Egyptian Informatics Journal (2011) **12**, 29–36

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

Securing DSR with mobile agents in wireless ad hoc networks

Ahmed Abosamra [a,*], **Mohamed Hashem** [b], **Gamal Darwish** [c]

[a] *Department of Information Technology, Cairo University, Egypt*
[b] *Department of Information Systems, Ain Shams University, Egypt*
[c] *Department of Information Technology, Cairo University, Egypt*

Received 25 May 2009; accepted 3 January 2011
Available online 27 March 2011

KEYWORDS

Ad hoc networks;
Mobile agents;
Routing protocols;
DSR;
Network security

Abstract Ad hoc wireless network consists of a set of wireless nodes communicating with each other without a pre-defined infrastructure. They communicate by forwarding packets which can reach wireless nodes that do not exist in the range of the direct radio transmission. Designing ad hoc network routing protocols is a challenging task because of its decentralized infrastructure which makes securing ad hoc networks more and more challenging. Dynamic Source Routing (DSR) protocol is a popular routing protocol designed for use in wireless ad hoc networks. Mobile agent is a promising technology used in diverse fields of network applications. In this paper, we try to implement DSR using mobile agents for securing this type of wireless network. Hybrid encryption technique (symmetric key encryption/public key encryption) is used to improve performance; where symmetric keys are used to encrypt routing data to authenticate and authorize node sending data, while, public keys are used for the exchange of symmetric keys between nodes. We found that DSR may be secured using mobile agents with competitive performance.

1. Introduction

Ad hoc wireless network consists of a set of wireless nodes communicating with each other without a pre-defined infrastructure while having the capability of delivering packets through routes created on the fly while nodes are on motion or not. These capabilities give ad hoc networks interesting attributes such as no need for centralized administration, and can be created quickly which make them suitable for applications such as military operations, business meetings outside the office and disaster recovery. These mentioned applications require secure and reliable communication as a prerequisite for using ad hoc networks [1].

* Corresponding author. Tel.: +20 181666831.
E-mail addresses: abuhmeid@gmail.com (A. Abosamra), mohamed.hashem.2006@gmail.com (M. Hashem), gdarwish@mcit.gov.eg (G. Darwish).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2011.02.003

Production and hosting by Elsevier

Designing ad hoc network routing protocols is a challenging task which makes securing ad hoc networks more and more challenging. There are two types of routing protocols for ad hoc networks: reactive (on-demand), and proactive (table-driven or periodic) routing protocols. In reactive routing protocols sending nodes discovers routes whenever they need to send data to target nodes, but, proactive protocols maintain fresh lists of destinations and their routes by periodically distributing routing tables throughout the network. Reactive protocols need fewer amounts of data for maintenance and give faster reaction on restructuring and failures in most cases when compared to proactive protocols [2].

In this paper, we introduce a new protocol for securing a reactive protocol called DSR (or Dynamic Source Routing) [3] using mobile agents [4].

Section 2 shows how DSR works. Section 3 encloses a brief description of mobile agents. Section 4 illustrates a new protocol that combines mobile agents with DSR for securing ad hoc networks. Section 5 compares the new protocol with DSR. Conclusion and future work are given in Section 6.

2. How DSR works?

DSR is composed of two parts; route discovery and route maintenance. It is based on source routing which means that the node sending a data packet lists in its header the nodes that the packet shall go through. A brief discussion for route maintenance and route discovery is given in the following subsection.

2.1. Route discovery

When an initiating node (initiator) wants to send a data packet to a target node (target), it looks in its route cache for a route to the target node, if a route is found it is used to send the packet. In case no route is found in the route cache the initiator node broadcasts a route request with a unique identifier with respect to the route requests recently sent before from this node to the nodes in its direct radio transmission range in the ad hoc network. In case a receiving node has seen a route request from the initiator with the same identifier before; it discards the route request, otherwise, if it is the target of the route request; it sends a route reply with the passed nodes, otherwise, it looks for a route to the target of the route request in its route cache and sends a route reply with the route if found, if not found, it appends its address to the passed nodes in the route request and re-broadcasts the route request to the surrounding nodes in its direct radio transmission range. Route Discovery and route reply are summarized in Figs. 1 and 2.

2.2. Route maintenance

Route maintenance in DSR is the process of making sure that a sent data packet has reached the destination and there are no broken hops through the route because two nodes became too far, for example. To apply this process, each node receiving a data packet throughout the route listed in the packet's header sends an acknowledgment to its previous node. In case, no acknowledgment is received after a fixed number of re-transmission of a data packet this hop is considered as broken and a route error is sent to the sending node using the same route used by the packet to reach the current node. Then, the initiator removes the broken hop from the routes in its route cache and initiates a route request to the target if needed.

Many optimizations have been made to DSR, but, we are focusing here on securing DSR with no optimizations using mobile agents.

3. Mobile agents

To give a definition for mobile agents, remote procedure call (RPC) and remote evaluation must be addressed, As well as the history evaluation of mobile agents.

RPC is an inter-process communication technology that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction [5].

Remote evaluation lets one computer send another computer a request in the form of a program. A computer that receives such a request executes the program in the request and returns the results to the sending computer. Remote evaluation provides a new degree of flexibility in the design of distributed systems. For distributed systems that use remote procedure calls, server computers are designed to offer a fixed set of services. In a system that uses remote evaluation, server computers are more properly viewed as programmable processors. One consequence of this flexibility is that remote evaluation can reduce the amount of communication that is required to accomplish a given task [5] which is useful for wireless ad hoc network where resources – battery power, for example – may be limited.

The term agent comes from Greek "agein", which means to drive or to lead. The software mobile agent paradigm arose as an extension of the remote evaluation paradigm. Software mobile agents are programs that can migrate from computer to computer through a wired/wireless/hybrid network. Moving here means that a mobile agent can stop its execution, saves its status, moves to another computer, and continues its execution, Fig. 3 shows a mobile agent system architecture.

The mobile agent paradigm (shown in Fig. 4) differs from the client/server paradigm. In the client/server paradigm, resource owners (servers) are physically distant from their clients (users). The communication among these parts occurs through a network of computers, being mediated by mechanisms as remote procedure calls, message exchange, sockets and so on. In this paradigm, the reliability of the communication links and the synchronicity of the remote procedure calls are important requirements of the majority of such applications.

On the other hand, in the mobile agent paradigm, the agents migrate to interact locally, at the same host as the resources. Migration of mobile agent may be weak migration or strong migration. In weak migration, mobile agent moves to the next host and restart its execution from the code beginning while in strong migration the mobile agent completes execution from the point it reached at the host where it was before migration, refer to Fig. 5 for a brief view on degrees of mobility.

Fig. 6 shows an example of a discovery mobile agent created by a browser and moving through the network searching for computers.

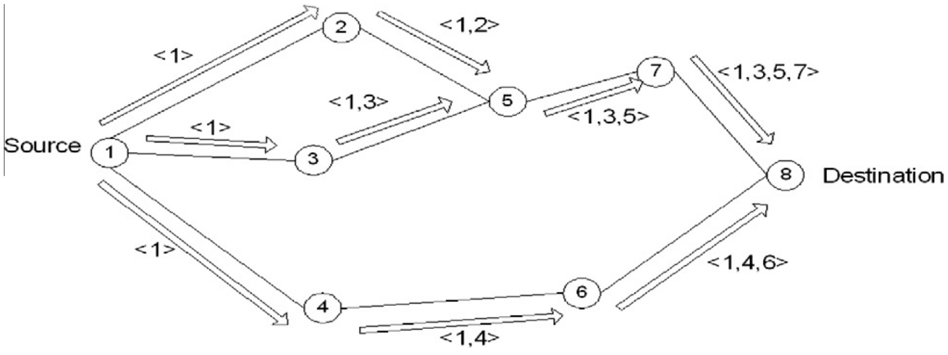

Figure 1 Route discovery.

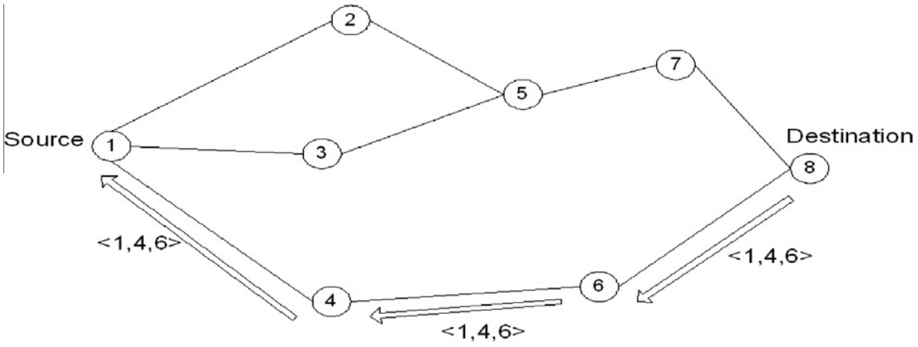

Figure 2 Route reply.

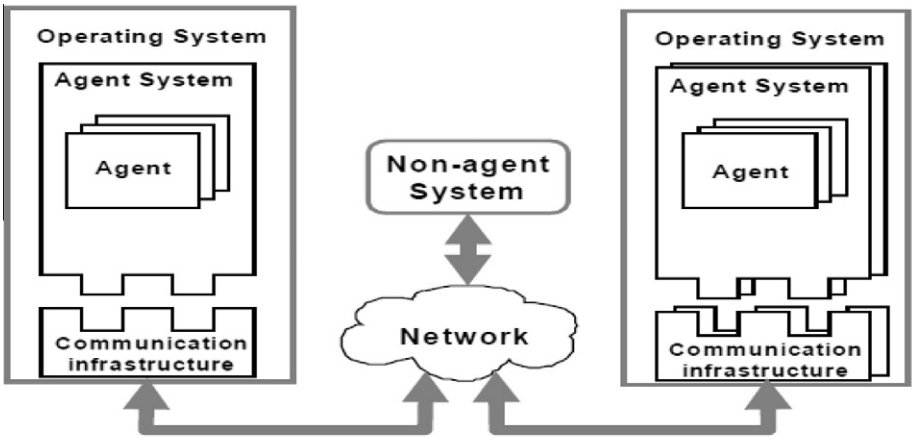

Figure 3 Mobile agent facility architecture.

4. DSR with mobile agents

This section handles how an ad hoc network may be secured by modifying DSR to use mobile agents. There are three types of mobile agents used in this routing protocol:

1. Discovery/reply of mobile agent.
2. Maintenance of mobile agent.
3. Update/approve for symmetric key mobile agent.

4.1. Discovery/reply process

When a node (A) wants to send a data packet to target node (C) it creates a discovery mobile agent (DMA) with a new discovery ID, source node = (A), target node = (C), new public key, empty list of passed nodes, an empty list of symmetric keys, and sets its state to discovery. Then, node (A) broadcasts the mobile agent to the nodes in its wireless direct radio transmission range.

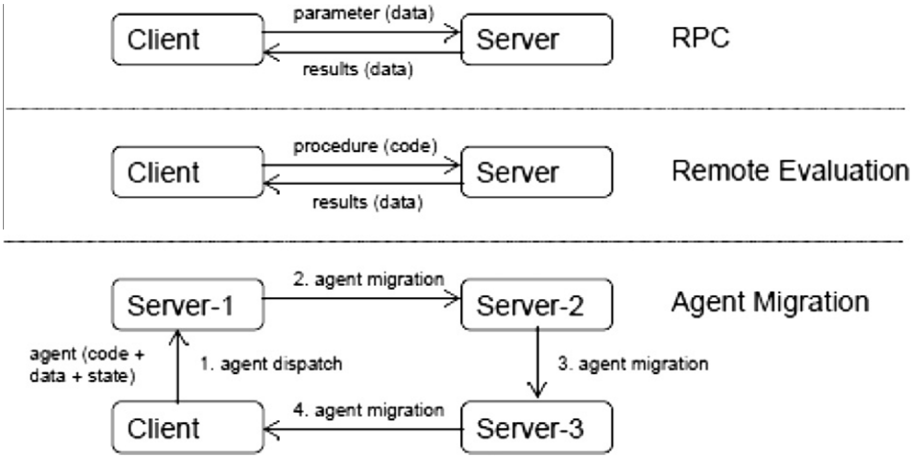

Figure 4 Evolution of the mobile agent paradigm.

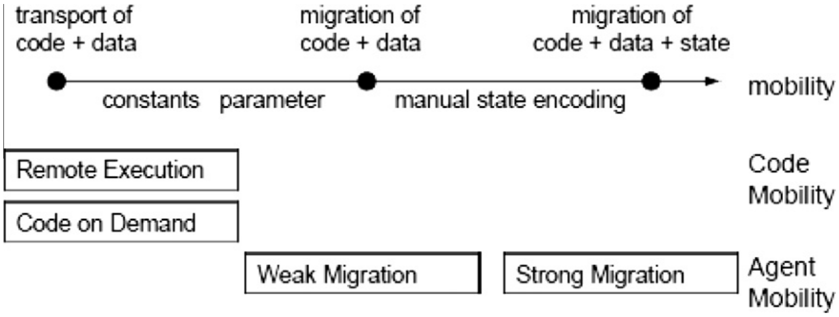

Figure 5 Degrees of mobility.

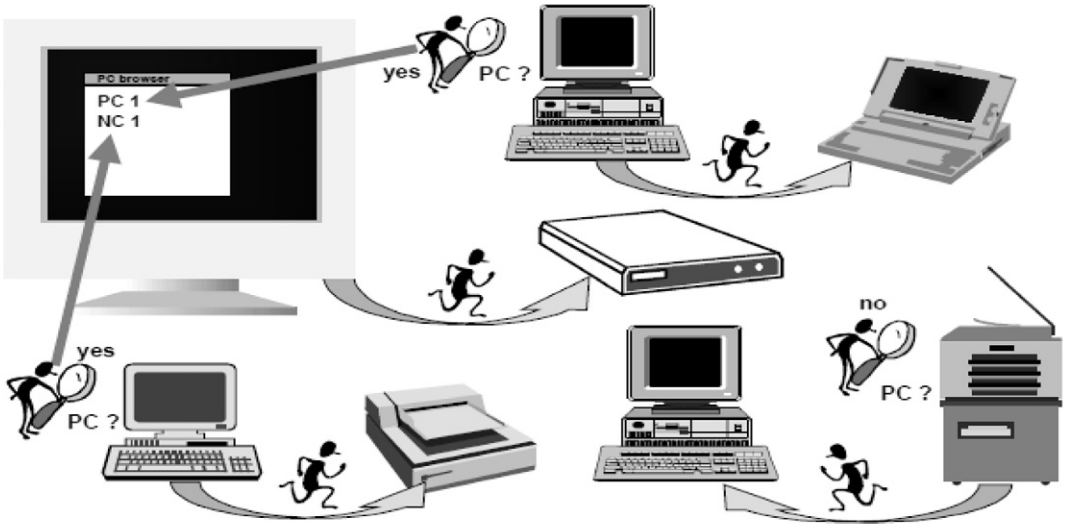

Figure 6 Mobile agents example.

Once a discovery mobile agent is received by a wireless node it is discarded in case it has been received before by the same node, otherwise, the node checks if it is the target of the discovery. In case, it is not the target it appends its node address to the list of passed nodes in the mobile agent and it adds a new symmetric key encrypted with the public key in the mobile agent to the list of symmetric keys in case it does not already have a symmetric key shared with the source node, otherwise it adds the existing symmetric key encrypted with the public key in the mobile agent to the list of symmetric keys in the mobile agent.

In case, it is the target of the discovery, the wireless node appends its node address to the list of passed nodes, appends the shared symmetric key with the source node (creates a new one in case there is no existing shared symmetric key) encrypted with the public key in the mobile agent, then, changes the state of the mobile agent to reply and sends it through the route in the passed nodes after reversing it.

Now, the source wireless node has a route to the target node and a symmetric key with all the nodes along the route to the target node.

4.2. Maintenance process

When there is a truncated hop in a route discovered before, the maintenance process is used to notify the sending node with the truncated hop to re-discover a route to the target. A maintenance mobile agent is used to implement the maintenance when it is sent by the first node in the truncated hop having a new maintenance ID, first node in truncated hop, second node in truncated hop, and all the previous fields encrypted by the symmetric key shared with the source node, refer to Fig. 7 for a simple example of secured DSR with the source node. Fig. 8, illustrates a flow chart for DSR with mobile agents.

4.3. Update/approve symmetric key process

The symmetric keys shared between nodes may be updated by the update symmetric key process. A request to update symmetric key shared between node (A) and node (B) is sent from (A) to (B) with a public key and an update Id and all the previous fields encrypted using the shared symmetric key. (B) Replies with a new created symmetric key to (A) encrypted with the public key sent from (A) then (A) approves the new symmetric key with the Id sent in the update.

The mobile agent types with their internal fields are as following:

(a) Discovery/reply mobile agent:
- Discover ID.
- Source node.
- Target node.
- Number of hops.
- Maximum number of hops.
- List of passed nodes.
- Public key generated to encrypt symmetric keys.

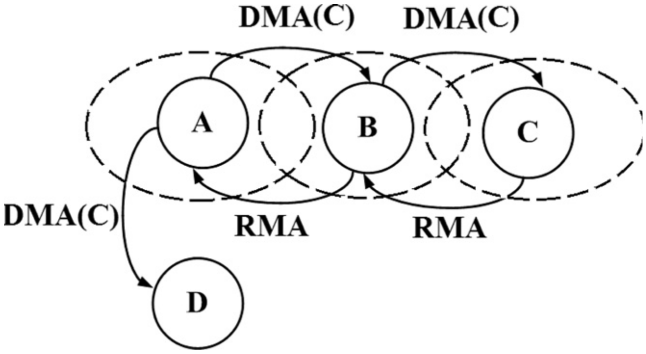

Figure 7 Simple example of secured DSR with mobile agent protocol.

- Symmetric keys encrypted using public key.
- State {discovery/reply}.

(b) Maintenance mobile agent:
- Maintenance ID.
- First node in truncated hop.
- Second node in truncated hop.
- MAC (message authentication code): All previous fields encrypted using symmetric key shared between originating node of the message and first node in truncated hop.

(c) Update/approve symmetric key mobile agent:
- Update ID.
- Public key.
- New symmetric key encrypted using public key (in case approve only).
- State {update/approve}.
- MAC (message authentication code): all previous fields encrypted with shared symmetric key (old key in update case and new key in approve case).

5. Protocol evaluation

To evaluate the proposed DSR with mobile agents' protocol, a new simulator has been developed by the author using Microsoft .Net technology. To evaluate the trustworthiness of the simulator, results obtained have been compared to a trusted simulator NS-2. The NS-2 [7] has been used extensively in evaluating the performance of ad hoc network routing protocols. We did not use NS-2 since it does not support mobile agents.

Each node in the simulation moves according to the random waypoint model [8] which means that a node starts at a random position chooses a target location and moves to this target location with a randomly chosen velocity, stops for a duration called pause time and repeats the process. We used a rectangular space of size 1500 m × 300 m to increase the average number of hops in the routes used relative to a square space of equal area, creating a more challenging environment for the routing protocol in this respect.

The DSR in NS-2 (DSRNS), DSR in the simulator developed by the author (DSRD), and DSR with mobile agents (DSRM) (also in the simulator developed by the author) were run with different pause times (0, 30, 60, 120, 300, 600, 900 s) to evaluate each of them. For each run four metrics have been computed:

1. Packet delivery ratio; representing the ratio of data packets received at its destination (Fig. 9).
2. Average end-to-end delay; representing the average time taken for data packet before it was received at its destination.
3. Packet overhead; representing the number of transmissions of control routing packets (e.g., a route request sent over three hops would count as three packets).
4. Path optimality; compares the length of routes used to the optimal (minimum possible) hop length as determined by an off-line algorithm.

After giving a quick look to the four charts (Figs. 9–12), we can see the similarity between DSRNS curves and DSRD.

Fig. 9, shows the packet delivery ratio for DSRNS, DSRD, and DSRM. DSRD outperforms DSRM by an average of 0.998% which is a small difference that is because; the only

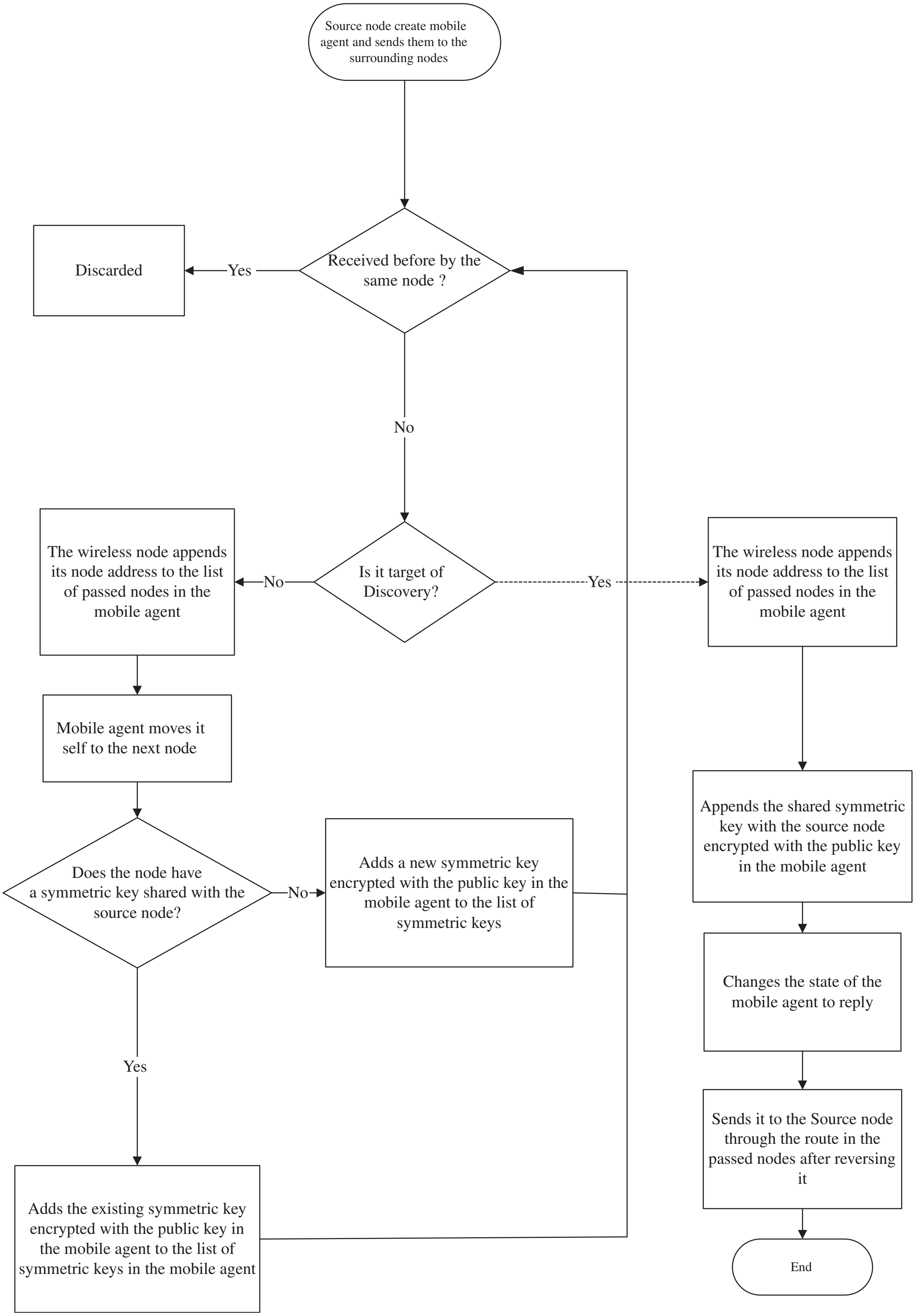
Source node create mobile agent and sends them to the surrounding nodes
Received before by the same node ?
Yes
Discarded
No
Is it target of Discovery?
No
The wireless node appends its node address to the list of passed nodes in the mobile agent
Yes
The wireless node appends its node address to the list of passed nodes in the mobile agent
Mobile agent moves it self to the next node
Does the node have a symmetric key shared with the source node?
No
Adds a new symmetric key encrypted with the public key in the mobile agent to the list of symmetric keys
Yes
Adds the existing symmetric key encrypted with the public key in the mobile agent to the list of symmetric keys in the mobile agent
Appends the shared symmetric key with the source node encrypted with the public key in the mobile agent
Changes the state of the mobile agent to reply
Sends it to the Source node through the route in the passed nodes after reversing it
End

Figure 8 Flow chart for the new proposed protocol.

added packets are for maintaining a symmetric encryption key between data packet sender and receiver.

Fig. 10, presents the average end-to-end delay for DSRNS, DSRD, and DSRM. DSRD outperforms DSRM by an aver-

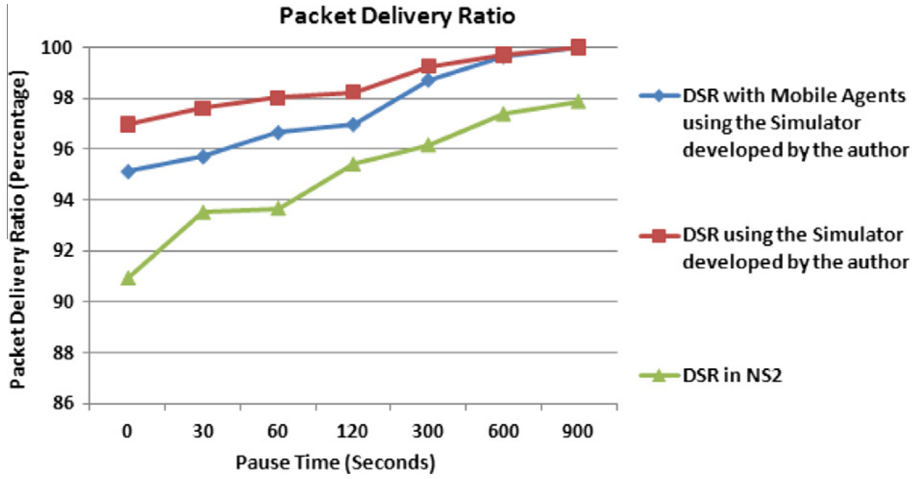

Figure 9 Packet delivery ratio.

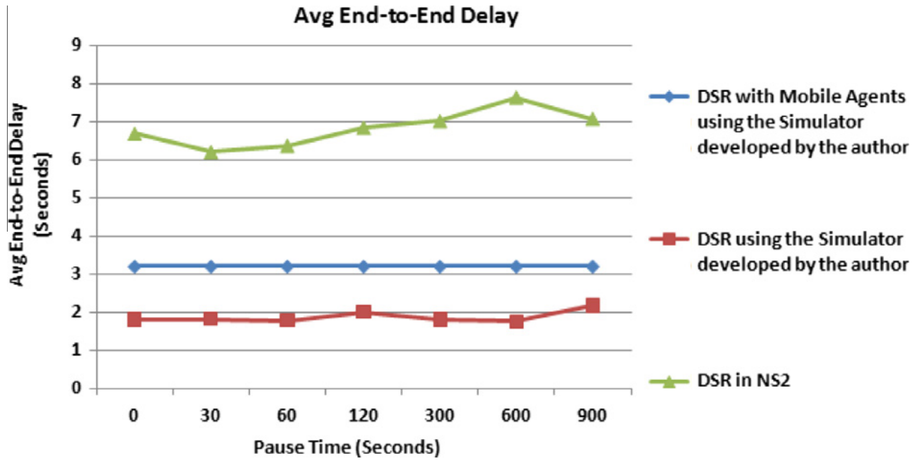

Figure 10 Average end-to-end delivery.

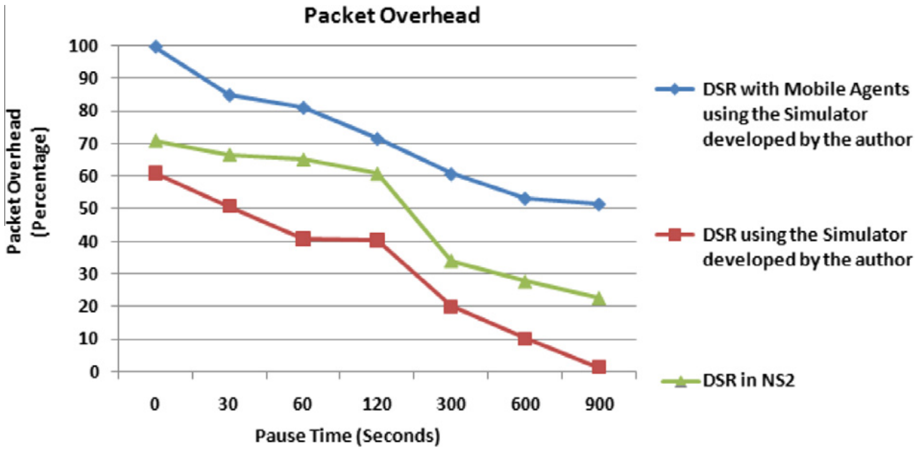

Figure 11 Packet overhead.

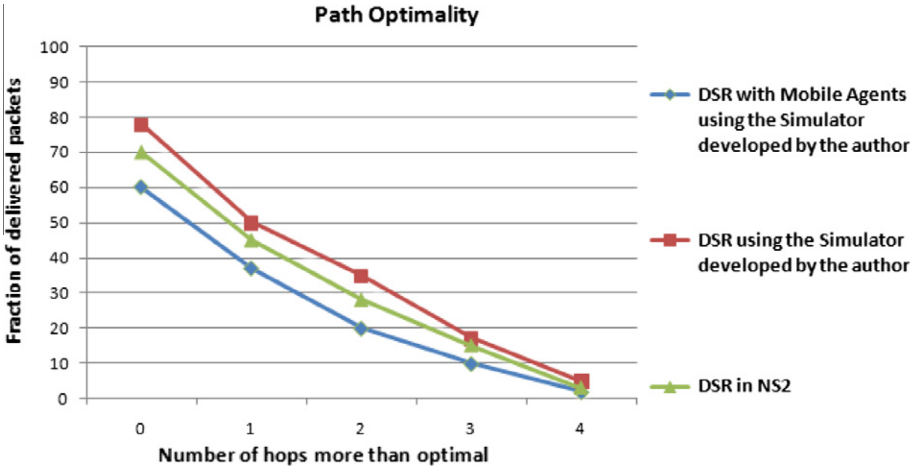

Figure 12 Path optimality.

age of 1.327 s which one can see that they differ slightly. This is due to the time taken by nodes to encrypt/decrypt data packets' content.

Fig. 11, shows the packet overhead for DSRNS, DSRD, and DSRM. DSRD outperforms DSRM by an average of 39.787% this is due to the encryption overhead.

Fig. 12, shows the path optimality for DSRNS, DSRD, and DSRM. DSRD outperforms DSRM by 8% when the number of hops is equal to the optimal number of hops, this is due to the time delay caused by encryption overhead.

Results show that there is a performance overhead for DSRM over DSRD which was expected because of the used mobile agents which affects packets delivery ratio and packet overhead. Also, time needed for the encryption/decryption of routing headers which affects packet delivery ratio and path optimality.

6. Conclusion

We have designed and implemented a simulator similar to NS-2, because, current release of NS-2 does not support mobile agents while our protocol uses mobile agents. We compared results obtained from our simulator to results obtained from NS-2 to trust the results obtained from our simulator. Also, we have added mobile agents to DSR to design and develop a new protocol to secure DSR using mobile agents.

We compared results obtained from NS-2 and simulator developed by the author for (DSRNS), DSR in the simulator developed by the author (DSRD), and DSR with mobile agents (DSRM) (also in the simulator developed by the author.) We found that for packet delivery ratio DSRD outperforms DSRM by an average of 0.998%, for the average end-to-end delay DSRD outperforms DSRM by an average of 1.327 s, for the packet overhead DSRD outperforms DSRM by an average of 39.787%, for the path optimality DSRD outperforms DSRM by 8%. Results show that there is a performance overhead for DSRM over DSRD which was expected because of the used mobile agents which affect packets delivery ratio and packet overhead. Also, time needed for the encryption/decryption of routing headers which affects packet delivery ratio and path optimality.

In future, mobile agents may be added to NS2 so that implementing a new simulator is not required. Also, mobile agents may be applied to protocols other than DSR, like AODV (ad hoc on demand distance vector), and TORA (temporally ordered routing algorithm.)

References

[1] Hong X, Kong J, Gerla M. Mobility changes anonymity: new passive threats in mobile ad hoc networks. Wireless Netw Secur 2006;6(3):281–93.

[2] Wikipedia article about Ad hoc wireless networks <http://en.wikipedia.org/wiki/Ad_hoc_routing_protocol_list>.

[3] Papageorgiou C, Kokkinos P, Varvarigos E. Implementing distributed multicost routing in mobile ad hoc networks using DSR. In Proceedings of the 6th ACM international symposium on mobility management and wireless access; 2008. p. 35–42.

[4] Stamos J, Gifford D. Remote evaluation. ACM Trans Program Lang Syst (TOPLAS) 1990;12(4):537–64.

[5] Wikipedia article about RPC. <http://en.wikipedia.org/wiki/Remote_procedure_call>.

[7] NS-2 simulator. <http://www.isi.edu/nsnam/ns/>.

[8] Johnson D, Maltz D. Dynamic source routing in ad hoc wireless networks. In: Imielinski Tomasz, Korth Hank, editors. Mobile computing. Kluwer Academic Publishers; 1996. p. 153–81 [Chapter 5].

Further reading

[6] Khemakhem M, BenAbdallah H, Belghith A. Towards an agent based framework for the design of secure web services. In Proceedings of the 2008 ACM workshop on secure web services; October 2008. p. 81–6.