Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Towards a reputation-based routing protocol to contrast blackholes in a delay tolerant network

Gianluca Dini, Angelica Lo Duca *

Dept. of Ingegneria della Informazione, University of Pisa, L.go Lazzarino 1, I-56126 Pisa, Italy

ARTICLE INFO

Article history:
Received 4 September 2011
Received in revised form 17 February 2012
Accepted 15 March 2012
Available online xxxx

Keywords:
Delay tolerant network
Reputation
Blackhole

ABSTRACT

A Delay Tolerant Network (DTN) relies on the implicit assumption that nodes cooperate towards message forwarding. However, this assumption cannot be satisfied when there are malicious nodes acting as *blackholes* and voluntarily attracting and dropping messages.

In this paper we propose a *reputation-based* protocol for contrasting blackholes. Every node *locally* maintains the reputation of forwarding nodes it comes in touch with and, then, upon selecting the next forwarding node, the node chooses among those having the highest reputation. The proposed reputation protocol is composed of three basic mechanisms—*acknowledgments*, *node lists*, and *aging*—that make communication efficient and capable of adapting to the changing operating conditions of a DTN.

The protocol has been used to extend CAR [1]. The resulting protocol RCAR (*reputation-based CAR*) has been compared with T-ProPHET [2], a state-of-the-art reputation-based DTN routing protocol, from several standpoints. As it turns out, RCAR is more effective than T-ProPHET and outperforms it in most cases.

1. Introduction

A Delay Tolerant Network (DTN) is a network paradigm where connectivity among mobile nodes is not always guaranteed. In order to guarantee messages delivery even in presence of network partitions, a DTN defines specific routing mechanisms to forward messages. The most common are *epidemic-based* [3] and *probability-based* [1,4]. In the *epidemic-based* mechanisms, many replicas of the same message are transmitted in the hope that at least one reaches the receiver. This mechanism is very expensive in terms of employed resources and it is not applicable when the nodes have limited resources (e.g., mobile nodes when they are battery operated). In the *probability-based* mechanism, the sender forwards the message to the node having the highest probability of successful message delivery. This mechanism relies on the implicit assumption that all the nodes cooperate to message forwarding. Unfortunately, malicious nodes may misbehave and act against the routing mechanism in different ways [5,6]. In this paper we deal with a specific malicious misbehavior according to which malicious nodes called *sinkholes* send wrong routing information to attract the largest possible number of messages. This *sinkhole attack* may be preparatory to other kinds of attacks, such as dropping all the attracted messages or dropping only some of them. In the former case the sinkhole acts as a *blackhole*, whereas in the latter as a *selective forwarder* [7].

In the paper we face with the problem of reducing the impact of the presence of blackholes in a DTN and propose an approach based on the concept of *reputation*. In short, upon selecting the next forwarding node, i.e., the node to forward a message to, a node estimates how well a candidate forwarding node has behaved on the basis of past interactions with that possible forwarding node. We call reputation such an estimation. In practice the reputation measures the trustworthiness of a node. The lower the reputation of a node, the higher the chance that the node is a

* Corresponding author.
E-mail addresses: dini@iet.unipi.it (G. Dini), loduca@iet.unipi.it (A. Lo Duca).

1570-8705/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2012.03.003

blackhole. The chances of selecting a node as a forwarding node are weighted by means of its reputation. Therefore a node having a low reputation is unlikely chosen as a forwarding node.

In our approach every node maintains a *local* notion of reputation. We make this choice in order to avoid the overhead and the technical complications related to maintaining a global shared notion of reputation [8]. Intuitively, reputation is maintained by means of three mechanisms, namely *acknowledgments*, *nodes list* and *aging*. Every message carries the list of forwarding nodes the message has traversed. Upon receiving the message, every node can update the reputation of the forwarding nodes specified in the list. Furthermore, upon handing a message to a forwarding node, the sender starts waiting for an acknowledgment from the ultimate destination. If the acknowledgment arrives, the sender increases the reputation of that forwarding node. Finally, reputations are periodically aged, i.e, decreased. The challenge here is to *adapt* the aging period to the highly changing delays of a DTN. We succeed in this by using a properly designed Kalman filter [9].

The reputation management mechanism employs both data messages and acknowledgment messages. In particular, the format of data messages has been extended to accommodate additional information concerning reputation management while an acknowledgment message is a unicast short message. It follows that, differently from other systems [2,10], the proposed reputation management protocol does not need to resort to communication expensive mechanisms for reputation updates dissemination based on broadcast/multicast communication.

The described reputation mechanism can be applied to all the DTN routing protocols using a probabilistic approach. In practice, it can be used whenever the next hop of a message is chosen as the one having the highest probability to deliver the message.

In this paper we apply the reputation-based approach to the Context Aware Routing (CAR) protocol, a probability-based routing protocol previously defined in [1]. CAR has no protection mechanisms against blackholes. We show that by augmenting CAR by means of the proposed reputation mechanism, the delivery ratio increases from 20% up to 60% according to the considered scenario.

We call *Reputation-based CAR* (RCAR) the resulting protocol. An early version of RCAR has already been proposed in [11]. With respect to that version, here we propose the following enhancements: (a) we protect the integrity of the reputation management information carried by messages; (b) we add a mechanism to dynamically determine the reputation aging period; (c) we compare the performance of RCAR with that of T-ProPHET from several viewpoints including delivery ratio, attraction ratio, and delivery delay. T-ProPHET is another reputation-based protocol for DTN [2] that we have chosen for comparison with RCAR because it is both quite recent and the most similar to RCAR. Performance evaluation tests show that in the best case RCAR is able to reach a delivery ratio of about 80% while T-ProPHET in the best case reaches a delivery ratio of about 70%. Furthermore RCAR works better than T-ProPHET when the number of blackholes in the network is low. In any case, RCAR is *more effective* than T-ProPHET, that is, the improvement that RCAR makes to CAR always outperforms the improvement that T-ProPHET makes to ProPHET.

The paper is organized as follows. Section 2 describes related works. Section 3 describes the network assumptions whereas Section 4 describes RCAR. Section 5 reports performance evaluation by means of simulation. Finally, Section 6 describes our conclusions and future work.

2. Related work

Secure cooperation in DTNs is quite a recent research challenge. However, a lot of work addressing the different aspects of secure cooperation has been produced [5,6]. The most common threats to DTNs are: *selfish nodes* and *blackhole nodes*. A selfish node is a node which is reluctant to cooperate in message forwarding in order to save its own energy. At the state of art, the most important protocols acting against selfish nodes try to reduce the selfishness by incentivating the selfish nodes to cooperate [12–14].

A blackhole is a node which drops all the received messages. At the state of art, the most important protocols acting against blackholes can be classified as: (a) *reputation-based*, (b) *reference-based*, and, finally, (c) *replication-based*. In *reputation-based systems* [8,2], each node observes the behavior of other nodes and assigns each of them a reputation which measures how well a node is behaving. The routing of messages is done on the basis of the reputation: the lower the reputation the lower the probability that a node is chosen as next hop (forwarding node) for a message.

In *reference-based systems* [15–17], each node wanting to forward a packet gives its *references* to its neighbors. A reference is a piece of evidence specifying that a node has cooperated to message forwarding. On the basis of references of a given node *j*, a node *i* can decide whether to forward its messages to *j* or not. In *replication-based systems* [10,18], secure cooperation is achieved by sending many replicas of the same message.

In the Secure Reputation-based Dynamic Window Scheme (SReD) [8], messages are forwarded to nodes having the highest reputation. The basic idea of SReD is to provide recommandations to each node based on the opinions of the other nodes. In practice, if a node *a* wants to calculate the reputation of a node *b*, it asks its neighbors, except node *b*, to give their opinion. Then, *a* calculates the reputation of *b* as the sum of its own opinion and the opinions given by the neighbors. In SReD reputations are shared among all nodes. The SReD protocol relies on the strong assumption that every node has a trusted hardware. Actually, in the absence of trusted hardware, a node could disseminate bogus recommandations. With respect to SReD, we propose a protocol in which the reputation is a local notion so that there is not the need of a trusted hardware.

In [2], the authors describe T-ProPHET, a reputation assisted data forwarding mechanism for opportunistic networks. Each node sends its messages to the node having the higher reputation. Upon receiving a message, a destination builds a *Positive Feedback Message* (PFM), which contains information to update the reputation of the carriers. The PFM is sent through epidemic routing, in order to reach more quickly the sender. When the sender receives the PFM, it can update the reputations of the carriers contained in it. The epidemic routing mechanism allows a sender to update reputations quickly, but it also adds traffic overhead. An extensive comparison with T-ProPHET is reported in Section 5.

Ren et al. propose a mechanism to detect blackhole nodes based on *packet exchange recording* [16]. When two nodes meet, they exchange their respective history records, containing the list of all node they have encountered. All the history records are authenticated through digital signatures. Comparing the other node's history with its own local history, each node is able to determine whether the other node has forwarded all the messages or not. This mechanism is called *sanity check*. If a node has not forwarded all the messages, the other nodes can detect it and classify that node as a blackhole. This mechanism is able to recognize a high number of blackhole nodes. However, this technique is not suitable when contact time between two nodes is short, because there may be no time to exchange long histories.

The same authors propose an improvement of the previous technique [17] where the sanity check is performed by a special node called *ferry node* rather than by every single node. Periodically, the ferry node visits all the nodes and asks them their history records. The ferry node classifies a node as a blackhole if the history of that node contains messages that the other nodes' histories do not contain. While this technique is very efficient, the ferry node features a single point of failure. If an adversary is able to compromise the ferry node, or take it down, all the system is compromised.

Chuah et al. propose a dynamic replication mechanism, where the number of generated message copies, called *redundant factor*, is calculated dynamically, on the basis of the current delivery ratio [10]. In particular, given a traffic flow originating in a sender and ending in a destination, this latter node measures the delivery ratio for the flow and sends it to the sender. The sender dynamically adjusts the redundancy factor according to the received delivery ratio. The main drawback of this protocol consists in the large number of messages caused by replication.

3. System model

We consider a Delay Tolerant Network (DTN) composed of several wireless nodes moving inside a fixed area. For instance, a node may be a device held by a human or embedded in a bus. In the network, messages are forwarded according to the following strategy. If a route already exists between the sender s and the receiver r, the message is forwarded using a standard routing protocol for ad hoc networks (e.g. DSDV). We call this mechanism *synchronous routing*. If the route does not exist, the sender uses an *asynchronous routing mechanism* according to which the message is forwarded to the forwarding node c having the highest chance of successful message delivery. The node c stores the message in its local buffer until it either establishes a route with the receiver r or encounters another forwarding node c' having a higher chance of message delivery to the destination. In the former case, c delivers the message to r by means of the synchronous routing. In the latter case, c applies again the asynchronous routing and forwards the message to c'. This routing process continues until the message eventually reaches its final destination r. Notice that a buffer has a limited size, therefore, when it gets full, the arrival of a new message causes a message loss. DTN mechanism inherently loses messages, unless the buffer size is unlimited. DTN management systems differ on buffer management and message replacement policies [19].

A crucial issue in asynchronous routing is how to select forwarding nodes [20]. Many solutions have been proposed [4,21,22]. In this paper we refer to the selection algorithm proposed by CAR (Context Aware Routing) [1]. In CAR, each node calculates its own *delivery probability*, i.e., the chance of successful message delivery on the basis of its own *context* information. A node context is defined as the set of *attributes* that describe the aspects of the system that can be used to drive the process of message delivery (e.g., mobility of node or battery level). Each node estimates its own delivery probability by means of a *Utility Function* $U(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i U_i(x_i)$ where x_i represents an attribute, $U_i(x_i)$ the Utility Function calculated over x_i, and w_i is the significance weight reflecting the relative importance of each attribute.

A node periodically computes an estimation of its own delivery probability. Then, the node broadcasts such an estimation to all the nodes it is able to reach through synchronous routing. The routing information necessary to build routing tables for synchronous routing is attached to the delivery probability estimation. CAR employs DSDV as synchronous routing protocol. In CAR, a node chooses the forwarding node among those reachable through the synchronous routing, and selects the one having the highest chance of message delivery, i.e., the node having the greatest value of U.

3.1. Adversary model

A DTN could be affected by many threats [23,18]. In this paper we assume that a node under the adversary control may behave as a *blackhole* [7]. This means that the malicious node strives to appear attractive for the other nodes as far as the routing algorithm is concerned. Then, upon receiving messages to forward—both synchronously or asynchronously—the blackhole drops them. In order to surreptitiously increase its own attractiveness, a blackhole may act in two ways, either modifying routing information in transit or exploiting the CAR mechanism for forwarding node selection. In the latter case, the blackhole could surreptitiously disseminate falsely high values of the Utility Function ($U \to 1$), so inducing other nodes to select it with high probability. We assume that blackholes do not collude.

Incidentally, we would like to point out that blackholes and *selfish nodes* have opposite behaviors. A selfish node is a node that uses the routing service but does not want to spend its own resources to cooperate towards that service

[7]. Therefore, in contrast to blackholes, selfish nodes strive to appear unattractive for the other nodes in order not to be selected as forwarding nodes. However, if selected, they will forward messages. In CAR, selfish nodes disseminate falsely small values of the Utility Function ($U \rightarrow 0$). It follows that countermeasures against selfish nodes tend to be very different from those against blackholes. In this paper we will not deal with selfish nodes and will not mention them any further. Countermeasures against them will be the subject of future work.

4. RCAR

In this section we describe RCAR (reputation-based CAR), an extension of CAR based on the concept of *reputation* and able to limit the effect of the presence of blackholes.

In Section 4.1 we introduce the concept of reputation in CAR. Then in Sections 4.2 and 4.3, we give some intuitions about its implementation and how nodes reputation gets updated.

4.1. The concept of reputation

Every node estimates how well another nodes behaves regarding the forwarding of messages. We call *reputation* (R) such an estimation. The range of R is $[0,1]$. The lower R, the higher the probability that the node is a blackhole. R is a local notion because it is calculated by each node on the basis of its own network experience. In other words, there is no global consensus on the reputation of a given node. This is in order to save the node energy and avoid both the traffic overhead and the technical complications due to the achievement of such a consensus. By R_{ij} we denote the *reputation* of node i calculated by node j. Every node j calculates the *reputation* R_{ij} of every node i it meets, as described below.

A node uses *reputation* to contrast blackholes as follows.

Definition 1 (*Local Utility Function*). Let U_i be the Utility Function of i and R_{ij} be the reputation of node i at node j, then the *Local Utility Function*, L_{ij}, is given by:

$$L_{ij} = R_{ij} \times U_i \tag{1}$$

Intuitively L_{ij} represents how capable of forwarding messages node j considers node i. Node j uses the local utility function to choose a node. In practice, it chooses the node i having the highest value of L_{ij} as the forwarder of a message. The rational basis of this choice is the following. Assume that a node i is a blackhole. Thus node j assigns a low reputation value to node i, i.e., $R_{ij} \rightarrow 0$. It follows that the value of $L_{ij} \rightarrow 0$ and thus j does not select i as a forwarding node.

More formally, let D be the event "node i delivers a message" and B the event "node i is not blackhole". The probability of successful message delivery $P(D)$ is given by the Bayes theorem:

$$P(D) = \frac{P(B)P(D|B)}{P(B|D)} \tag{2}$$

where $P(D|B) = U_i$, where U_i is the Utility Function of node i. This is because if a node is not blackhole (event $P(B) = 1$), the event D happens with a probability given by the chances of the node to forward a message (i.e. U_i). Furthermore, $P(B) = R_{ij}$, where R_{ij} is the reputation given by the node j to the node i. This is because a node is not blackhole with a probability equal to its reputation. Thus we have:

$$P(D) = \frac{R_{ij}U_i}{P(B|D)} \tag{3}$$

but $P(B|D) = 1$ because if i forwards messages, i is not blackhole. Therefore,

$$P(D) = U_i \times R_{ij} \tag{4}$$

where $P(D) = L_{ij}$.

Fig. 1 shows the flow diagram of the algorithm used by RCAR to select a forwarding node. Assume that a node s, the sender, sends a message m to a node r, the receiver. If a route exists between s and r, the node s exploits the synchronous routing, otherwise it exploits the asynchronous routing as described in Section 3. When the synchronous routing is available, the node s selects the next hop n to reach r according to the DSDV protocol, as in CAR. Once selected the next hop n, the node s checks whether $L_{sn} > 0$. If this is the case, it means that the node n is not blackhole, so that the node s sends it the message m. Otherwise, if $L_{sn} = 0$, the node s tries to exploit the asynchronous routing. As already said, the asynchronous routing is used also when a route from the sender to the receiver does not exist. In the case of asynchronous routing, the sender s selects the node c having the highest L_{sc}. In order to reach the node c, the node s exploits the synchronous routing, that is, it selects the next hop n according to the DSDV protocol, in order to reach c. Once received the message m, the node c stores it in its local buffer. As before, node s sends the message m to n only if $L_{sn} > 0$, otherwise it stores m in its local buffer. Periodically both the nodes s and c try to send the messages contained in their local buffer by repeating the previously described mechanism.

We have made the choice that node j forwards a message to a given next hop n if and only if $L_{nj} > 0$ because at the same time we want both to contrast blackholes and preserve the advantages introduced by the use of synchronous routing.

Note that when the synchronous routing is used, a node s sends the message to a node n only if $L_{sn} > 0$, while when the asynchronous routing is used, the node s selects the node n having the largest value of L_{sn}. This is due to the fact that in the synchronous routing the Utility Function of the node n does not influence the routing so that we can assume that $U_{sn} = 1$. This means that the value of L_{sn} depends only on the reputation. In this case, the message must not be sent to a node if it is a blackhole. The node n is blackhole if $R_{sn} \leqslant \theta$, where θ is a threshold to consider a node blackhole. Without loosing in generality, we can assume $\theta = 0$, because if $R_{sn} = 0$ the node n is certainly blackhole. In contrast, in the asynchronous routing, $0 \leqslant U_{sn} \leqslant 1$ and thus the value of L_{sn} depends also on it. This means that it is not sufficient to check whether $L_{sn} > 0$ to select the forwarding node, because the protocol must also take into account the node having the largest Utility Function.

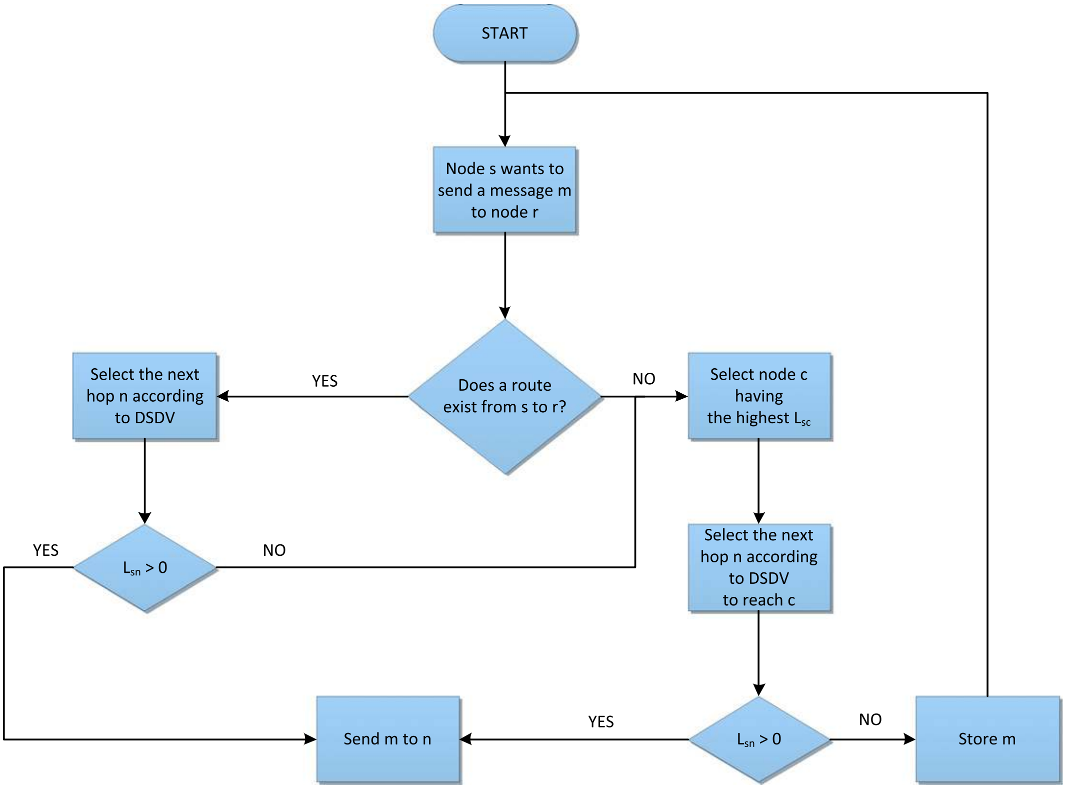
START
Node s wants to send a message m to node r
Does a route exist from s to r?
YES
NO
Select the next hop n according to DSDV
Select node c having the highest L$_{sc}$
L$_{sn}$ > 0
YES
NO
Select the next hop n according to DSDV to reach c
Send m to n
YES
L$_{sn}$ > 0
NO
Store m

Fig. 1. Flow diagram of the RCAR algorithm.

4.2. The reputation update protocol

The *Reputation Update Protocol* (RUP) is the mechanism used by RCAR to update reputations of nodes. Integrity and authenticity of RUP information is protected by means of digital signatures. In particular, we assume that every node has a pair (public key, private key) and a certificate binding its identifier to its public key signed by a Certification Authority (CA) trusted by all the nodes. By $\langle x \rangle_a$ we indicate the digital signature of node a on quantity x. Deploying a Public Key Infrastructure (PKI) is in general a problematic task due to the problems connected to certificate revocation and, more in general, the need of an online CA. Solutions have been proposed in [24–27]. However, at the state of art, PKI is the common solution used in DTNs to authenticate messages and provide their integrity [2, 14,15].

The basic idea behind the RUP is based on the following observation. If a node d receives a message, then all nodes the message passed through are well behaving or, otherwise, d would have not received the message. This means that, upon receiving a message, node d can increase the reputation of all nodes the message passed through, provided we can keep track of them.

We keep track of nodes a message has passed through as follows. Every message m carries the list of identifiers of nodes the message has passed through. We call *node list* (*nlist*) such a list. Upon receiving message m, a node adds itself to the *nlist*. A node adds itself only once, even though a message passes through that node many times. A malicious node could modify the *nlist* and add identifiers of nodes the message has not passed through in order to increase the reputation of other malicious nodes. In order to avoid such modifications, the message carries also a list of digital signatures $\sigma_1, \sigma_2, \ldots, \sigma_{i-1}$ that prove that the message has actually passed through the nodes specified in the node list. We call *slist* such a list. The digital signature σ_i establishes an unbreakable link between the node c_i receiving m and the node c_{i-1} from which it has received m. In practice through σ_i we are sure that the message m has gone from c_{i-1} to c_i without passing throughout any other intermediate node.

The length of the *slist* influences both the message length and thus the RCAR communication overhead as well as the message processing time and thus the RCAR processing overhead. However, as we will show in Section 5, the average number of nodes a message passes through is small, namely about 1.5 on average. This means that the overhead added by the digital signatures is practically negligible.

Let us suppose that a sender s sends a message $m = (mid, p, d, t_s, nlist, slist)$ where *mid* is the unique identifier associated to each message, p is the message payload, d is the destination node and t_s is the time at which the message is sent. Initially *nlist* and *slist* are empty. They are iteratively constructed as follows. Suppose that the forwarding node c_1 receives the message from s. It updates the *nlist* and the *slist* as follows:

$$nlist = c_1 \tag{5}$$
$$slist = \sigma_1 \tag{6}$$

In the most general case, node c_1 forwards the message *m*. When *m* passes through the *i*th forwarding node c_i, $i > 0$, let us assume that $nlist = \{c_1, c_2, \ldots, c_{i-1}\}$ and $slist = \{\sigma_1, \sigma_2, \ldots, \sigma_{i-1}\}$. Then, c_i updates *nlist* and *slist* as follows:

$$nlist \leftarrow nlist \| c_i \tag{7}$$
$$slist \leftarrow slist \| \langle \sigma_{i-1}, c_i \rangle_{c_i} \tag{8}$$

where $\|$ indicates the append operation and $\langle x \rangle_{c_i}$ indicates the digital signature made by node c_i on content *x*.

Upon receiving message *m*, the receiver *d* verifies (i) the digital signatures contained in the *slist* and (ii) if the list of nodes contained in the *nlist* corresponds to that contained in the *slist*. If the check is successful, it extracts the list of nodes from *nlist* and increases the reputation of all these nodes.

The described protocol allows only the receiver *d* to update the reputation of the nodes. This basic mechanism can be improved, using two additional mechanisms: ack-based and step-by-step. With the *ack-based*, the destination node *d* builds an *acknowledgment message* $ack = (mid, t_s, clist, slist)$, and sends it back to the sender *s*. The *nlist* and *slist* of the acknowledgment are initialized with the *nlist* and the *slist* of the original message. Furthermore, during the ack forwarding process, the *nlist* and the *slist* are updated as described before for standard messages. The *ack* behaves as a standard message except it is not acknowledged in its turn. The *ack* message may follow a different route from the original message *m*. In practice, each node forwarding the *ack* adds itself in the *nlist* and *slist*, only if it is not already contained in them. Upon receiving the *ack* message, the original sender *s* verifies the digital signatures contained in the *slist* and the correspondence between the nodes contained in the *nlist* and the *slist*, and if this check is successful, it increases the reputation of the nodes contained in the *nlist*.

The *step-by-step* mechanism is an improvement of the previous one. All the nodes traversed by the message and the corresponding *ack* extract the corresponding *slist* and *nlist*, verify the digital signatures contained in the *slist* and the mutual consistency of *nlist* and *slist*. Then they update the reputation of the nodes contained in the *nlist*. As it turns out, the reputation update process involves only certain nodes, namely those receiving the original message and the *ack*. We have made this choice to keep low the DTN management traffic. In the performance evaluation tests described in Section 5, we have used the *step-by-step* technique.

4.3. The aging mechanism

The just described mechanism allows every node to increase the reputation of all the nodes contained in the *nlist* of a message that passes through the node. However, if a message gets lost, a node has no means to know whether a blackhole has dropped it or another situation has occurred (e.g. message dropped for buffer overflow or TTL elapsed). Furthermore, even though the node could know that a message has not been delivered, it could not know which node along the path has misbehaved. Thus, in order to cope with blackholes, a mechanism based on *aging* is used to decrease the reputation of all the nodes. In practice, periodically a node decreases the reputations of all the nodes. This choice is done to have a conservative policy, because a node does not know which node has dropped the message.

We assume that the reputation increases and decreases linearly. That is, reputation *R* may be increased by a positive non-zero quantity *X*, i.e. $R \leftarrow \max(1, R + X)$, or decreased by a positive non-zero quantity *Y*, i.e., $R \leftarrow \max(0, R - Y)$. Quantities *X* and *Y* may be different if we require that the reputation increases and decreases at different paces. In an optimistic policy, node *i* may initially consider *j* trusted, i.e., initially $R_{ji} = 1$. In contrast, in a conservative policy, node *i* may not initially trust *j* at all, i.e., initially $R_{ji} = 0$. Intermediate policies can be defined.

In the aging mechanism, it is important the choice of the value of the *decrease period T*. On one hand, a too large value of *T* generates a high number of false negatives, because reputation of blackholes is decreased too slowly. On the other hand, a too small value of *T*, instead, produces a high number of false positives, i.e. well-behaving nodes are classified as blackholes because their reputation decreases too quickly before acknowledgments come back to the sender.

In order to fulfil the requirements of a DTN, the decrease period *T* cannot be fixed, because DTN conditions change. The decrease period *T* could be expressed as a function of the Round Trip Time (RTT). Upon sending a message, the sender attaches the message a timestamp specifying the moment in which the message was originated. So doing, upon receiving the corresponding acknowledgment, the sender node is able to calculate the RTT of the message. This means that whenever a node receives an acknowledgment, it can update the decrease period. In this way the decrease period follows the RTT and is updated dynamically according to network conditions. However, due to the nature of DTN, an acknowledgment may get lost so that the decrease period is not updated correctly. In order to dynamically update the decrease period even when the RTT is not (always) available, we employ a mechanism to predict the future values of RTT on the basis of the past history. We use Kalman filters [9]. Kalman filters were originally thought for automatic control systems theory. They are able to estimate the next state of a dynamic system on the basis of some observations. The advantage of using Kalman filters derives from their ability to predict the next state even when the current observation is not available. Furthermore, they do not require to store the whole past history of the system so they can be used also by resource constrained vehicles.

In more details, by T_k and D_k we denote the *decrease period* and the RTT at the step *k* respectively. The RTT is a measured value (i.e. it comes from real observations), while the *decrease period* is an estimated value (i.e. it is calculated through the Kalman algorithm). The purpose of the Kalman filter is to calculate the value of the *decrease period* T_{k+1} at the step $k+1$ given T_k. The Kalman filter is composed of two phases: (a) the *predictor*, (b) the *corrector*. Fig. 2 shows the architecture of a Kalman filter. The predictor takes the current value of the decrease period (T_k) as input and estimates the next value T^-_{k+1} on the basis of its own algorithm as described in [9]. The notation T^- indicates that the value has not been compared with the measured values yet. The value T^-_{k+1} represents the next predicted value for the *decrease period*. However, it must be corrected with the values coming from real measures of RTT in order to avoid that it drifts away too far from the real value of RTT. For this reason, the predicted value T^-_{k+1} constitutes the input for the corrector component, which takes also the value of the RTT D_k as input and produces the final predicted value T_{k+1}. It is important to note that if the value of the RTT is not available at a given step, the corrector component can be ignored and the next predicted value of the decrease period coincides with T^-_{k+1}.

5. Performance evaluation

In this section we evaluate performance of RCAR via simulation. RCAR has been compared to T-ProPHET [2], a reputation-based protocol for DTN that is both very recent and the most similar to RCAR. Furthermore, we also compare RCAR to Epidemic Routing (ER) [3]. In brief, ER relies on the idea that both the sender and any forwarding node forward a message to all the nodes they meet until the message arrives at destination. However, in order to avoid overwhelming the network, a forwarding node never forwards a message twice.

In our evaluation we also compare the improvement introduced by RCAR on CAR [1] with the improvement introduced by T-ProPHET on ProPHET [4].

Both CAR and ProPHET are probabilistic routing protocols for DTNs. They both calculate the next hop of a message as the one having the highest delivery probability. Both in ProPHET and in CAR a node *a* calculates the delivery probability of a node *b* on the basis of the past encounters between the two nodes. However, if the past encounters are not available or they are too old, in CAR the node *a* locally forecasts the delivery probability of *b* by using particular forecasting techniques (i.e. Kalman filters). In ProPHET, instead, the node *a* asks its neighbors their past encounters with *b*. On the basis of the received information, the node *a* calculates the delivery probability for node *b*.

CAR and ProPHET have already been compared in [1]. In the absence of blackholes CAR outperforms ProPHET from several standpoints. In particular, CAR achieves a higher delivery ratio, a smaller number of sent messages and a smaller propagation delay than ProPHET. This is due to the fact that CAR has a more efficient mechanism to update delivery probabilities than ProPHET. However, as we are going to describe below, this update mechanism allows blackholes to attract a larger number of messages in CAR than in ProPHET. It follows that CAR is more susceptible to blackholes than ProPHET.

In order to make performance evaluations and comparisons, we have simulated RCAR, CAR and ER using the OMNet++ simulator, while for ProPHET and T-ProPHET we have taken results from [2]. In order to compare CAR, RCAR, ER, ProPHET and T-ProPHET in the same conditions, we have considered the same simulation scenarios as described in [2]. In particular, we have created blackholes and well-behaving nodes moving around a fixed area. Only well-behaving nodes send/receive messages while blackholes have been implemented as nodes distributing a delivery probability $U_b = 1$ for all the destinations. Once attracted a message, a blackhole drops it. In the case of ER, it does not employ delivery probability, so that a blackhole is a node which drops received messages.

We have set the number of well-behaving nodes to 20. We have performed two sets of simulations: (a) we have varied the network area size (1000 m × 1000 m, 2000 m × 2000 m and 3000 m × 3000 m), while keeping fixed the number of blackholes (6 blackholes), (b) we have varied the number of blackholes (2, 6, 10, 14) while keeping fixed the network area size (1000 m × 1000 m). Each node has a communication range equal to 25 m, with a variable speed from 0 to 5 m/s. Each node generates a message with

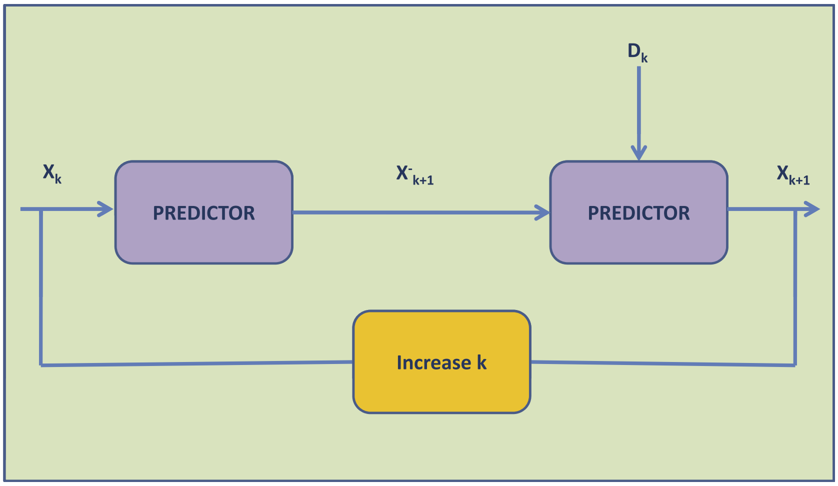

Fig. 2. The Kalman filter architecture.

a rate of 1 message/s and has a buffer size of 1000 messages. For each simulation we have performed 20 runs, each lasting 8900 s. In each run we have set the warm up period to 300 s. The warm up period is the initial time needed to build the routing tables for synchronous routing. We have set the routing table transmission interval (RTTI) to 3 s. The RTTI represents the interval of retransmission of the routing tables. In practice, each node retransmits its routing table to the other nodes every RTTI seconds. We have set the initial reputation to 1, the reputation increment X to 0.1, while the reputation decrement Y to the half of the reputation increment (i.e. 0.05). Table 1 resumes configuration parameters.

We have measured the following metrics:

- *Node List Length*. It is the number of nodes contained into an acknowledgment, when it arrives to the original sender of the message.
- *Delivery ratio*. It is the ratio between the number of received messages and the number of sent messages. Acknowledgments do not contribute to the delivery ratio. In practice, it measures the fraction of messages the network is able to deliver in the presence of blackholes. Ideally the delivery ratio should be equal to 1.
- *Attraction Ratio*. It is the ratio between the number of attracted data message by the blackholes and the number of sent data messages. Acknowledgments do not contribute to the attraction ratio, although blackholes attract acknowledgments too. In practice, the attraction ratio measures the fraction of messages the blackholes are able to attract. Ideally the attraction ratio should be equal to 0.
- *Average Delay*. It is the time between the generation of a message and its delivery to the final receiver. It is calculated as the average of all the messages received, included acknowledgments. Ideally the average delay should be equal to 0.
- *Total Number of Sent Messages*. It is the total number of messages sent in the network, including routing messages and acknowledgments.

Table 1
Configuration parameters.

Parameter	Value
Network area	1000 m × 1000 m, 2000 m × 2000 m, 3000 m × 3000 m
Number of well behaving nodes	20
Number of blackholes	2, 6, 10, 14
Communication range	25 m
Node speed	0–5 m/s
Data generation rate	1 message/s
Simulation time	8900 s
Warmup period	300 s
RTTI	3 s
Number of runs	20
Buffer size	1000 messages
Reputation increment	0.1
Reputation decrement	0.05
Initial reputation	1

5.1. Node List Length

Fig. 3 shows the distribution of the Node List Length vs. the network area sizes. Each tick on the x-axis represents the Node List Length, that is, the number of nodes contained in the acknowledgment. We note that for about the 50% of messages the node list contains 1.5 nodes and only sporadically the Node List Length is two or longer.

5.2. Delivery ratio

Fig. 4 shows the delivery ratio of RCAR, CAR, ER, ProPHET and T-ProPHET with respect to different network area sizes. In all the cases ER outperforms all the other protocols, paying the cost of a very high number of sent messages, as explained in Section 5.5. However, its delivery ratio decreases while increasing the network area size. This is due to the fact that when the network area size increases, the probability that two nodes meet is low so that their buffers are not emptied. This causes buffer overflow and many messages get lost.

In all the cases RCAR outperforms T-ProPHET. It is interesting to note that when the network area size is 1000 m × 1000 m and 2000 m × 2000 m, CAR reaches the lowest delivery ratio. This is due to the fact that when the network is small, in CAR the synchronous routing works, while ProPHET has not a synchronous routing. If the synchronous routing is very efficient when there are not blackholes [1], it does not work as expected in presence of blackholes. Actually, when the network is small, the probability to meet blackholes is so high that many messages are sent to them and consequently are dropped. When the network area size increases (3000 m × 3000 m), CAR outperforms both T-ProPHET and ProPHET. This is due to the fact that when the network area is large, CAR uses the asynchronous routing. As the network area is large, the probability to meet a blackhole is smaller so that the asynchronous routing works quite efficiently.

Fig. 5 shows the performance increase in delivery ratio of RCAR (T-ProPHET) on CAR (ProPHET) with respect to different network area sizes. The performance increase in delivery ratio is calculated as the difference between the delivery ratios of RCAR (T-ProPHET) and CAR (ProPHET), respectively. The greater the performance increase the greater the increase introduced by RCAR (T-ProPHET) on CAR (ProPHET). We note that in all cases the performance increase in delivery ratio introduced by RCAR on CAR is higher than the one introduced by T-ProPHET on ProPHET.

Fig. 6 shows the delivery ratio of RCAR, CAR, ER, ProPHET and T-ProPHET with respect to an increasing number of blackholes. ER outperforms all the other protocols, except in the scenario with 14 blackholes where it reaches a delivery ratio comparable to the one of RCAR and CAR and smaller than the one of T-ProPHET. This is due to the fact that when the number of blackholes increases, the probability that a node meets a blackhole is higher and consequently it is higher the probability that a node forwards replicas of messages to blackholes. As a result many messages are dropped and the delivery ratio decreases.

When the number of blackholes is small (2 and 6), RCAR outperforms all the other protocols. In contrast when the

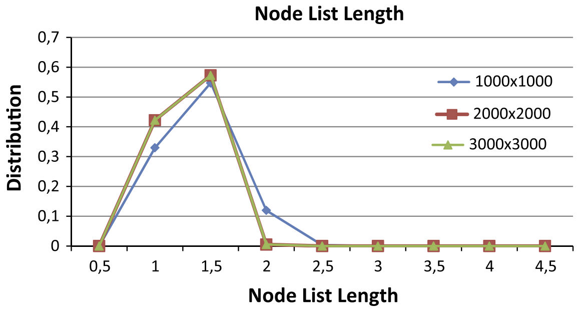

Fig. 3. Nodes List Length distribution.

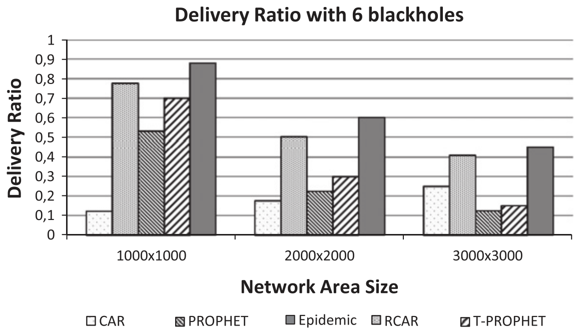

Fig. 4. Delivery ratio vs. network area size.

number of blackholes is high (10 and 14), T-ProPHET outperforms all the other protocols. However, in all cases, the performance increase in terms of delivery ratio introduced by RCAR on CAR is higher than the one introduced by T-ProPHET on ProPHET (Fig. 7). It is interesting to note that T-ProPHET seems to be independent on the number of blackholes, because the delivery ratio remains quite constant while varying the number of blackholes. This is due to the way it updates delivery probabilities, as described in Section 5.3.

5.3. Attraction ratio

Figs. 8 and 10 show the attraction ratio of RCAR, CAR, ProPHET and T-ProPHET with respect to an increasing network area size and an increasing number of blackholes respectively. We do not show the values for ER, because in ER a blackhole simply drops the messages it receives without attracting them. In ER, in fact, a blackhole has no means to attract a message, as the routing mechanism sends messages to all the nodes.

We note that the attraction ratio of CAR is higher than the one of ProPHET in all scenarios so that also RCAR attracts more messages than T-ProPHET. However this is due to the different mechanisms adopted by CAR and ProPHET to update delivery probabilities. In fact in CAR a node calculates another node delivery probability exactly as the delivery probability sent by that node. In particular, if a blackhole sends a delivery probability $U_b = 1$, every node i receiving it assumes that the blackhole has a delivery probability $U_{ib} = U_b = 1$. In ProPHET, instead, a node i calculates the delivery probability of a node j as the combination of three factors: (a) the previous delivery probability assigned by i to j, (b) the probability for i to meet j, which is calculated locally by i, (c) the delivery probability sent by j to i. This means that if a blackhole sends a delivery probability $U_b = 1$ to a node i, the node i calculates the delivery probability U_{ib} for that blackhole keeping into account the described factors so that in general $U_{ib} \leqslant U_b$. It follows that in CAR the probability to choose a blackhole is higher than in ProPHET.

It is interesting to note that when the network area size increases, the attraction ratio of CAR decreases, while the one of ProPHET increases. This is due to the different mechanisms used by CAR and ProPHET to update the delivery probabilities. When the network area size increases, the probability that a node meets a blackhole is lower. However, in ProPHET the lower the probability to meet a carrier, the higher the influence of the delivery probability sent by that carrier. In other words, when a blackhole sends a delivery probability $U_b = 1$ to a node i, node i calculates $U_{ib} \leqslant U_b$, if the network area size is small, whereas node i calculates $U_{ib} = U_b$ [4], if the network area size is large. In CAR, instead, the less the probability to meet a carrier, the less the probability that the carrier is chosen as forwarder.

Figs. 9 and 11show the attraction ratio increase of RCAR over CAR and T-ProPHET over ProPHET. The attraction ratio increase is calculated as the difference between the attraction ratio of CAR (ProPHET) and RCAR (T-ProPHET). The greater the attraction ratio increase the greater the improvement introduced by the RCAR (T-ProPHET) over

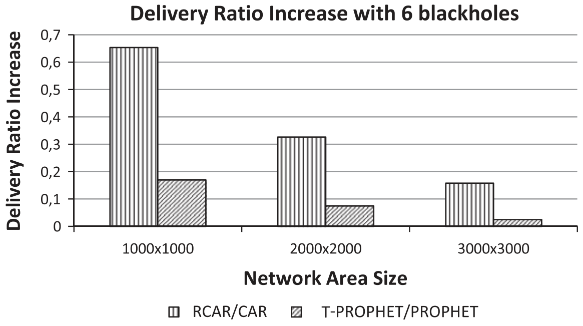

Fig. 5. Delivery ratio increase vs. network area size.

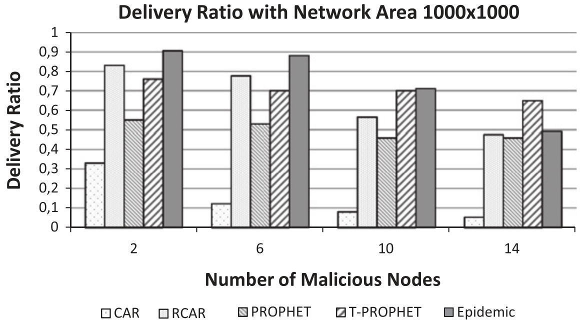

Fig. 6. Delivery ratio vs. number of malicious nodes.

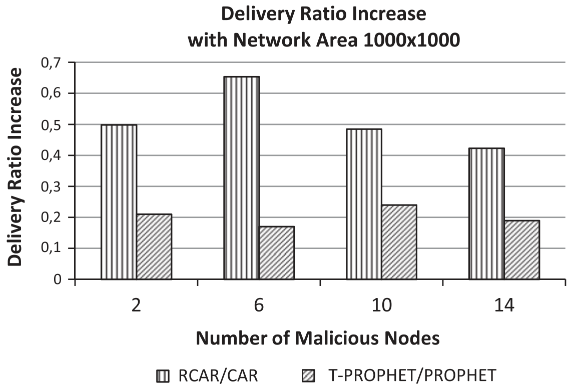

Fig. 7. Delivery ratio increase vs. number of malicious nodes.

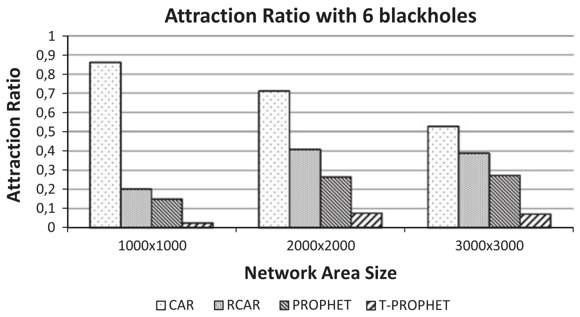

Fig. 8. Attraction ratio vs. network area.

CAR (ProPHET). RCAR outperforms T-ProPHET in the scenario with a small network area, while when the network area size increases T-ProPHET outperforms RCAR.

5.4. Average delay

Figs. 12 and 13 show the average delay of RCAR, CAR, ER, ProPHET and T-ProPHET with respect to an increasing network area size and an increasing number of blackholes, respectively. We note that ER reaches a smaller average delay than T-ProPHET and ProPHET and a greater average delay than RCAR and CAR. Furthermore, the average delay of ProPHET is higher than the one of CAR in all scenarios so that also the average delay of T-ProPHET is higher than the one of RCAR. The fact that the average delay of CAR is lower than the one of ProPHET has already been discussed in [1]. In practice, CAR employs also synchronous routing while ProPHET does not.

5.5. Total number of sent messages

Figs. 14 and 15 show the total number of sent messages in the network by RCAR, CAR and ER. We do not show the values for T-ProPHET and ProPHET, because the authors in [2] do not analyze this metric. We note that ER sends about 5.38×10^6 messages in the worst case (scenario with 6 blackholes and network area of 1000 m × 1000 m) and 1.046×10^6 messages in the best case (scenario with 6 blackholes and network area of 3000 m × 3000 m). RCAR instead sends about 1.01×10^6 messages in the worst case (scenario with 10 blackholes and network area of 1000 m × 1000 m) and 0.2×10^6 messages in the best case (scenario with 6 blackholes and network area of 3000 m × 3000 m).

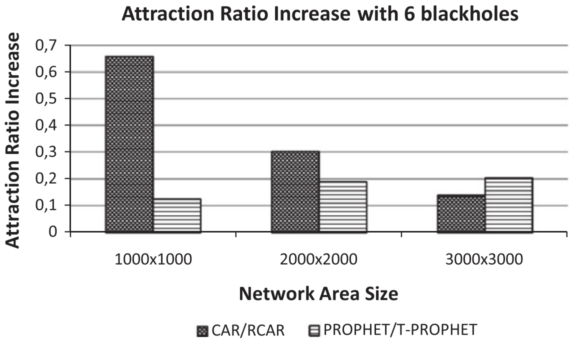

Fig. 9. Attraction ratio increase vs. network area.

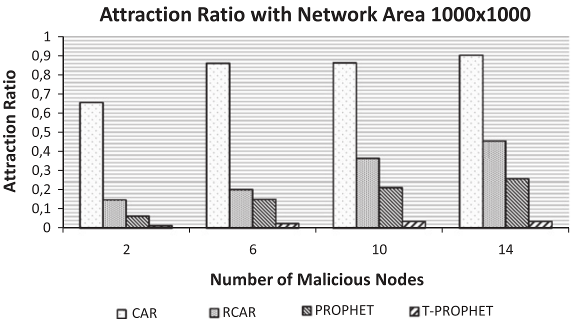

Fig. 10. Attraction ratio vs. number of malicious nodes.

We note that ER transmits a number of messages which is about five times greater than RCAR. This large overhead allows ER to outperform the other protocols in terms of delivery ratio. However, such a gain is very limited. For instance, in the case of 6 blackholes and a network size equal to 3000 m × 3000 m the delivery ratio of ER is 0.88 whereas that of RCAR is 0.77 (Fig. 14). It follows that from a practical point of view such an improvement of delivery ratio is not sufficient to justify the corresponding large communication overhead. For this reason, ER is not advisable in a DTN.

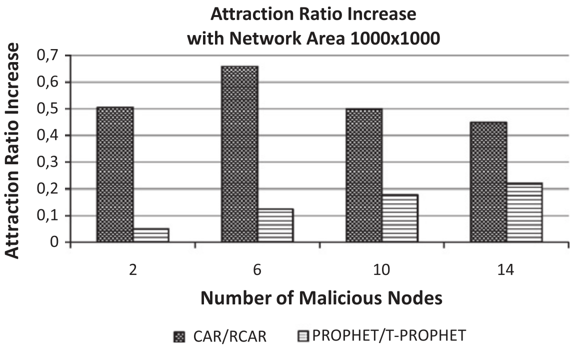

Fig. 11. Attraction ratio increase vs. number of malicious nodes.

Please cite this article in press as: G. Dini, A. Lo Duca, Towards a reputation-based routing protocol to contrast blackholes in a delay tolerant network, Ad Hoc Netw. (2012), http://dx.doi.org/10.1016/j.adhoc.2012.03.003

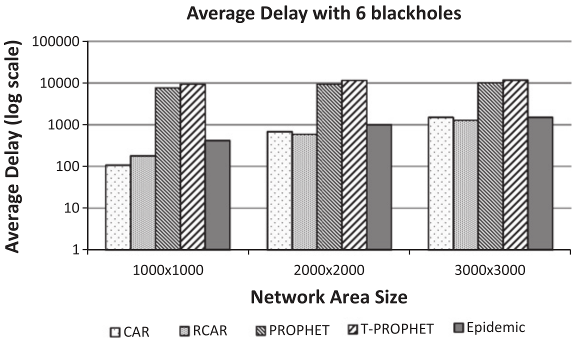

Fig. 12. Average delay vs. network area.

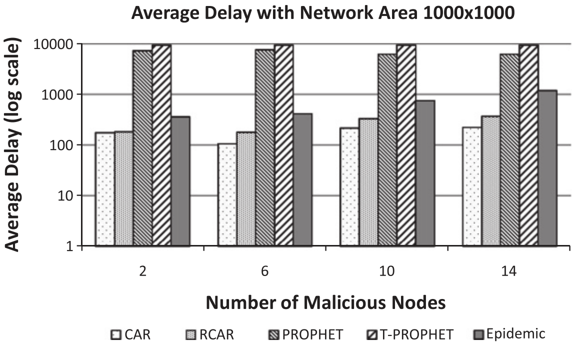

Fig. 13. Average delay vs. number of blackholes.

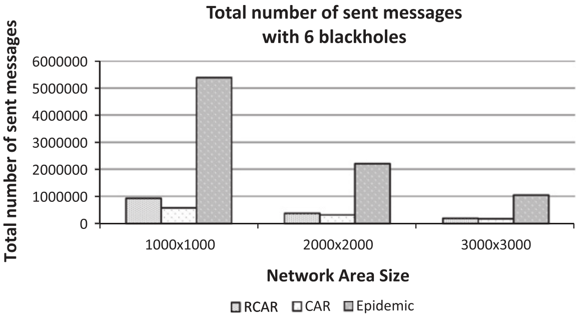

Fig. 14. Total number of sent messages vs. network area size.

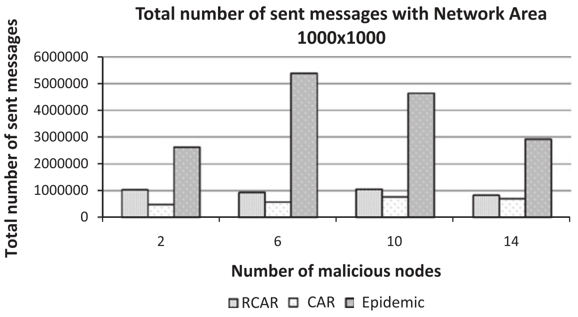

Fig. 15. Total number of sent messages vs. number of malicious nodes.

RCAR generates a greater number of messages than CAR because of the acknowledgment mechanism. The overhead in number of messages introduced by RCAR to CAR is 53.95% in the worst case (scenario with 2 blackholes and network area of 1000 m × 1000 m) and 12.36% in the best case (scenario with 6 blackholes and network area of 3000 m × 3000 m). The high overhead introduced by RCAR to CAR in the worst case is because in this scenario CAR has also a high Attraction Ratio (Fig. 10). This means that CAR delivers a low number of messages, because many of them are dropped by blackholes.

6. Conclusions

In this paper we have presented a reputation-based protocol to contrast blackholes in a DTN. The protocol has been integrated in CAR and the resulting RCAR protocol has been compared with T-ProPHET, a state-of-the-art reputation-based routing protocol deriving from ProPHET, from several viewpoints. As it turns out RCAR outperforms T-ProPHET in terms of delivery ration and average delay. Furthermore, the improvement of RCAR with respect to CAR always outperforms the improvement of T-ProPHET with respect to ProPHET. In addition to this, we believe that RCAR has the following merits.

- RCAR proves that an effective and efficient reputation-based routing protocol for DTN can be based on a *local* notion of reputation. As it turns out from Section 5, such a local notion allows us to effectively contrast blackholes without incurring in the overhead and the technical complications inherent to maintaining a global notion of reputation.
- Furthermore, RCAR has a reduced overhead. Actually, in RCAR the information for reputation management is properly integrated in data and acknowledgment messages. So, RCAR does not need to resort to expensive broadcast/multicast communication for reputation management. In addition, simulations show that the node list is short so making sustainable the related communication and computation overhead.
- Finally, RCAR shows that it is able to adapt to the highly changing conditions of a DTN. In particular RCAR is able to adapt its aging period so reducing the chance of false positives and false negatives.

Acknowledgments

We wish to thank the anonymous reviewers for their constructive criticisms and suggestions that helped us to improve the paper. We also wish to thank Dr. Cecilia Mascolo, University of Cambridge (UK), and Dr. Mirco Musolesi, University of St. Andrews (UK) who have provided us with CAR source code. Finally, we wish to thank Dr. Na Li who has given us useful information about T-ProPHET.

This work has been partially supported by the EU FP7 Project UAN (Grant Agreement No. FP7-2007-2013-225669), the EU FP7 Integrated Project PLANET (Grant agreement No. FP7-257649), and the EU FP7 Network of

Excellence CONET (Grant Agreement No. FP7-2007-2-224053).

References

[1] M. Musolesi, C. Mascolo, Car: context-aware adaptive routing for delay-tolerant mobile networks, IEEE Transactions on Mobile Computing 8 (2009) 246–260.

[2] N. Li, S.K. Das, A trust-based framework for data forwarding in opportunistic networks, Ad Hoc Networks, in press (Corrected Proof).

[3] A. Vahdat, D. Becker, Epidemic Routing for Partially Connected Ad Hoc Networks, Technical Report, Department of Computer Science, Duke University, 2000.

[4] A. Lindgren, A. Doria, O. Schelén, Probabilistic routing in intermittently connected networks, SIGMOBILE Mobile Computing Communication Review 7 (2003) 19–20.

[5] A. Seth, S. Keshav, Practical security for disconnected nodes, in: ICNP Workshop on Secure Network Protocols, (NPSec), Boston, Massachusetts, USA, 2005.

[6] N. Bhutta, G. Ansa, E. Johnson, N. Ahmad, M. Alsiyabi, H. Cruickshank, Security analysis for delay/disruption tolerant satellite and sensor networks, in: International Workshop on Satellite and Space Communications (IWSSC 2009), Siena, Italy, 2009.

[7] C. Karlof, D. Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, in: IEEE Internationl Workshop on Sensor Network Protocols and Applications, Anchorage, Alaska, USA, 2003, pp. 113–127.

[8] Z. Xu, Y. Jin, W. Shu, X. Liu, J. Luo, Sred: A secure reputation-based dynamic window scheme for disruption-tolerant networks, in: Military Communications Conference (MILCOM 2009), Boston, 2009, pp. 1–7.

[9] R.E. Kalman, A new approach to linear filtering and prediction problems, Transactions of the ASME Journal of Basic Engineering 82 (1960) 35–45.

[10] M. Chuah, P. Yang, Impact of selective dropping attacks on network coding performance in DTNS and a potential mitigation scheme, in: Computer Communications and Networks (ICCCN 2009), San Francisco, CA, 2009, pp. 1–6.

[11] G. Dini, A. Lo Duca, A reputation-based approach to tolerate misbehaving carriers in delay tolerant networks, in: IEEE International Symposium on Computers and Communications (ISCC'10), Riccione (IT), 2010.

[12] L. Buttyn, L. Dra, M. Flegyhzi, I. Vajda, Barter trade improves message delivery in opportunistic networks, Elsevier Ad Hoc Networks 8 (1) (2010) 1–14.

[13] I. Koukoutsidis, E. Jaho, I. Stavrakakis, Cooperative content retrieval in nomadic sensor networks, in: Infocom MObile Networking for Vehicular Environments Workshop (MOVE 2008), Phoenix, AZ.

[14] R. Lu, X. Lin, H. Zhu, X. Shen, B. Preiss, Pi: a practical incentive protocol for delay tolerant networks, IEEE Transactions on Wireless Communications 9 (2010) 1483–1493.

[15] F. Li, J. Wu, S. Avinash, Thwarting blackhole attacks in disruption-tolerant networks using encounter tickets, in: International Conference on Computer Communications (INFOCOM 2009), Rio de Janeiro, 2009, pp. 2428–2436.

[16] Y. Ren, M.C. Chuah, J. Yang, Y. Chen, Detecting blackhole attacks in disruption-tolerant networks through packet exchange recording, in: World of Wireless Mobile and Multimedia Networks (WoWMoM 2010), Montreal, QC, Canada, 2010, pp. 1–6.

[17] Y. Ren, M.C. Chuah, J. Yang, Y. Chen, Muton: Detecting malicious nodes in disruption-tolerant networks, in: Wireless Communications and Networking Conference (WCNC 2010), Sidney, NSW, 2010, pp. 1–6.

[18] J. Burgess, G.D. Bissias, M.D. Corner, B.N. Levine, Surviving attacks on disruption-tolerant networks without authentication, in: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc '07), Montreal, Quebec, Canada, 2007, pp. 61–70.

[19] A. Krifa, C. Baraka, T. Spyropoulos, Optimal buffer management policies for delay tolerant networks, in: Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2008), San Francisco, CA, 2008, pp. 260–268.

[20] S.Q. Ali, A. Junaid Baig, Routing protocols in delay tolerant networks – a survey, in: International Conference on Emerging Technologies (ICET 2010), Islamabad, Pakistan, 2010, pp. 70–75.

[21] J. Burgess, B. Gallagher, D. Jensen, B.N. Levine, Maxprop: Routing for vehicle-based disruption-tolerant networks, in: Conference on Computer Communications (INFOCOM 2006), Barcelona, Spain, 2006, pp. 1–11.

[22] A. Balasubramanian, B.N. Levine, A. Venkataramani, Dtn routing as a resource allocation problem, in: Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2007), Kyoto, Japan, pp. 373–384.

[23] L.-J. Chen, C.-L. Chiou, Y.-C. Chen, An evaluation of routing reliability in non-collaborative opportunistic networks, 2009, pp. 50–57.

[24] R.N. Wright, P.D. Lincoln, J.K. Millen, Dependent graphs: a method of fault-tolerant certificate distribution, Journal of Computer Security 9 (2001) 323–338.

[25] R.N. Wright, P.D. Lincoln, J.K. Millen, Efficient fault-tolerant certificate revocation, in: Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS '00), Athens, Greece, 2000, pp. 19–24.

[26] S. Capkun, L. Butty, J.-P. Hubaux, Self-organized public-key management for mobile ad hoc networks, IEEE Transactions on Mobile Computing 2 (2003) 52–64.

[27] J. Baek, J. Newmarch, R. Safavi-naini, W. Susilo, A survey of identity-based cryptography, in: Australian Unix Users Group Annual Conference (AUUG 2004), Melbourne, Australia, 2004, pp. 95–102.

Gianluca Dini received his "Laurea" degree in Electronics Engineering from the University of Pisa, Pisa, Italy, in 1990, and his Ph.D. in Computer Engineering from Scuola Superiore "S. Anna", Pisa, Italy, in 1995. From 1993 to 1994 he was Research Fellow at the Department of Computer Science of the University of Twente, The Netherlands. From 1993 to 1999 he was Assistant Professor at the Dipartimento di Ingegneria della Informazione, Elettronica, Informatica, Telecomunicazioni of the University of Pisa, where he is now Associate Professor. His research interests are in the field of distributed computing systems, with particular reference to security. Currently he is working on security in networked embedded systems. He has published more than eighty papers in international conferences, books and journals and has participated to many projects funded by the Commission of the European Community, the Italian Government and private companies.

Angelica Lo Duca is a Ph.D. student at Department of Ingegneria della Informazione of University of Pisa, Italy. She received her B.S. degree in Computer Science from University of Pisa, in 2007. Her current research interests include privacy and security issues in challenging networks, like Underwater Acoustic Networks (UANs) and Delay Tolerant Networks (DTNs).