security and Privacy, 1980.

[23] The Network Simulator - ns-2, At: http://www.isi.edu/nsnam/ns/

[24] A. Nasipuri, R. Castaneda, and S.R. Das, Performance of Multipath Routing for On-demand protocols in Mobile Ad Hoc Networks, in ACM Mobile Networks and Applications (MONET), pp. 339-349, 2001.

[25] A. A. Pirzada and C. McDonald, Establishing Trust In Pure Ad hoc Networks, in the Proceedings of 27th Australasian Computer Science Conference (ACSC'04), pp. 47-54, 2004.

[26] Y. Huang and W. Lee, A Cooperative Intrusion Detection System for Ad Hoc Networks, in Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03), pp. 135-147, 2003.

[27] J. Hightower and G. Borriello, Location sensing techniques, Technical Report of the University of Washington CS Department, UW-CSE-01-07-01, July 2001.

[28] C. Savarese, J. Rabaey, and K. Langendoen, Robust Positioning Algorithms for Distributed Ad hoc Wireless Sensor Networks, in USENIX Technical Annual Conference, 2002.

[29] J. Li, J. Jannotti, D.S.J. De Couto, D.R. Karger, and R. Morris, A scalable location service for geographic ad hoc routing, in ACM MobiCom, pp. 120-130, 2000.

[30] J.-H. Song, V. Wong, V. Leung, Network protocols: A framework of secure location service for position-based ad hoc routing, in Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, pp. 99-106, 2004.

[31] W. Wang and B. Bhargava, Visualization of Wormholes in Sensor Networks, in Proceedings of the ACM workshop on Wireless security (Wise'04), pp. 51-60. 2004.

[32] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang, Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach, in the IEEE Wireless Communications and Networking Conference (WCNC05), Volume 2, 13-17, pp. 1193 – 1199, 2005.

[33] L. Hu and D. Evans, Localization for Mobile Sensor Networks, in ACM MobiCom'04, pp. 45-57, 2004.

[34] D. Liu, P. Ning, and W. Du, Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks, in the 25th International Conference on Distributed Computer Systems (ICDCS'05), pp. 609-619, 2005.

[35] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, Global Positioning System: Theory and Practice, in the Fourth Edition, Springer-Verlag, 1997.

[36] L. Lazos and R. Poovendran, SeRLoc: Robust Localization for Wireless Sensor Networks, in ACM Transactions on Sensor Networks (TOSN), Volume 1, Issue 1, pp. 73-100, 2005.

[37] K. Sun, P. Ning, and C. Wang, Fault-tolerant cluster-wise clock synchronization for wireless sensor networks, in the IEEE Transactions on Dependable and Secure Computing (TDSC), Volume 2, Issue 3, pp.177–189, 2005.

[38] L. Eschenauer and V. D. Gligor, A key-management scheme for distributed sensor networks, in the Proceedings of the 9th ACM Conference on Computer and Communication Security, pp. 41- 47, 2002.

[39] R. Durrett, Essentials of Probability, Duxbury Press, 1994.

[40] W. Du, L. Fang, P. Ning, LAD: Localization Anomaly Detection for Wireless Sensor Networks, in the Journal of Parallel and Distributed Computing (JPDC), Volume 66, Issue 7, pp. 874-886, 2006.

[41] N. Sastry, U. Shankar, and D. Wagner, Secure verification of location claims, in ACM Workshop on Wireless Security (WiSe'03), pp. 1-10, 2003.

[42] J. Newsome, E. Shi, D. Song, and A. Perrig, The Sybil attack in Sensor Networks: Analysis & Defenses, IPSN'04, pp. 259-268, 2004.

[43] Q. Zhang, P. Wang, D. S. Reeves, and P. Ning, Defending against Sybil Attacks in Sensor Networks, in SDCS'05, pp. 185-191, 2005.

[44] G. Khanna, S. Bagchi, and Y.-S. Wu, Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Network, in IEEE DSN'04, pp. 739-748, 2004.

[45] J. H. Kim and J. K. Lee, Performance analysis of Mac protocols for wireless LAN in Rayleigh and shadow fast fading, in IEEE Global Telecommunications Conference (GLOBECOM), Vol. 1, pp. 404-408, 1997.

[46] T. Khattab, M. El-Hadidi, and H. Mourad, Analysis of wireless CSMA/CA network using single station superposition (SSS), International Journal of Electronics and Communications (AE), vol. 56, pp. 71-81, 2002.

**Issa Khalil** received the B.Sc. degree in computer engineering from Jordan University of Science and Technology (JUST), Jordan, in 1994, and the MS degree in computer engineering from JUST in 1996. He is currently pursuing a PhD in the Dependable Computing Systems Lab of Prof. Bagchi S. His research interest includes key-management, secure routing protocols, and intrusion detection in Ad Hoc and Sensor networks. He has worked as the director of computer and communication center of Alquds Open University, West Bank, for more than 6 years.

**Saurabh Bagchi** joined the department of Electrical and Computer Engineering at Purdue University in West Lafayette, Indiana as an Assistant Professor in August 2002. Before that, he did his Ph.D. from the Computer Science department of the University of Illinois at Urbana-Champaign with Prof. Ravishankar Iyer at the Coordinated Science Laboratory. His Ph.D. dissertation was on error detection protocols in distributed systems and was implemented in a fault-tolerant middleware system called Chameleon.

**Ness B. Shroff** received his Ph.D. degree from Columbia University, NY in 1994. He is currently a full Professor in the School of Electrical and Computer Engineering at Purdue University. His research interests span the areas of wireless and wire line communication networks, and more recently network security. Dr. Shroff is an editor for IEEE/ACM Trans. on Networking and the Computer Networks Journal, and past editor of IEEE Communications Letters. He was the conference chair for the 14th Annual IEEE Computer Communications Workshop in Estes Park, CO, October 1999) and program co-chair for the symposium on high-speed networks, Globecom 2001 (San Francisco, CA, November 2000). He was the Technical Program co-chair for IEEE INFOCOM'03 and panel co-chair for ACM Mobicom'02. He received the NSF CAREER award in 1996 and the best paper of the year award for Computer Networks, 2003.