Discovering Minimum Exposed Path to Attack in Mobile Ad hoc Networks in optimal $O(|P|)$ time after pre-processing

NEELIMA GUPTA
Department of Computer Science, University of Delhi, Delhi, India
and
SANDHYA KHURANA
Institue of Informatics and Communication, University of Delhi South Campus, Delhi, India

Several protocols have been proposed to handle attacks on routing protocols in Ad hoc Networks ($MANET$s). However, no single protocol handles all the attacks. Then a related problem would be to establish routes which are at maximum distance from nodes in danger of attack. We say that such paths are exposed minimally to the attack. In this paper, we present an algorithm to find Minimum Exposed Path to Attack ($MEPA$) in optimal $O(|P|)$ time where $|P|$ is the length of $MEPA$ route. The algorithm works in two phases: the preprocessing phase, and the route establishment phase. In the preprocessing phase minimum distances from endangered nodes are computed for all nodes in the network. Once the initial distances are computed, $MEPA$ routes are discovered in the route establishment phase. Whenever a route with greater distance from the endangered node is found, the previous route is discarded and the new one is kept. In the absence of mobility the algorithm converges in $(O|P|)$ time after a preprocessing step. The preprocessing step takes $O(D)$ time where D is the diameter of the network. When a node moves, maintenance phase takes $O(D)$ time to recompute the distances and then the algorithm takes $(O|P|)$ time to recompute the $MEPA$ route. Assuming that the packets are received from the shortest path first, the algorithm computes the shortest $MEPA$ route. We simulated our protocol in Network Simulator($NS2$). The performance of our protocol is comparable with Ad hoc On-demand Distance Vector routing algorithm ($AODV$) in the absence of endangered nodes and better than many existing ones in presence of blackhole /wormhole/ both the attacks. It was also observed that in absence of mobility the algorithm converges in optimal $O(|P|)$ time.

Keywords: Ad hoc Network, Attack, Routing, Security, Endangered Node, Minimum Exposed Path.

1. INTRODUCTION

Ad hoc networks [Ch. E. Perkins 1994] have been proposed to support scenarios where no wired infrastructure exists. They can be set up quickly where the existing infrastructure does not meet application requirements for reasons such as security, cost, or quality. Examples of applications for ad hoc networks range from military operations, emergency disaster relief to community networking and interaction between attendees at a meeting or students during a lecture.

Attacks in $MANET$s [Michiardi and Molva 2002b] are threat for basic network functions like packet forwarding and routing. Achieving $MANET$s free from attacks is challenging due to lack of infrastructure, dynamically changing topologies, wireless links vulnerable to attacks like eavesdropping and spoofing. The dynamic nature of the network emphasizes the need for solutions to be dynamic. Routing of packets is one of the basic functions performed in any network. Nodes of an ad hoc network rely on each other to forward the packets due to the limited range of the nodes. As a result, embedding solutions in routing protocols to handle attacks poses a challenge to the researchers. An attacker can cripple the network by inserting erroneous routing updates, replaying old routing information, changing routing updates, or advertising incorrect routing information so that the network is not able to provide service properly.

Two main approaches are used to mitigate attacks in ad hoc networks. The first approach aims

Author's address: S. Aneja, Institue of Informatics and Communication, University of Delhi South Campus, Delhi, India.

at detecting the malicious nodes while computing the route in the network and re-routing the packets around it, mostly along the shortest path among them. Most of these protocols [Buchegger and Boudec 2002; Buttyan and Hubaux 2001; Hu et al. 2002; Hu et al. 2002; Khurana et al. 2006; Michiardi and Molva 2002a; Papadimitratos and Haas 2002; Sanzgiri et al. 2002; Yi et al. 2002] are based on existing ad hoc routing protocols like Dynamic Source Routing (DSR) [Johnson and Maltz 1996], Ad-hoc On demand Distance Vector ($AODV$) [Perkins et al. 2003] and Destination Sequence Distance Vector ($DSDV$) [Perkins and Bhagwat 1994] redesigned to handle attacks. The second approach [Huang and Lee 2003; Zhang and Lee 2000; Zhang et al. 2003] separates the detection of malicious nodes from routing.

None of the existing protocols handles all types of attacks. Hence it is imperative to find a solution that would reduce the impact of attacks on routing. In this paper, we present an algorithm to find a path between a pair of nodes such that the path is at maximum distance from the nodes which are in danger of attack. We call such paths as Minimum Exposed Path to Attack ($MEPA$). To the best of our knowledge the problem has not been addressed earlier for ad hoc networks. The related problem in the context of sensor networks is 'maximal breach path' problem [Buragohain et al. 2006; Huang et al. 2005; Li et al. 2003; Li et al. 2003; Liu et al. 2009; Meguerdichian et al. 2001; Mehta et al. 2003]. Maximal breach path is defined as the path which is as far away as possible from the sensors i.e the minimum distance from sensors is maximized on this path. The path finds the area of low observability from the sensor nodes. A solution to 'maximal breach path' problem can be used to compare two given configurations of sensors in a battlefield or, to add new sensors to a network to increase the observability or the coverage area.

Our algorithm (henceforth referred to as $MEPA$) consists of a preprocessing phase where all the nodes compute their distances from the endangered nodes. Once the initial distances are computed, $MEPA$ routes are discovered in a manner similar to that in $AODV$ in the route establishment phase. Whenever a route with greater distance from the endangered node is found, the previous one is discarded and the new one is kept. We show, theoretically as well as through simulations, that the algorithm converges in $(O|P|)$ time after a preprocessing step where $|P|$ is the length of the $MEPA$ route. The preprocessing step takes $O(D)$ time where D is the diameter of the network. When a node moves, maintenance phase takes $O(D)$ time to recompute the distances and the new $MEPA$ route is computed in $(O|P|)$ time. Assuming that the packets are received from the shortest path first, the algorithm computes a shortest $MEPA$ route. As in $AODV$ a path with minimum number of hops is shortest.

We simulated our protocol in $NS2$. $MEPA$ was compared with $AODV$ in the absence of endangered nodes and the performance was found to be comparable. We also compared $MEPA$ with RAODV(Reliable Ad-hoc On demand Distance Vector) [Khurana et al. 2006] and Deng's algorithm (henceforth referred to as $DENG$) [Deng et al. 2002] to address the blackhole attack and with EEW(End-to-End scheme to secure a network against Wormhole attack) [Khurana and Gupta 2010] to mitigate wormhole attack. $MEPA$ out performs all the three algorithms in presence of blackhole/wormhole/both attacks in terms of packet delivery ratio and average end to end delay. Routing overhead of $MEPA$ is slightly higher due to pre-processing phase when the number of communicating pairs is less. However as the number of connections increase, the time taken in the path establishment phase starts dominating and $MEPA$ performs better than $DENG$ and $RAODV$ and, is comparable to EEW in terms of routing overhead. We have not included overhead of intrusion detection scheme into overhead incurred by our scheme.

2. NETWORK MODEL

In this section, we present network model used for MEPA. We assume that there are two types of nodes present in the network. First type of nodes are good nodes and second type of nodes are those that are under the danger of attack. Information about the endangered nodes spreads with time as they are discovered to be under the danger of attack.

3. PROBLEM STATEMENT AND NOTATIONS

Endangered nodes are defined as the nodes in a network which are under the danger of attack. 'Minimum Exposed Path to Attack ($MEPA$)' is a path which is farthest from the endangered nodes, i.e. a path whose minimum distance from the endangered nodes is maximized.

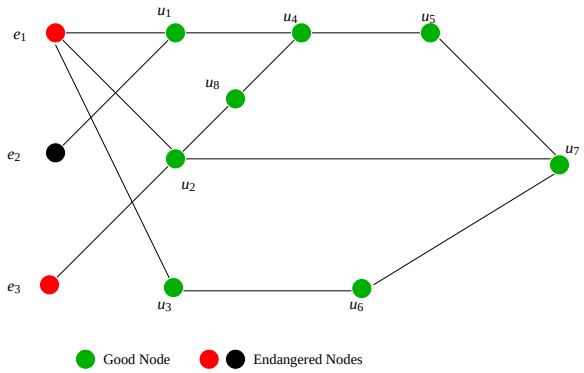
Definition 3.1 *Let E be the known subset of nodes that are in danger of attack or jamming. For any node u in the network,*

(1) let $\delta(u)$ denote the minimum distance of node u from the set E i.e. $\delta(u) = min_{\ e\in E}\{dist(u,e)\}$, where $dist(u,e)$ is the distance of u from e in terms of number of hops.

(2) let $Nb(u)$ denote the set of neighbors of u. Define $pr(u)$ to be the node, in the neighborhood of u, through which the distance of u from E is minimum i.e. $pr(u) = u_i$ where $u_i = argmin_{v\in Nb(u)}\{\delta(v)\}$. For example, in Figure 1, let $E = \{e_1, e_2, e_3\}$. $\delta(u_1) = 1, \delta(u_2) = 1, \delta(u_3) = 1, \delta(u_4) = 2, \delta(u_5) = 3, \delta(u_6) = 2, \delta(u_7) = 2, \delta(u_8) = 2$, $pr(u_4) = u_1, pr(u_5) = u_4, pr(u_6) = u_3$, $pr(u_7) = u_2$ and $pr(u_8) = u_2$.

(3) For a set S, $Nb(S) = \cup_{u\in S}\{Nb(u)\}$. For example, $Nb(E)$ in Figure 1 is $\{u_1, u_2, u_3\}$.

These terms are used in the pre processing phase of our algorithm.



Computation of $\delta(u)$ and $pr(u)$.

Definition 3.2 *For a path P between s and t, define $dist(P,E) = min_{u\in P: u\neq s,t}\{\delta(u)\}$ then $MEPA(s,t) = argmax_P\{dist(P,E)\}$.*

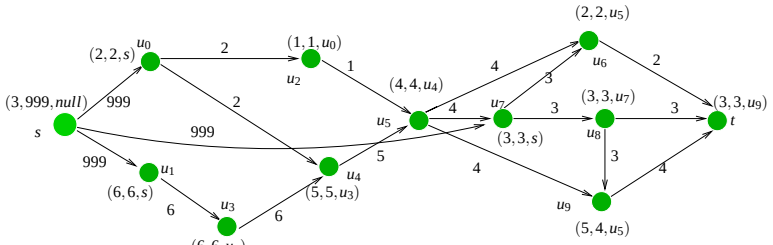
Definition 3.3 *Let a node s wants to establish a $MEPA$ route to another node t. Let $MEPA(s,v)$ denote the 'Minimum Exposed Path to Attack' from s to a node v . Let $DEN(s,v)$ denote the distance of the $MEPA$ route $MEPA(s,v)$ from the endangered nodes. Then,*

(1) $DEN(s,v) = min\{max_{v_i\in Nb(v)}\{DEN(s,v_i)\}, \delta(v)\}$ for $v \neq s,t$,
$DEN(s,t) = max_{v\in N(t)}\{DEN(s,v)\}$ and $DEN(s,s) = \infty$.

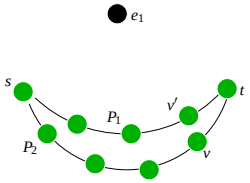
(2) $p(s,v) = argmax_{v_i\in Nb(v)}\{DEN(s,v_i)\} for\ v \neq s$, and $p(s,s) = Null$.

$p(s,v)$ denotes the next hop for v on the reverse route of $MEPA$, we call it the precursor of v on the $MEPA$ route. These terms are used in the route-establishment phase of the algorithm.

Consider Figure 2. Each node u is labeled with the triplet ($\delta(u)$, $DEN(s,u)$, $p(s,u)$) and edge (u, v) is labeled with $DEN(s,u)$. Consider node u_5. Three routes from s to u_5 have 'distance from E' as $1, 2$ and 5, maximum being 5. $DEN(s,u_5) = min\{5, \delta(u_5)\} = min\{5,4\} = 4$. Since distance 5 was received from the node u_4 therefore $p(s,u_5) = u_4$. As can be seen from the figure that this indeed is the next hop at u_5 on the reverse $MEPA$ route from t to s. $DEN(s,s)$ is set to ∞ so that $DEN(s,u)$ is set appropriately for all $u \in Nb(s)$. Note that the distances of s and t from the endangered nodes should not be considered while computing the $MEPA$ distances . See Figure 3, if we will include $\delta(t)$ or $\delta(s)$ in the computation of $MEPA$ route, routes like P_2 will never be discovered if P_1 has already been established.



Computation of $DEN(s,u)$. Each node u is labeled with the triplet $(\delta(u)\ , DEN(s,u), p(s,u))$ and edge (u,v) is labeled with $DEN(s,u)$.



Effect of $\delta(t)$ and $\delta(s)$ on the computation of $MEPA$ route where e_1 is an endangered node.

3.1 Related Work

Several secure routing protocols [Buchegger and Boudec 2002; Buttyan and Hubaux 2001; Hu et al. 2002; Hu et al. 2002; Khurana et al. 2006; Michiardi and Molva 2002a; Papadimitratos and Haas 2002; Sanzgiri et al. 2002; Yi et al. 2002] exist to handle attacks in $MANET$s. For example, Secure Routing Protocol proposed in [Yi et al. 2002] based on DSR assures that a node initiating route discovery is able to identify and discard replies providing false routing information but it fails when two or more malicious nodes cooperate resulting in wormhole attack. $ARIADNE$ [Hu et al. 2002] and Security-aware Ad-hoc Routing (SAR) [Papadimitratos and Haas 2002] protocol handle attacks such as spoofing, changing routing updates with the help of keys and certification of messages. However, they do not handle the attack by selfish nodes. Selfish nodes do not intend to cripple the network but do not cooperate in the normal functioning of the basic services like packet forwarding in order to save energy. In [Buttyan and Hubaux 2001] Buttyan and Hubaux have suggested a solution based on virtual currency called Nuglet to locate the selfish nodes in the network. It handles attacks due to selfish nodes but not the attacks due to malicious nodes. $CONFIDANT$(Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) by Buchegger and Boudec [Buchegger and Boudec 2002], $CORE$ (COllaborative REputation) by Michiardi and Molva [Michiardi and Molva 2002a] and $RAODV$ by Khurana et al. in [Khurana et al. 2006] detect malicious or selfish nodes , isolate them and route the packets around them. However they are vulnerable to spoofing attacks.

In [Meguerdichian et al. 2001] Meguerdichian et al. have addressed the problem of computing a Maximal Breach Path in a sensor network. The algorithm uses Voronoi diagram and takes $O(n^2 \log n)$ time where n is the number of sensor nodes. Dinesh et al. [Mehta et al. 2003] and Lie et al. [Li et al. 2003] have improved the running time to $O(n \log n)$. In [Mehta et al. 2003], after a preprocessing step, they compute the maximal breach path P in optimal $O(|P|)$ time, where $|P|$ denotes the number of edges on the optimal path. The preprocessing takes $O(n \log n)$ time. In [Buragohain et al. 2006] and [Liu et al. 2009], authors consider a sensor network which senses obstacles and develop distributed protocols to compute a safe path. Chiranjeeb et al. have proposed a distributed algorithm to determine safe paths in sparse networks of size $O(n^{1/2+\varepsilon})$, where n is the total number of nodes in the networks, where the quality (length, exposure, etc.) of the path is within a small constant factor of the optimal. Li et al. [Li et al. 2003] propose a protocol that guarantees safest path (path with the largest clearance of obstacle zones). They also develop a protocol that distributes this information in the sensor network.

In [Huang et al. 2005] Hai Huang et al. have addressed the problem of maintaining the distance of the maximal breach path in a dynamic scenario. They have given an algorithm that

approximates the distance of maximal breach path within an approximation factor of $(\sqrt{2}+\varepsilon)$, for any $\varepsilon > 0$ in polylogarthmic time. If required, the maximal breach path can be computed in $O(n \log n)$ time.

To the best of our knowledge the problem of finding a $MEPA$ path has not been addressed earlier in the context of ad hoc networks.

4. ALGORITHM TO COMPUTE A $MEPA$ ROUTE

We assume the existence of a mechanism that enables the nodes to detect the endangered nodes. In [Zhang and Lee 2000; Huang and Lee 2003; Zhang et al. 2003] Lee et al have used intrusion detection techniques to detect the presence of an intruder in wireless ad hoc networks.

Our algorithm works in two phases. Phase-I is a 'preprocessing phase' in which all the nodes compute their distances from the endangered nodes. Once the distances from the endangered nodes have been computed, nodes can set up $MEPA$ routes in Phase-II. Due to the highly mobile nature of the nodes, the distances may have to be updated dynamically as the nodes move. This is done in the 'maintenance' phase. For the sake of simplicity and better understanding, we have presented the phases separately but in actual implementation 'maintenance' phase and the 'route-establishment' phase occur simultaneously.

4.1 Phase-I : The Preprocessing Phase

Let $in\_nbhd(u)$ be a flag that denotes whether u is in the neighborhood of E or not. Initially the flag $in\_nbhd(u)$ is off and the value of $\delta(u)$ is ∞ for all the nodes u in the network. Nodes compute their distances from E as follows:

(1) Any node $u \in Nb(E)$ knows that it is in $Nb(E)$ when it hears from an endangered node. It sets its $in\_nbhd(u)$ flag to 1, $\delta(u)$ to 1 and broadcasts its distance to all its neighbors.

(2) Let u be a node not in $Nb(E)$. When u receives $\delta(v)$ from its neighbor v, it updates its $\delta(u)$ as follows: if the distance of u from E is shorter through v than its current value (that is, if $\delta(v)+1 < \delta(u)$) then it updates $\delta(u)$ to $\delta(v)+1$ and sets $pr(u)$ to v.

Whenever a node updates its distance from E, it broadcasts the updated distance to all its neighbors except to the one through which the new distance was computed. Algorithm 1 summarizes phase-I of $MEPA$.

Assuming that packets are received from the shortest path first, no updation is done in the preprocessing phase and the distances are established in $O(D)$ time where D is the diameter of the network. We will see in section 4.3 that, as the nodes move, distances are updated in the maintenance phase.

Initialization: $in\_nbhd(u) \leftarrow 0$, $\delta(u) \leftarrow \infty$
The Preprocessing Phase: Computing distances from Endangered nodes
1.1 a node $x \in Nb(E)$ hears from an endandgered node:
$in\_nbhd(x) \leftarrow 1,\ \delta(x) \leftarrow 1,\ pr(x) \leftarrow NULL.$
it broadcasts its distance to all its neighbors.
1.2 when a node u receives $\delta(v)$ from its neighbor v:
if *(* $\delta(v)+1 < \delta(u)$*)* **then**
  $\delta(u) \leftarrow \delta(v)+1$, $pr(u) \leftarrow v$, it forwards its $\delta(u)$ to all its neighbors.
else
  it discards the packet.
end

Algorithm 1: Phase-I of $MEPA$

We have used a packet called 'Distance Unit' (DU) to broadcast the 'distance from E'. '$\delta elta$' field in this packet holds the distance of the node from the endangered nodes. Format of the DU

packet is

Distance Unit(DU)				
Type	DU id	Src id	Src Seq Number	$\delta elta$

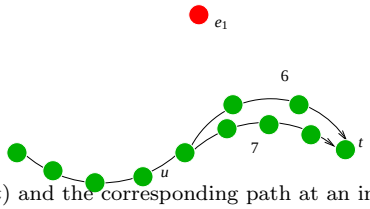
4.2 Phase-II : Establish a $MEPA$ route

Let a node s wants to establish a $MEPA$ route to another node t. The algorithm works in a breadth first manner starting from s. $DEN(s,u)$ is initialized to 0 for all $u \neq s$ and $DEN(s,s)$ is set to ∞. s broadcasts its $DEN(s,s)$ (included in route request packet) to its neighbors. Let $u \in Nb(s)$, u updates its $DEN(s,u)$ and $p(s,u)$ (explained below) if required and, broadcasts its $DEN(s,u)$ (included in route request packet) to its neighbors if updated. $u's$ neighbors then update their DENs and precursors if required and so on. In the process, a node may receive DENs more than once. Each time it receives a new DEN, it updates its information if required and broadcasts further (all this happens while processing the route request packets).

Suppose an intermediate node u receives $DEN(s,v)$ from a node v then u updates $DEN(s,u)$ and $p(s,u)$ as follows: If $DEN(s,v) > DEN(s,u)$ then $DEN(s,u)$ will be updated to $min\{DEN(s,v), \delta(u)\}$ and $p(s,u)$ will be updated to v. This case is exhibited in Figure 2 at node u_4. Suppose node u_4 receives $DEN(s,u_0) = 2$. Since $DEN(s,u_4)$ was initialized to 0 therefore $DEN(s,u_0) > DEN(s,u_4)$ and hence $DEN(s,u_4)$ is updated to $min\{\ DEN(s,u_0), \delta(u_4)\} = min\{2,5\} = 2$ and $p(s,u_4) = u_0$. Later when it receives $DEN(s,u_3) = 6$, then since $DEN(s,u_3) > DEN(s,u_4)$, $DEN(s,u_4)$ is updated to $min\{DEN\ (s,u_3), \delta(u_4)\} = min\{6,5\} = 5$ and $p(s,u_4)$ is updated to u_3.

When the destination t receives $DEN(s,v)$ from a node v then t updates $DEN(s,t)$ and $p(s,t)$ as follows: If $DEN(s,v) > DEN(s,t)$ then it updates $DEN(s,t)$ to $DEN(s,\ v)$ and $p(s,t)$ to v. t then sends a reply packet to v. Note that the destination may reply back to several nodes in case it receives a path with a higher DEN value later. The destination copies the value of $DEN(s,t)$ in the reply packet.

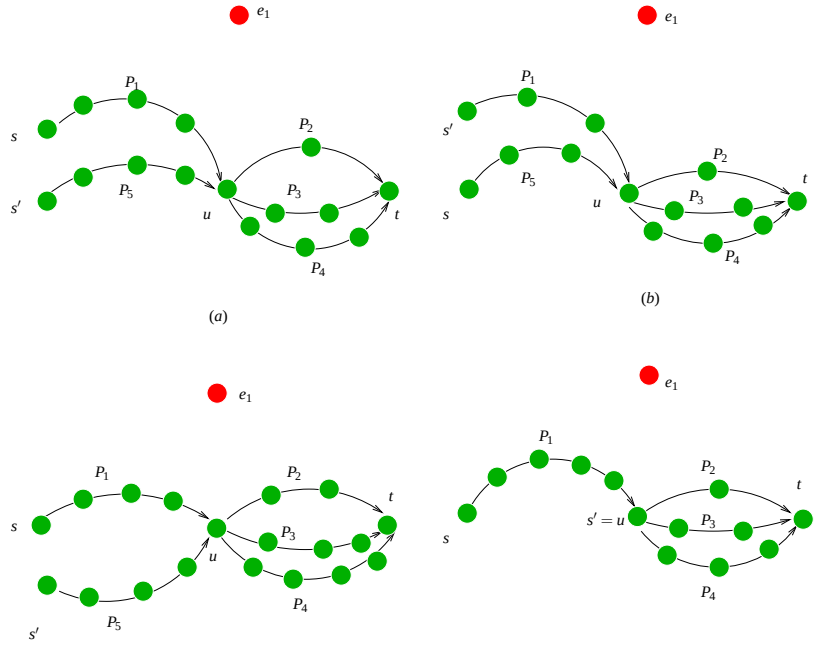
When an intermediate node v receives the reply packet, it makes an entry for the destination t, sets the next hop on the $MEPA$ path and copies $DEN(s,t)$ in its routing table. It also forwards the packet to its precursor $p(s,v)$ on the reverse path. In case a node receives multiple replies, it updates its routing table entries and forwards the reply packet only if received DEN is higher than the previous one. When the source node s receives the reply packet, it starts sending the data packets on that path. In case the source node receives multiple replies, it updates its $MEPA$ path only if the received DEN is higher than the previous one.



Updating $DEN(s,t)$ and the corresponding path at an intermediate node u

Notice that we need to store $DEN(s,t)$ at every intermediate node u so that when destination sends multiple replies with the updated DENs, this updation takes effect on u as well. Consider the scenario in Figure 4. Suppose t sends a reply with DEN set to 6 first and then set to 7. u needs to store $DEN(s,t)$ to compare the stored DEN value and the received DEN value so as to be able to take a decision whether to update its next hop or not. If u receives reply with DEN value 6 before the one with DEN value 7 then it should update its table entries but if it receives DEN value 7 before 6 than it should not.

Next, we will show that this decision does not depend upon s and hence $DEN(s,t)$ can be stored in the entry for t in the routing table for u. Consider a path P for which u contains a value Δ for $DEN(s,t)$. Suppose while computing a $MEPA$ path between another source s' and t, t sends a route reply with $DEN(s',t)$ equal to Δ' through u. u updates its next hop for t,



Impact of another communication on $DEN(s,t)$ and the corresponding path at an intermediate node u. (a) $DEN(P_1) < DEN(P_2) < DEN(P_5) < DEN(P_3) < DEN(P_4)$ (b) $DEN(P_1) < DEN(P_5) < DEN(P_2) < DEN(P_3) < DEN(P_4)$ (c) $DEN(P_1) < DEN(P_2) < DEN(P_3) < DEN(P_4) < DEN(P_5)$ (d) $s' = u$, $DEN(P_1) < DEN(P_2) < DEN(P_3) < DEN(P_4)$.

only if $\Delta' > \Delta$. Let us see the impact of doing this on P. See Figure 5, $MEPA$ path between s and t would be of the form $P1$ -$P2$ or $P1$-$P3$ or $P1$-$P4$. Note that due to s'-t communication it might change from one of these to another but would still remain $MEPA$ route i.e. the next hop in the entry for t in the routing table of u might change but the new route would still be a $MEPA$ route between s and t.

Algorithm 2 summarizes phase-II of $MEPA$. Processing of $MEPA\_REQ$ and $MEPA\_REP$ is shown in Figures 12, 13, 14 and 15. We replaced the hopcount field in the $RREQ$ and $RREP$ packets of $AODV$ by the DEN field. They are now called $MEPA\_REQ$ and $MEPA\_REP$ respectively. Similarly hopcount field in the routing table entry of $AODV$ was replaced with DEN. Formats of routing table entry of $MEPA$, $MEPA\_REQ$ and $MEPA\_REP$ packets are shown below:

$MEPA$ Routing Table Entry Format							
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	DEN	Next Hop	Precursors	Life Time

MEPA_REQ						
Type	MEPA_REQ id	Dest id	Dest Seq Number	Src id	Src Seq Number	DEN

MEPA_REP					
Type	Dest id	Dest Seq Number	Src id	DEN	Life Time

Initialization: $DEN(s,s) \leftarrow \infty$, $DEN(s,u) \leftarrow 0$, $DEN(s,t) \leftarrow 0$
Phase-II: Establish a $MEPA$ route

2.1 Source node broadcasts a route request containing $DEN(s,s)$.

2.2 When an intermediate node u receives a route request from another node say v containing $DEN(s,v)$ it checks its routing table:
if *($DEN(s,v) > DEN(s,u)$)* **then**
  $DEN(s,u) \leftarrow min\{DEN(s,v), \delta(u)\}$.
  $p(s,u) \leftarrow v$.
  it broadcasts the route request containing $DEN(s,u)$ further to its neighbors.
else
  it discards the packet.
end

2.3 When destination receives a route request from a node v containing $DEN(s,v)$ it checks its routing table:
if *($DEN(s,v) > DEN(s,t)$)* **then**
  $DEN(s,t) \leftarrow DEN(s,v)$.
  $p(s,t) \leftarrow v$.
  it sends route reply packet containing $DEN(s,t)$ to v.
else
  It discards the packet.
end

2.4 When an intermediate node receives a route reply packet containing $DEN(s,t)$ from a node say u it checks its routing table:
if *(stored $DEN(s,t) <$ received $DEN(s,t)$)* **then**
  it updates stored $DEN(s,t)$ with the received $DEN(s,t)$.
  it sets the next hop for t to u.
  it forwards reply packet to next hop on reverse path.
else
  It discards the packet.
end

2.5 When source node receives a $MEPA\_REP$ packet it checks its routing table:
if *(stored $DEN(s,t) <$ received $DEN(s,t)$)* **then**
  it updates stored $DEN(s,t)$ with the received $DEN(s,t)$.
  it sets the next hop for t to u.
  it starts sending data packets on the discovered path.
else
  It discards the packet.
end

Algorithm 2: Phase-II of $MEPA$

In the next theorem, we show that as the updated $DENs$ propagate from s to t, the entire algorithm converges in $O(|P|)$ time, where $|P|$ is the length of the $MEPA$ route. Note that we do not need to know the length of the $MEPA$ path for the result to hold.

Theorem 4.1 *Suppose a $MEPA$ route is to be established between the nodes s and t. Then at any node u on the $MEPA$ route between s and t, final value of $DEN(s,u)$ will be set in l hopcounts, where l is the length of the $MEPA$ route from s to u. In particular t sets final value of its $DEN(s,t)$, in $|P|$ hopcounts where $|P|$ is the length of the $MEPA$ route.*

Proof Consider a $MEPA$ route $s, n_1, \ldots n_l, \ldots t$. Let $n_l \neq t$ be the l^{th} node on the $MEPA$ route. Then, n_l receives $MEPA\_REQ$ originated by s after l hopcounts via $MEPA$ route. In the worst case, n_l receives $l-1$ $MEPA\_REQs$ from other paths before getting $MEPA\_REQ$ from its precursor node on the $MEPA$ route. Thus, n_l updates $DEN(s,n_l)$ at most $(l-1)$ times; once it gets a $MEPA\_REQ$ along its $MEPA$ route no further updation to $DEN(s,n_l)$

takes place in future. To see this consider See Figure 7.4 :

Let the part of the $MEPA$ route from s to n_l be denoted by P_1. That is the $MEPA$ route from s to t is P_1 $n_l \dots t$. Suppose n_l receives a $MEPA\_REQ$ from a node v' along P_1 first and then it receives $MEPA\_REQ$ with higher DEN from a node v via another path P_2 in future. There are following cases:

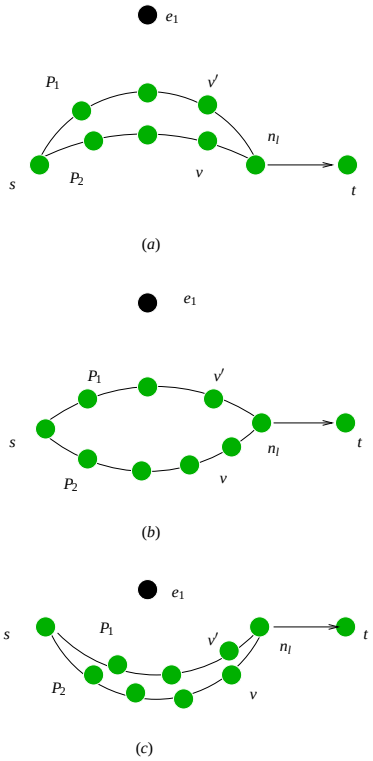
case1(a): See Figure 6(a). $DEN(s, v') < \delta(n_l)$, $DEN(s, v) < \delta(n_l)$. Such cases cannot arise because in these cases $MEPA$ route from s to t would be $P_2 n_l \dots t$ and not $P_1 n_l \dots t$.

case1(b): See Figure 6(b). $DEN(s, v') < \delta(n_l)$, $DEN(s, v) > \delta(n_l)$. Such cases cannot arise because in these cases $MEPA$ route from s to t would be $P_2 n_l \dots t$ and not $P_1 n_l \dots t$.

case2: See Figure 6 (c). $DEN(s, v') > \delta(n_l)$, $DEN(s, v) > \delta(n_l)$. In this case n_l receives a higher $DEN(s, v)$ from v on P_2. Then, as per step 1 of Section 13, $DEN(s, n_l)$ is not updated as $min\{DEN(s, v), \delta(n_l)\} = \delta(n_l)$ and $DEN(s, n_l)$ is already set to $\delta(n_l)$. However $p(s, n_l)$ is updated in this case.

Thus, once n_l gets a $MEPA\_REQ$ along its $MEPA$ route, $DEN(s, n_l)$ is not updated in future and the final value of $DEN(s, n_l)$ is set in l hopcounts.

Step 1 of Figure 13 is not applicable when $n_l = t$ (instead, step 1 of Figure 14 is applicable) but these cases do not exist for t as in that case P_2 would be the $MEPA$ route instead of P_1. Thus, t also sets final value of its $DEN(s, t)$ in $|P|$ hopcounts where $|P|$ is the length of the $MEPA$ route. □



e_1 is an endangered node (a) $DEN(s, v') < \delta(n_l)$, $DEN(s, v) < \delta(n_l)$ (b) $DEN(s, v') < \delta(n_l)$, $DEN(s, v) > \delta(n_l)$ (c) $DEN(s, v') > \delta(n_l)$, $DEN(s, v) > \delta(n_l)$

Theorem 4.2 *In the absence of endangered nodes $MEPA$ route is the shortest route .*

Proof Let s be the source node that wants to establish $MEPA$ route to destination t. In the absence of endangered node, for any node u, $\delta(u) = \infty$. Let v be a node on $MEPA$

path from s to t then initially $DEN(s,v) = 0$. Initially s will broadcast $DEN(s,s) = \infty$ to its neighbors. For node $v_i \in Nb(s)$, $DEN(s,v_i) = min\{max\{DEN(s,s), DEN(s,\ v_i)\}, \delta(v_i)\}$ $= min\{max\{\infty, 0\}, \infty\} = \infty$. Then for node $v \notin Nb(s)$ on $MEPA$ route, $DEN(s,v) = min\{max_{v_i \in Nb(v)}\{DEN(s,v_i)\}, \delta(v)\} = min\{max_{v_i \in Nb(v)}\{\infty, \infty\}\} = \infty$.

Thus the $MEPA\_REQ$ packets are treated the same way as $RREQ$ packets in $AODV$ except that an intermediate node does not reply to a $MEPA$ route request. There is no effect of $\delta's$ and $DENs$ on the discovered route.

Note that for an intermediate node v, once $DEN(s,v)$ is set to ∞ it will never change. So v will broadcast $DEN(s,v) = \infty$ to it neighbors and so on. When destination receives $DEN(s,v)$ from any node v first time it will also set $DEN(s,t) = max_{v \in N(t)}\{DEN(s,v)\} = max_{v \in N(t)}\{\infty\} = \infty$. For destination also $DEN(s,t)$ is set to ∞ , it is never changed since it cannot receive DENs higher than ∞. So in the absence of endangered node, once $MEPA$ path is set it never changes. Assuming that the request which reaches the destination first follows the shortest path, $MEPA$ route is the shortest route.

□

Note: By keeping an additional hopcount field and storing the path with the smallest hopcount when $DENs$ are equal, we get $MEPA$ route which is shortest in terms of hopcount as well.

4.3 Maintenance of distances

As the nodes in an ad hoc network are highly mobile, distances must be updated as the nodes move or the links break or new links are established. Our algorithm updates the distances as described below:

Consider a case when a node moves out of the range of another node.

(1) Let $u \in Nb(E)$. As long as, it is in the range of at least one endangered node, it does nothing. As soon as it stops hearing from all the endangered nodes (for example, when an endangered node has moved or u has moved), it switches off its in_nbhd flag. Now, distances of all those nodes v (including u), whose minimum distance from E was through u, need to be updated. This is done as follows: u broadcasts a request for 'distance from E' to its neighbors, which in turn pass on the request to their neighbors. The process continues till the request is received by some other neighbor of E.
Now, consider a node v. It may receive the request from its $pr(v)$ or from a node which is not $pr(v)$. That is, it may receive the request from a node through which v had the shortest path to E or it may not be from such a node. (For example, in Figure 1, the node u_7 may receive the request from $pr(u_7)$ i.e. u_2 when both the nodes e_1 and e_3 have moved out of the range of u_2 or from u_6 when e_1 has moved out of the range of u_3). In the former case, $\delta(v)$ needs to be recomputed. So it resets $\delta(v)$ to ∞ on seeing the request and broadcasts it to its neighbors. In the latter case, $\delta(v)$ need not be recomputed (one can see that $\delta(u_7)$ is not affected in case e_1 has moved out of range of u_3) so, it simply broadcasts the request without resetting $\delta(v)$.
When other neighbors of E receive the request they do not broadcast it further and reply to the request by broadcasting their 'distance from E' . Distances of all the nodes are recomputed as it is done in phase-I. For example in Figure 1, in case u_2 receives the request of u_3 (when e_1 has moved out of the range of u_3) via u_7, it broadcasts its 'distance from E' ; it is received by u_7 and u_8. Next u_4 receives the distance broadcast from u_2 via u_8. Since $\delta(u_8) + 1 = 3 > 2 = \delta(u_4)$ (when u_4 received the request of u_3 from u_7 earlier then, since $u_7 \neq pr(u_4)$, $\delta(u_4)$ was not reset to ∞) therefore $\delta(u_4)$ does not change. Same is the case of u_7. But when u_6 receives $\delta(u_7) + 1$ then, since $\delta(u_6)$ was set to ∞, therefore it will be updated to 3. If we had set $\delta(u_4)$ and all nodes receiving the request of u_3 to ∞, $\delta(u_4)$ would have been updated to 3 which is wrong. Also, if $\delta(u_6)$ was not set to ∞ then it would not have been updated from 2 to 3.

(2) Let u be a node not in $Nb(E)$ and $pr(u)$ moves out of the range of u or u moves out of range

of $pr(u)$. In this case u will broadcast the request for 'distance from E' to its neighbors and the procedure explained above is repeated.

Next, consider a scenario in which a node u has moved into the range of another node v or v has moved into the range of u. If u is an endangered node, then v switches its $in\_nbhd(v)$ flag on, sets $\delta(v)$ to 1 and broadcasts its distance to its neighbors. The distances of all other nodes not in $Nb(E)$ are updated as explained earlier. If u is not an endangered node, its distance from E may have changed. Thus it broadcasts the request for 'distance from E' to its neighbors and the distances of all the nodes including u and v are updated according to the procedure explained above. A packet called 'Distance Discovery Unit' (DDU) is used to request for 'distance from E'. Algorithm 3 summarizes the maintenance phase of $MEPA$.

Initialization: $in\_nbhd(u) \leftarrow 1$, $\delta(u) \leftarrow x$

Let u be a node in the network.

3.1 **if** *($u \in Nb(E)$)* **then**
  if *(u stops hearing from all the endangered nodes)* **then**
    $in\_nbhd(u) \leftarrow 0$.
    it broadcasts a DDU packet.
  else
    it does nothing.
  end
end

3.2 Suppose a node u moves out of the range of another node v
if *($pr(u)$ moves out of the range of u or u moves out of the range of $pr(u)$)* **then**
  u broadcasts a DDU packet.
end

3.3 Suppose a node u moves into the range of another node v or v moves into the range of u
if *(u is an endangered node)* **then**
  $\delta(v) \leftarrow 1$.
  v broadcasts DU packet to its neighbors.
else
  distance of u from E may have changed; it broadcasts a DDU packet.
end

3.4 When a node v receives a DDU packet
if *($v \notin Nb(E)$)* **then**
  if *(the DDU packet is from pr(v))* **then**
    $\delta(v) \leftarrow \infty$.
    it broadcasts the DDU packet to its neighbors.
  else
    It broadcasts the DDU packet to its neighbors.
  end
else
  It replies back with a DU packet.
end

3.5 When a node receives a DU packet
it updates its distance from E, if required, as it is done in phase-I and broadcasts the packet to its neighbors if the distance is updated.

Algorithm 3: Maintenance Phase of $MEPA$

It clearly takes $O(D)$ time for a request of 'distance from E' to reach a node in $Nb(E)$ and come back. The format of DDU packet is

Distance Discovery Unit(DDU)			
Type	DDU id	Src id	Source Seq Number

5. SIMULATION STUDY

We simulated our protocol using $NS2$. Performance of $MEPA$ was studied in the presence of multiple attacks (blackhole and wormhole attacks) and compared with $RAODV$, $DENG$ and EEW. The proposed scheme was also compared with $AODV$ in the absence of endangered nodes. We have not included overhead of intrusion detection scheme into overhead incurred by our scheme.

5.1 Simulation Design

Simulation results were obtained for 50 nodes located over $1000m$ by $1000m$ region. The traffic sources are CBR (constant bit rate), 512-byte data packet, sending rate 1 pkt/sec and with maximum load of 300 packets for one transaction. The node movement speed is set from 0 to 80, which will be closer to real applications. The mobility are done with pause time 100 second. Script was executed for 300 seconds.

5.2 Simulation Results

To study the performance of $MEPA$ in presence of attack, a node was selected randomly to behave like a blackhole node and a pair of nodes were selected randomly to implement a wormhole channel. The performance of $MEPA$ was compared with $RAODV$, $DENG$ and EEW. It was observed that no packet was delivered in case of $RAODV$ and $DENG$ as a path through wormhole was established and packets were dropped there after. Similarly no packet was delivered in case of EEW as a path was established through blackhole node.

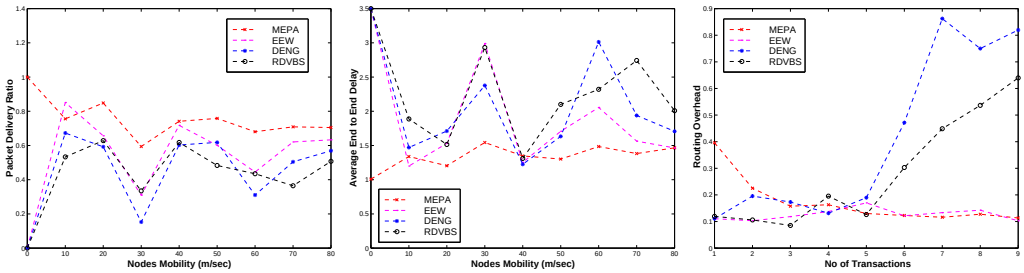
In the absence of mobility, packet delivery ratio of $MEPA$ was 1. $AEED$ is not defined (ND) for $DENG$, EEW and $RAODV$ as no packet was delivered. These results are summarized in Table I. Routing overhead of $MEPA$ is a little more initially when the number of connections is less as the advantage of pre-processing phase is less in that case. As the number of connections increase, the time taken in the path establishment phase starts dominating and $MEPA$ performs better than $DENG$ and $RAODV$ in terms of routing overhead. Routing overhead of $MEPA$ is comparable with EEW. This is shown in Figure 7c. Even in presence of mobility, $MEPA$ outperforms the other three algorithms as shown in Figures 7a and 7b.

We also compared the performance of $MEPA$ with $DENG$ and $RAODV$ in presence of **blackhole node only**. Packet delivery ratio and average end to end delay of $MEPA$ and $RAODV$ were found to be better than that of $DENG$ (Figures 8a and 8b) and routing overhead of $MEPA$ was better than that of $RAODV$ and $DENG$ (Figure 8c). Comparison of performance of $MEPA$ and EEW in presence of **wormhole node only** revealed that the packet delivery ratio and average end to end delay of $MEPA$ and EEW were comparable (Figures 9a and 9b). Routing overhead of $MEPA$ was also comparable to EEW when the number of connections was more (Figure 9c).

	PDR	AEED
$RAODV$	0	ND
$MEPA$	1	.9731
$DENG$	0	ND
EEW	0	ND

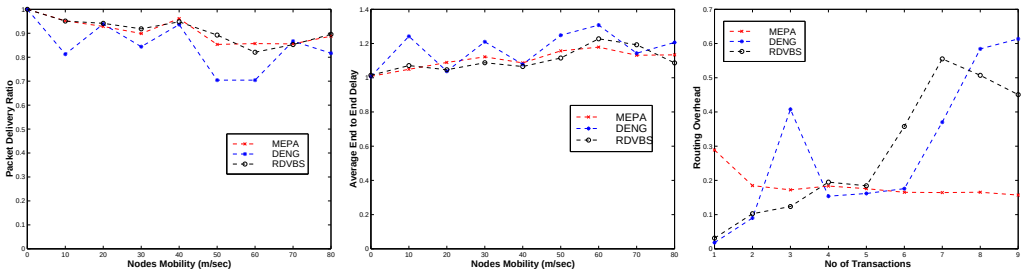
Table I: Comparison of $MEPA$ with other algorithms in the absence of mobility and in the presence of blackhole and wormhole nodes

We compared our protocol with $AODV$ in the absence of endangered nodes. When there are no endangered nodes, the pre processing phase and maintenance phases do not come into effect and $MEPA$ reduces to route-discovery phase. Mobility in $MEPA$ is handled in a similar way as it is done in $AODV$. Thus our results (packet delivery ratio and average end to end delay)



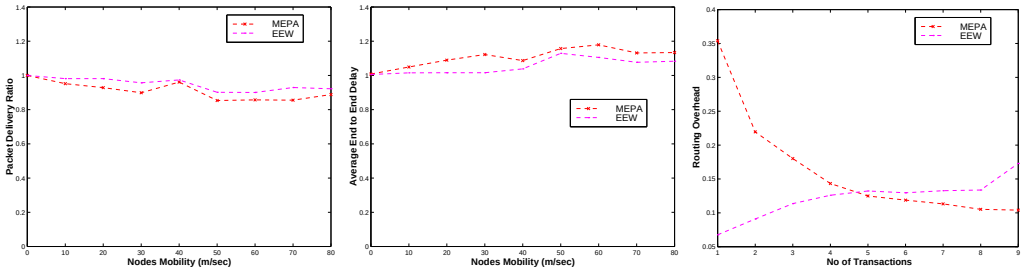
(a) Packet delivery ratio (b) Average End to End Delay (c) Routing Overhead

Comparison in presence of blackhole and wormhole nodes



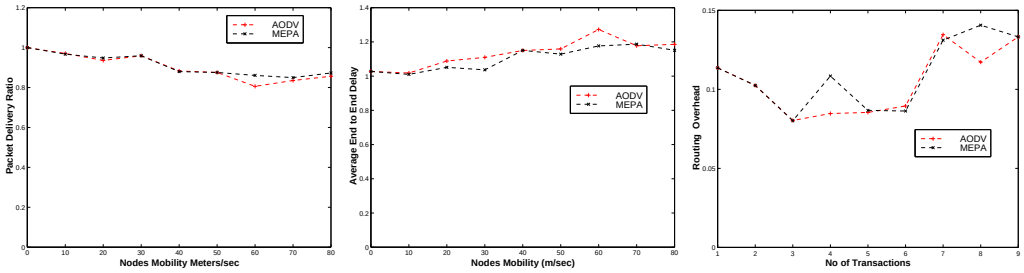
(a) Packet delivery ratio (b) Average End to End Delay (c) Routing Overhead

Comparison in presence of blackhole node only



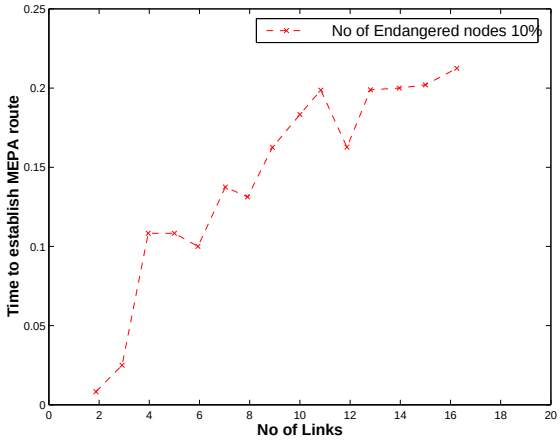
(a) Packet delivery ratio (b) Average End to End Delay (c) Routing Overhead

Comparison in presence of wormhole node only



(a) Packet delivery ratio (b) Average End to End Delay (c) Routing Overhead

Comparison in absence of endangered node



(a)

Time to establish a $MEPA$ route

are comparable to those of $AODV$, see Figures 10a and 10b. The routing overhead is also comparable except in few cases where an intermediate node has a path in case of $AODV$. This is exhibited in Figure 10c

To show that the $MEPA$ path is established in $O(|P|)$ time the algorithm was simulated for a network with 10% of nodes as endangered with various communicating pairs of varying length of $MEPA$ routes. Each simulation was run for 1000 seconds. It was observed that the destination received multiple $MEPA\_REQ$ with $DENs$ at increasing time intervals. Destination sends $MEPA\_REP$ corresponding replies to each $MEPA\_REQ$. Whenever the destination received a $MEPA\_REQ$ with higher $DENs$, it discarded the previous route and replied to the new $MEPA\_REQ$ and a new route was established. In Figure 11a the time taken to establish the $MEPA$ route is plotted against the length of the $MEPA$ route. Observe that the time to establish the $MEPA$ route is bounded by a linear function of the length of the route.

When source s wants to communicate with node t, it creates a $MEPA\_REQ$ packet with $DEN(s,s)$ set to ∞ and broadcasts the packet to its neighbors.

Processing of $MEPA\_REQ$ at the source node in $MEPA$

For an intermediate node u, $DEN(s,u) = 0$. When u receives a $MEPA\_REQ$ packet from another node v, it compares the received $DEN(s,v)$ from the packet with its $DEN(s,u)$.

(1) if $DEN(s,v) > DEN(s,u)$ then it sets $DEN(s,u) = min\{DEN(s,v), \delta(u)\}$, $p(s,u) = v$, replaces the DEN in $MEPA\_REQ$ packet with its own $DEN(s,u)$ and broadcasts it to its neighbors.

(2) else it discards the packet.

Processing of $MEPA\_REQ$ at an intermediate node in $MEPA$

$DEN(s,t)$ is initialized to 0. When $MEPA\_REQ$ packet arrives at the destination from node v, it compares $DEN(s,v)$ with $DEN(s,t)$.

(1) if $DEN(s,v) > DEN(s,t)$ then it sets $p(s,t) = v$, $DEN(s,t) = DEN(s,v)$ and sends $MEPA\_REP$ to v.

(2) else it discards the packet.

Note that the destination may send multiple $MEPA\_REPs$ to the source if it receives $MEPA\_REQ$ packets with higher $DENs$ later.

Processing of $MEPA\_REQ$ at the destination in $MEPA$

Let v be an intermediate node that receives a $MEPA\_REP$ packet from a node u.

(1) if v is receiving $MEPA\_REP$ packet for the first time, it sets $DEN(s,t)$ to the value it receives from the packet, sets the next hop for t to u and forwards $MEPA\_REP$ to $p(s,v)$ on the reverse path.

(2) else (a node may receive multiple replies) v updates $DEN(s,t)$ and sets the next hop for t in its routing table if the received $DEN(s,t)$ is higher than the stored value and forwards $MEPA\_REP$ to $p(s,v)$ on the reverse path.

When source node s receives a $MEPA\_REP$ packet from a node u:

(1) if s is receiving $MEPA\_REP$ packet for the first time, it sets $DEN(s,t)$ to the value it receives from $MEPA\_REP$, sets the next hop for t to u and starts sending the data packets on this path.

(2) else (source node may also receive multiple replies) it updates its $DEN(\ s,t)$ and sets the next hop for t in its routing table if the received $DEN(s,t)$ is higher than the stored value, discards the previous path and starts sending the data packets on the new path.

Processing of $MEPA\_REP$ in $MEPA$

6. CONCLUSION AND FUTURE WORK

None of the existing algorithms to secure routing protocols in ad hoc networks handle all types of attacks. In this paper, we do not address any particular type of attack but discover a path that is exposed minimally to the set of endangered nodes. The problem was posed by Farago [Basagni and Conti and Giordano and Stojmenovic 2004] as a challenging algorithmic problem for ad hoc networks. Our algorithm is simple and fast. It computes the shortest secure path in optimal $O(|P|)$ time after a preprocessing step of $O(D)$ time.

Assuming that the packets are received from the shortest path first, the algorithm computes the shortest $MEPA$ route. By keeping the hop count field and keeping the path with smallest hop count when $DENs$ are equal we get $MEPA$ routes which are shortest in terms of hop count as well.

The protocol can be improved to the one where an intermediate node keeps a $MEPA$ route to the destination and reply to a $MEPA\_REQ$. Moreover, with more careful insight into the protocol, the size of the routing table can also be reduced to the case where we do not keep an entry for every communicating pair but only at most one entry per node. In large $MANETs$ (comprising hundreds or thousands of nodes), using $MEPA$ paths might lead to long paths leading to increase in the message transmission times/delays when the messages could be transmitted over a shorter path around the malicious nodes. In such a scenario, it would be interesting to discover a path which is at least some k hop away from endangered nodes instead of finding a path which is at maximal distance from the endangered nodes. In future we would like to modify the protocol to find these types of paths.

Further, in future we would like to make the protocol self-sufficient without the use of schemes like intrusion detection for two reasons:one, they are not completely error free and may provide false positives/ false negatives and secondly, they add an overhead. In case of false negative, an endangered node may be included into the MEPA route and our algorithm will fail. In case of false positive, the algorithm will run correctly but will slow down since it is excluding a good node which could be an important part of the network.

REFERENCES

Basagni and Conti and Giordano and Stojmenovic. 2004. *Mobile Ad Hoc Networking*. Wiley IEEE Press.

Buchegger, S. and Boudec, J.-Y. L. 2002. Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*. Lausanne, Switzerland, 226–236.

Buragohain, C., Agrawal, D., and Suri, S. 2006. Distributed navigation algorithms for sensor networks. In *Proceedings of the IEEE INFOCOM*.

Buttyan, L. and Hubaux, J. P. 2001. Nuglets: a virtual currency to stimulate cooperation in selforganized ad hoc networks. In *Technical Report DSC/2001/001, Swiss Federal Institute of Technology – Lausanne*.

Ch. E. Perkins. 1994. *Ad Hoc Networking*. Addison Wesley.

Deng, H., Li, W., and Agrawal, D. P. 2002. Routing security in wireless ad hoc networks. In *IEEE Communications Magazine*. Vol. 40. 70–75.

Hu, Y. C., Johnson, D. B., and Perrig, A. 2002. 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)*. Calicoon, NY, 3–13.

Hu, Y. C., Perrig, A., and Johnson, D. B. 2002. Ariadne :a secure on-demand routing protocol for ad hoc networks. In *proceedings of IEEE MOBICOM*.

Huang, H., Richa, A. W., and Segal, M. 2005. Dynamic coverage in ad-hoc sensor networks. In *Mobile Networks and Applications*. 9–17.

Huang, Y. and Lee, W. 2003. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the First ACM Workshop Security of Ad Hoc and Sensor Networks*. Fairfax, Virginia.

Johnson, D. B. and Maltz, D. A. 1996. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*. Kluwer Academic Publishers, 153–181.

Khurana, S. and Gupta, N. 2010. End-to-end protocol to secure ad hoc networks against wormhole attacks. *Wiley Journal of Security and Communication Networks 4*, 10.1002/sec.272.

Khurana, S., Gupta, N., and Aneja, N. 2006. Reliable ad-hoc on-demand distance vector routing protocol. In *Proceeding of the IEEE International Conference on Networking(ICN)*. 98–103.

Li, Q., DeRosa, M., and Rus, D. 2003. Distributed algorithms for guiding across a sensor network. In *Proceedings of the IEEE MOBIOCOM*.

Li, X.-Y., Wan, P.-J., and Frieder, O. 2003. Coverage in wireless ad-hoc sensor networks. *IEEE Transactions on Computers 52*, 1–11.

Liu, Y., Wang, J., and Yang, Z. 2009. Sensor network navigation algorithms without locations. In *Proceedings of the IEEE INFOCOM*.

Meguerdichian, S., Koushanfar, F., Potkonjak, M., and Srivastava, M. 2001. Coverage problems in wireless ad hoc sensor networks. In *Proceedings of 6th IEEE World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*. Orlando, FL, 1380–1387.

Mehta, D. P., Lopez, M. A., and Lin, L. 2003. Optimal coverage paths in ad-hoc sensor networks. In *Proceedings of ICC*. Vol. 1. 507–511.

Michiardi, P. and Molva, R. 2002a. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the Sixth IFIP conference on security communications and multimedia*. Portoroz, Slovenia.

Michiardi, P. and Molva, R. 2002b. Simulation-based analysis of security exposures in mobile ad hoc networks. In *Proceedings of European Wireless Conference*.

Papadimitratos, P. and Haas, Z. 2002. Secure routing for mobile ad hoc networks. In *Proceedings of CNDS*.

Perkins, C. E. and Bhagwat, P. 1994. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*. 234–244.

Perkins, C. E., Royer, E. M. B., and Das, S. R. 2003. Ad-hoc on-demand distance vector (AODV) routing. In *Mobile Ad-hoc Networking Working Group*. Internet Draft.

Sanzgiri, K., Dahill, B., Levine, B. N., Shields, C., and Belding-Royer, E. M. 2002. ARAN: A secure routing protocol for adhoc networks. In *UMass Tech Report*. 02–32.

Yi, S., Naldurg, P., and Kravets, R. 2002. A security-aware routing protocol for wireless ad hoc networks. In *Proceedings Of ACM Symposium On Mobile Ad hoc Networking & Computing (MOBIHOC)*. 286–292.

Zhang, Y. and Lee, W. 2000. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking(MobiCom)*.

Zhang, Y., Lee, W., and Huang, Y. 2003. Intrusion detection techniques for mobile wireless networks. In *Wireless Networks 9, Kluwer Academic Publishers*. 545–556.

Neelima Gupta is Associate Professor at Department of Computer Science, University of Delhi. She completed her M.Tech (CS) and Ph.D. (CS) from IIT Delhi. She is interested in Routing Algorithms in Ad hoc Networks, Network Security, Algorithms, Data Mining and Bio-informatics.

Dr. Sandhya Khurana Sandhya Khurana is Assistant Professor at institute of Informatics and Communication, University of Delhi South Campus. She completed her M.Tech. (Computer Science) from IIT Delhi and PhD (Computer Science) from University of Delhi. Her areas of interest are Network Security and Routing in Mobile Ad hoc Networks, Delay Tolerant Networks and Cognitive Radio Ad hoc Networks.