L'Hospital: Self-healing Secure Routing for Mobile Ad-hoc Networks

Jiejun Kong, Xiaoyan Hong, Joon-Sang Park, Yunjung Yi, Mario Gerla
Department of Computer Science
University of California
Los Angeles, CA 90095
jkong@cs.ucla.edu, hxy@cs.ua.edu, {jspark,yjyi,gerla}@cs.ucla.edu

Abstract

Mobile ad hoc networks (MANETs) are vulnerable to a myriad of attacks. Purely cryptographic countermeasures are effective against outsiders, but not non-cooperative members including selfish and compromised members. Non-cryptographic means like intrusion detection must be devised to answer the challenge. However, as recently studied in "jellyfish attack" [1], the effectiveness of wireless intrusion detection systems (IDS) are limited when security attacks can be "cloaked" under protocol-compliant actions. For instance, since packet loss is common in mobile wireless networks, the adversary can exploit this fact by hiding its malicious intents using compliant packet losses that appear to be caused by environmental reasons.

In this paper we study two routing disruption attacks that use non-cooperative network members and disguised packet losses to deplete ad hoc network resources and to reduce ad hoc routing performance. These two routing attacks have not been fully studied in previous research. We propose the design of "*self-healing community*" to counter these two attacks. Our design exploits the redundancy in deployment which is typical of most ad hoc networks; namely, it counters non-cooperative attacks using the probabilistic presence of nearby "good" cooperative network members.

To realize the new paradigm, we devise localized simple schemes to maintain self-healing communities. The localized design virtually constructs a "*localized hospital*" (L'Hospital) to save the (optimal) route discovered by the underlying routing protocol. We develop an analytic model to prove the effectiveness of our design. Then we design and implement our secure ad hoc routing protocols in simulation to verify the cost and overhead incurred by maintaining the communities. Our study confirms that the community-based security is a cost-effective strategy to make off-the-shelf ad hoc routing protocols secure.

I. Introduction

A mobile ad hoc network (MANET) is an infrastructureless mobile network formed by a collection of peer nodes using wireless radio. It can establish an instant communication structure for civilian and military applications. Unfortunately, the mobility and radio broadcast medium make MANETs very vulnerable to malicious attacks. First, outsiders (or non-network members) can monitor the open wireless medium to intercept legitimate traffic or to inject illegitimate traffic. Fortunately, cryptographic schemes can protect the network from such external attacks. Second, some previously cooperative mobile nodes may turn selfish due to various reasons (like resource deprivation); or, some mobile nodes with inadequate physical protection may be captured and compromised. Purely cryptographic countermeasures are not effective against compromised or selfish members because cryptographic trust is rendered to whoever owns the cryptographic keys, independent of node's networking behavior. The network must rely on non-cryptographic means like intrusion detection systems (IDS) to cope with these non-cooperative (compromised or selfish) members. However, as studied in [1], the non-cooperative members may try to hide their attacks under protocol-compliant behaviors. In this case, behavior discrimination is not an effective countermeasure. For example, it is very hard to discriminate between losses caused by normal network and environmental conditions and those caused by selfish and malicious behaviors as they could all appear to be protocol-compliant.

In this paper our goal is to propose a new intrusion protection mechanism, namely *community-based security*, and evaluate its effectiveness in defending ad hoc routing protocols against non-cooperative nodes. The basic idea is

Part of the work is funded by MINUTEMAN project. Contract/grant sponsor: Office of Naval Research. Contract/grant number: N00014-01-C-0016.

to mitigate the adverse (albeit seemingly protocol-compliant) actions of selfish and malicious nodes by distributing the network service in question (e.g., packet forwarding) to a community of neighboring nodes. We will call such a community the localized *"self-healing community"*. At the node level, the service provisioning is untrustworthy and is allowed to be disrupted. However, at the community level, the service provisioning becomes trustworthy—even if some of the community members are selfish or malicious, the self-healing service remains available and reliable if there is at least one "good" cooperative community member that can provide the needed service.

The concept of "self-healing" is inherently consistent with *self-organizing* mobile ad hoc networks. Self-organization refers to the ability of an ad hoc network to construct and change its internal organization on its own account—neither in response to conditions in another system nor as a consequence of its membership in a larger metasystem.

In a self-organizing network, it is possible to realize *self-reconfigurability* which means the network actively reconfigures the arrangement of its constituent parts. In the MANET context, this implies that each ad hoc node must autonomously monitor and adapt its properties to environmental conditions (rather than rely on design-time pre-configuration and manual reconfiguration) in order to achieve a robust network in any environment.

In a self-reconfigurable network, *self-healing* means the network adapts automatically to defects and attacks in its node connectivity, service provisioning and performance disturbances to provide the best possible level of service to the communicating parties. In this paper, the term "self-healing routing" refers to a MANET's capability of achieving following three goals: (1) monitoring and detection of routing failures; (2) immediate healing of the failed routes to avoid significant disruption; and (3) fully distributed implementation without centralized system control.

Clearly, there are challenges in realizing such self-healing communities. In particular,

- *Community creation and configuration*: A self-healing community can be created and configured anywhere and anytime, but the related process should only incur reasonably low overhead.
- *Community reconfiguration*: The self-healing community must adapt to changes in the network topology and other dynamics. The impact of mobility, channel fluctuation, community member join and leave, and selfish and malicious nodes must be addressed and resolved.

In balance, the contributions of this paper are in two areas:

1) *Development of a new network security concept based on "self-healing community"*. We build self-healing routing communities between each source and destination pair. The conventional "per node forwarding" is replaced by the "community forwarding" concept: a chain of self-healing communities along the path will forward the packet, where each community is comprised of multiple peer members each of which can provide the needed packet forwarding service. The proposed scheme tolerates the presence of non-cooperative nodes and stops disruption attacks immediately. Such self-healing strategy has not been studied previously in the context of secure routing.
2) *Easy integration of community-based security with conventional routing schemes*. Our localized design realizes a *"localized hospital"* (L'Hospital) to save the (optimal) route discovered by the underlying routing protocol. In this paper we use the IETF standard AODV [23] as the example. We implement the new design in network simulators and study the new design's impact on the underlying routing protocol.

The rest of the paper is organized as follows. In Section II, we present security threats that have not been fully addressed by existing secure routing schemes. In Section III, we describe the concept of "self-healing community" as well as related self-configuration and self-reconfiguration protocols. An analytic model is presented in SectionIV to verify the effectiveness of community-based secure routing. Simulation results in Section V confirm the efficiency of community-based secure routing. In Section VI we compare our work with related work. Finally Section VII concludes this paper.

II. Problem: Routing Disruption

On-demand routing in MANET In this paper we apply the "community" concept to protect on-demand routing. Though the attacks and countermeasures are also applicable to proactive routing schemes, they are treated as future work for the ease of presentation. While proactive routing protocols exchange routing information even when there is no data transmission, the on demand approach pays the cost of routing overhead only when it is needed. An on-demand routing protocol is composed of two parts: *route discovery* and *route maintenance*. In route discovery, the source sends out a route request (RREQ) to the network when it needs a route to destination. A neighbor either

forwards the RREQ if it does not know the route to the destination, or sends back the needed routing information to the source. Upon receiving one or more RREQs, the destination sends back at least one route reply (RREP) to the source. Contrary to RREQ flooding, an RREP message is typically forwarded by a limited set of chosen forwarders, which are called *"RREP forwarders"* (or *"RREP nodes"*) in this paper. Although various on-demand routing protocols use different algorithms to process RREQ and RREP messages, the combination of RREQ and RREP processing establishes a route between the source and the destination. Due to mobility and network dynamics, an established route may be broken at any time. On-demand routing schemes use route error (RERR) notification to inform the source or the destination about the status. Then the source will initiate a new route discovery procedure to find new routes towards the destination. To overcome the overhead of a fresh restart from the source after each route outage, local recovery techniques are often applied (e.g., cached routes at neighbors are used when available).

Limitations of cryptographic protection Cryptography is an essential building block of network security. It relies on secrecy of keys, which are secret random variables maintained by each individual network member. Qualitative cryptographic algorithms ensure that any computationally bounded adversary cannot break the cryptosystem *if these secret keys are not compromised*.

However, in a self-organizing ad hoc network the power of cryptographic protection is limited. First, purely cryptographic solutions cannot answer the challenge imposed by non-cooperative (either compromised or selfish) network members. An autonomous ad hoc node who has earned cryptographic trust from other ad hoc nodes is not necessarily protocol-compliant, though it appears to be so. (1) If adequate physical protection cannot be guaranteed for *all* mobile nodes, then node compromise is inevitable in a long time window. After that, cryptosystems cannot differentiate compromised members from uncompromised ones. (2) Moreover, selfish nodes are uncompromised, but they will not provide the needed service. The network has to rely on non-cryptographic means like intrusion detection system (IDS) to cope with these non-cooperative members.

Second, as pointed out in [1], it is hard to differentiate various packet loss scenarios, for example, to identify those protocol-compliant cases caused by natural reasons (e.g., channel interference or node mobility) and those cases caused by non-cooperative behaviors (e.g., selfishness or maliciousness). A malicious sender can intentionally corrupt at least 1 random bit in the packet being transmitted, then it is hard for a good receiver to judge whether the corruption is caused by environmental reasons or otherwise. A malicious node can also selectively drop some critical packets, so that its packet loss pattern appears to be random as expected. In route discovery, a mobile node participated in RREQ forwarding may fail to forward RREP and data packets due to all kinds of reasons—random mobility, selfishness or maliciousness. There is no fail-safe method for loss discrimination between environmental reasons and non-cooperative behaviors.

Not fully-addressed ad hoc routing threats Although many secure ad hoc routing protocols [13][27][20][33][3] have been proposed to secure on-demand routing schemes, the following security attacks are not fully addressed in the existing proposals.

Attack 1: **(RREQ resource depletion)** A malicious node can attempt to deplete network resource by repeatedly initiating superfluous RREQ. In this attack, an attacker sends RREQ packets, which the underlying on-demand routing protocol floods throughout the network. If the attacker is not a network member, cryptographic authentication can be added to RREQ packets to filter out those forged route discovery requests. However, if the attacker is a compromised or selfish network member, non-cryptographic countermeasures must be used. □

A rate-limit approach is proposed in [13][23] to reduce number of RREQ packets each node is allowed to initiate. In this paper we seek to achieve this goal without compromising routing performance. Approaching the ideal case, where a routing protocol only incurs one initial RREQ flooding for each end-to-end connection, the community-based design significantly reduces the number of RREQ packets that each node initiates.

Attack 2: **(RREP and data packet loss)** A malicious or selfish node may cause the loss of certain critical packets. In a route discovery procedure initiated by a good network member, an attacker can use "wormhole attack" [13] or "rushing attack" [15] to surpass other nodes with respect to the underlying routing metric. Then when the RREP comes back it may not forward or may forward a corrupted one. The result is equivalent to RREQ resource depletion attack, except now the RREQ initiator is not the one to blame. In "jellyfish attack" [1], an attacker can severely degrade data delivery performance by selectively dropping data packets during certain periods of time. □

We will show how the proposed self-healing approach can counter all such attacks, including non-cooperative

RREP forwarders and data forwarders. When an RREP or data packet is lost, the damaged route is locally healed within minimal latency.

III. Community-based secure routing protocols

A. Network assumptions

At the routing layer, community-based security is applicable to a broad variety of routing schemes. Backward compatibility is one of our design goal. Given an underlying on-demand routing scheme (e.g., AODV [22], ARAN [27], DSR [16], Ariadne [13]), all original RREQ/RREP packet formats and packet forwarding requirements are accounted for in our design. This will make it possible to seamlessly integrate the proposed community-based paradigm with most existing ad hoc routing protocols.

At the link layer, *L'Hospital* assumes that a node can always monitor ongoing transmissions even if the node itself is not the intended receiver. This typically requires the network interface stay in the "receive mode" (i.e., promiscuous reception mode) during all transmissions, which is less energy efficient than listening only to packets directed to oneself. *L'Hospital* also assumes radio links are symmetric; that is, if a node X is in transmission range of some node Y, then Y is in transmission range of X. In an 802.11 style MAC protocol, this can be enforced by CSMA/CA using RTS/CTS handshake.

At the physical layer, transmissions are vulnerable to jamming. Fortunately, mechanisms like erasure coding, spread spectrum, and directional antenna have been extensively studied as means of improving resistance to jamming. In addition, jamming attackers are more easily to be detected and counter-attacked. In this work we consider packet loss attacks rather than jamming attacks. *L'Hospital* explores *physical* node redundancy in a self-organizing network as a method to stop route disruptions. *L'Hospital* assumes that in a network locality there are redundant network members with high probability. These peer members will have identical capabilities and responsibilities in community-based communication. No centralized control or hierarchical control is assumed.

B. Network security assumptions

L'Hostipal assumes every packet transmission is protected by data origin authentication service. *Every packet is authenticated and the packet sender's identity is unforgeable* (of course, when the packet sender is uncompromised). This can be implemented by signing each packet by the sender's digital signature or using efficient symmetric key protocols like TESLA [24][13]. Therefore, the adversary cannot forge packet transmissions from uncompromised nodes, and cannot launch Sybil attack [10] by faking uncompromised nodes' identities.

L'Hostipal also assumes that the ad hoc nodes are equipped with hardware needed by packet leashes [14] or Brands-Chaum protocols [7]. Hence by secure distance bounding, any pair of topological neighbors in ad hoc routing are indeed physical neighbors.

C. Design principles

Local monitoring with minimized monitoring coverage A significant distinction between secure routing and robust routing is the threat model. In many robust routing schemes, it is assumed that ad hoc nodes are motivated to forward packets and actively fix a damaged route using local queries. Unfortunately, in a secure routing scheme, we are facing malicious and selfish nodes who would not follow this cooperative assumption at all. Instead, local monitoring must be used to identify any anomalous packet forwarding, and the monitoring range must be minimized—each "good" node should monitor a minimal number of local nodes—otherwise the incurred monitoring overhead can be explored by adversarial nodes to launch denial-of-service attacks. We call this principle the "*minimal monitoring principle for secure routing*".

Localized and immediate self-healing When a packet forwarder is a non-cooperative node that loses the packet, we use a localized, immediate and efficient self-healing scheme to elect a substitution within minimal time. The "healed" path is a close approximation of the (optimal) path discovered by the underlying on-demand routing protocol. Extra self-healing overheads are incurred only in the localized apposite areas around the damaged links.

Limit the frequency of flooding (either network-wide or limited-scope) Control packet flooding, either network-wide or limited-scope, incurs tremendous energy expense and wireless channel contention. Malicious nodes can

explore this feature to deplete needed network resource. *L'Hospital* seeks to realize a secure routing paradigm that only requires a single initial RREQ flood per end-to-end connection (in the ideal case), despite of unpredictable node mobility and wireless packet loss.

End-to-end maintenance Due to the possible presence of non-cooperative nodes, the intermediate forwarders cannot be trusted. Therefore, the two ends of a connection should pay reasonable cost to maintain the intermediate self-healing communities (whose shape degenerates due to node mobility). End-to-end maintenance may include monitoring end-to-end data delivery ratio, implementing end-to-end probing, maintaining fresh routes, and finding new routes when a community en route is empty (e.g., completely compromised).

D. Community-based security (CBS)

Configuring and reconfiguring self-healing communities is the central part of our community-based scheme. For each end-to-end connection, a chain of self-healing communities along the optimal path (discovered by the underlying routing protocol) are established to thwart route disruption. This section details how a secure community at each forwarding step is created and how the secure communities are maintained facing network dynamics and possible attacks.

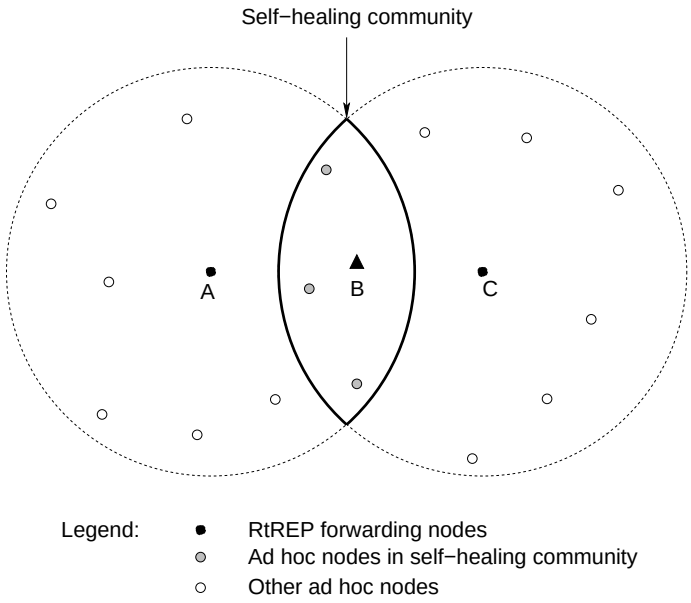

Fig. 1. **A self-healing community between a 2-hop source and destination pair**

1) Self-healing community overview: The concept of "self-healing community" is based on the observation that wireless packet forwarding typically relies on more than one immediate neighbor to relay packets. Figure 1 shows the simplest case that node B relays packets from node A to node C. Typically, node B is within the intersection of node A and C's radio range while A and C cannot hear each other. In principle, all nodes within the "moon"-shape intersection can relay packets from A to C. Nodes in such an intersection[1] form our *self-healing community*. Figure 2 depicts a chain of self-healing communities along a multi-hop path. Community-based security explores node redundancy at each forwarding step so that the conventional per-node based forwarding scheme is seamlessly converted to a new per-community based forwarding scheme. We do not require unusually high node redundancy—a self-healing community is functional as long as at least one cooperative "good" node is in the community. Intuitively, a self-healing community is a "big virtual node" that replaces a single forwarding node in conventional routing schemes (Figure 3).

[1]The actually community area in our design is the further intersection of the "moon"-shape and B's one-hop transmission circle. For the clarity of presentation we spare B's one-hop circle in all depictions in this paper.

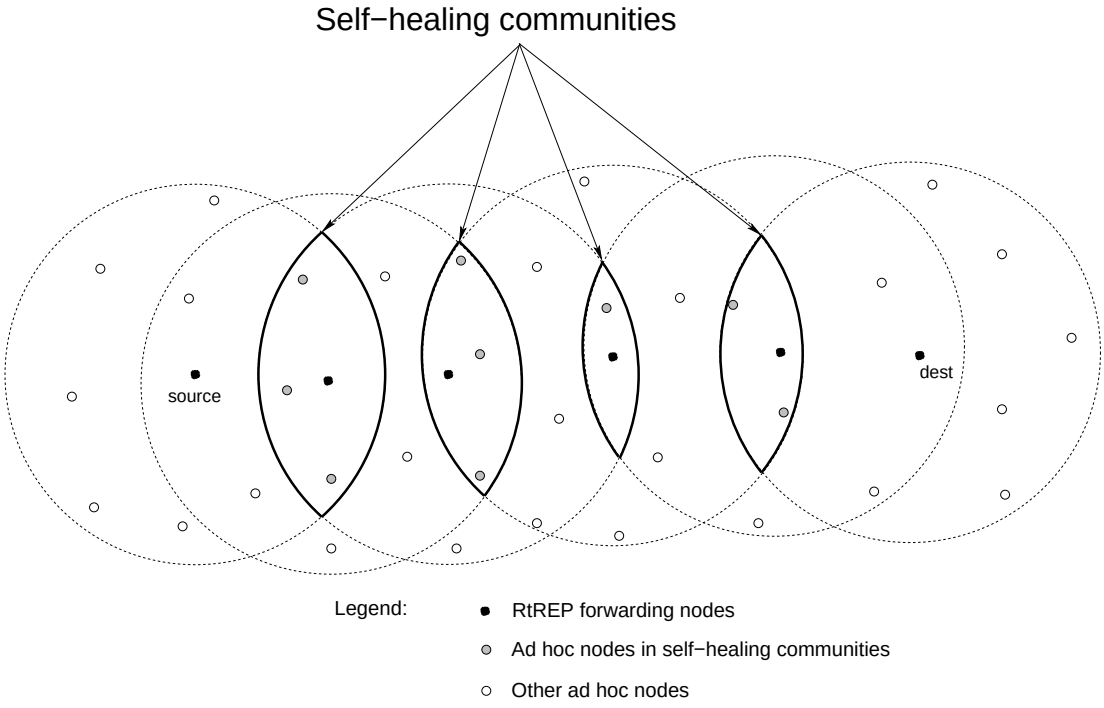

Fig. 2. **Packet forwarding self-healing communities along a multi-hop path**

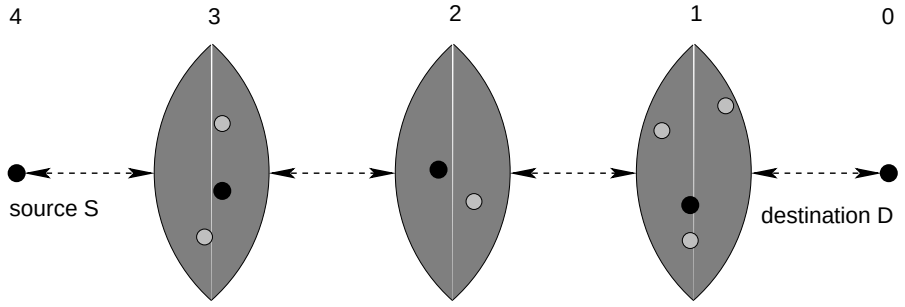

Fig. 3. **Self-healing communities as "big" virtual nodes**

2) Self-healing route discovery: A self-healing community must be formed properly. As a comparison to Figure 1, Figure 4 shows an inappropriate community between A and C. Because A and C are one-hop neighbors, it is inefficient to introduce an extra forwarder B and pay the overhead to configure the community around B, which directly violates the "minimal monitoring principle for secure routing".

To avoid such improper community configurations, we slightly change the underlying on-demand routing protocol's RREQ packet format, so that when B forwards its RREQ packet, it adds its immediate upstream A in the RREQ packet. The new RREQ packet format is[2]:

$$\langle RREQ, \underline{upstream\_node}, \dots \rangle$$

where the underlined part is newly added. The distributed Algorithm A specifies each autonomous node's action during the RREQ phase. The distributed Algorithm B specifies how RREP forwarding can be healed by nearby network members en route.

[2]We do not include detailed packet formats of the underlying on-demand routing protocol. Interested parties may check [23] for AODV and [17] for DSR. Note that in DSR the upstream node is already in its forwarding list, thus RREQ packet format is unchanged for community-based DSR.

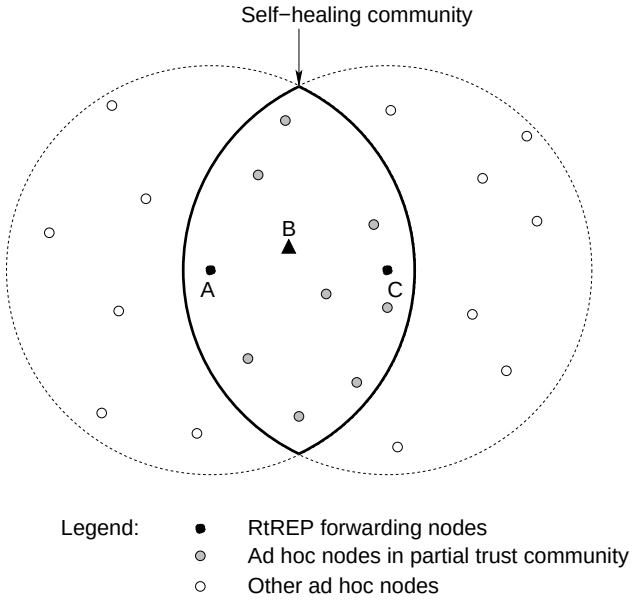

Fig. 4. **An inappropriate self-healing community**

Algorithm A: During an RREQ flood, a node just received an RREQ packet $V \rightarrow *$ for the current route discovery:

0 Insert V in my soft-state neighbor set;
1 U := the $upstream\_node$ field in the RREQ packet;
2 In my soft-state, record U as V's upstream;
3 IF {(I have not forwarded RREQ for the S-D connection) AND
4 (I have not heard U in my neighborhood during the RREQ)}
5 Record V as my RREQ upstream for the connection. Change the packet's $upstream\_node$ field to V;
6 Process the RREQ packet according to the underlying routing protocol;
7 Locally rebroadcast the RREQ packet.

In Algorithm A, lines 4 and 5 ensure that a self-healing community is only formed between two nodes that are two hops away. Line 0 exploits the RREQ flood as a precious chance to understand the current neighborhood. *Each mobile node will explore every wireless packet reception to maintain its soft-state neighbor set, which is separately maintained from its soft-state routing states.* Each node inserted into the neighbor set will be removed from the set if not refreshed in a predefined timeout (e.g., $\frac{R}{v}$ where R is the well-known one-hop transmission range and v is the estimated average node mobility speed).

Algorithm B: During RREP, a node $V'(\neq V)$ just heard the first RREP packet $E \rightarrow V$ for the current route discovery:

```
00  Remember E as my RREP upstream. Insert E in my soft-state neighbor set;
01  IF (V is not in my soft-state neighbor set)
02      End of the algorithm execution, no need to continue.

/* V must correctly ACK E, or I take over V via randomized competition */
03  WHILE {(My soft state for connection S − D is still alive) AND
04          (V didn't correctly ACK E within the bounded 802.11 backoff window)}
05      Wait for an autonomously decided random time;
06      IF (During the waiting period nobody has taken over)
07          I take over; send a take-over ACK to E: V'→E replacing V (e.g., put V's id in the ACK's payload);
08          Quit Algorithm B; run line 02 and 03 of Algorithm B_V.
09      ELSE
10          V := the node who is ACKing to E. Insert V in my soft-state neighbor set;

/* V must correctly forward and receive W's ACK, or I take over V via randomized competition */
11  W := the RREQ upstream node recorded for V;
12  WHILE {(My soft state for connection S − D is still alive) AND
13          (Both V and W are in my soft-state neighbor set) AND
14          (V didn't correctly forward within the bounded 802.11 backoff window) AND
15          (W didn't correctly ACK V within the bounded 802.11 backoff window)}
16      Wait for an autonomously decided random time;
17      IF (During the waiting period nobody has taken over)
18          Transmit the RREP packet: V'→W (i.e., I try to take over);
19          SWITCH (received ACK)
20              CASE (ACK: W→V')
21                  I win the take-over.
22              CASE (ACK: W→V'')
23                  V'' wins the take-over.
24      ELSE
25          V := the node who is forwarding the RREP packet. Insert V in my soft-state neighbor set.
```

If the node is V, then it follows the protocol specified in Algorithm B_V for the first RREP[3]. For later RREPs of the current route discovery, V simply does line 00 and 01 (see "Discussion:ACK" for reasons behind the design).

Algorithm B_V: During RREP, a node V just received the first RREP packet $E \rightarrow V$ for the current route discovery:

```
00  Remember E as my RREP upstream. Insert E in my soft-state neighbor set;

/* ACK back to E */
01  Transmit unicast ACK: V→E;

/* Forward the RREP */
02  W := my RREQ upstream node according to my soft-state;
03  Transmit the RREP packet: V→W.
```

[3]Remember in *L'Hospital*, topological neighbors are indeed physical neighbors due to secure distance bounding. Thus the first RREP is forwarded on the optimal path discovered by the underlying routing protocol (rather than on paths with wormholes). Some readers may raise questions saying the compromised malicious network members may send out RREP at any time to disrupt routing. Please see Section III-E for a cryptographic countermeasure. In a nutshell, in *L'Hospital*, the first RREP received for the current route discovery is the valid RREP delivered by the underlying routing protocol as if there is no attack.

Algorithm B_V^{ACK}: During RREP, a node V just received an ACK packet $W' \rightarrow V$ for the current route discovery:

```
00  Insert W' in my soft-state neighbor set;
01  W := my RREQ upstream node according to my soft-state;
02  IF (W = W')
03      Immediately record that W has ACKed me;
04  ELSE
05      Wait a specific timeout (e.g., 100ms) for W's reply (this is for countering a rushing attacker W');
06      IF (My soft-state shows that W has not ACKed me)
07          Replace W by W' in my soft-state's RREQ upstream record;
08      ELSE /* W has already ACKed me */
09          The ACK is redundant due to take-over competition (line 07, Algorithm B); re-transmit RREP: V→W.
```

The source S and destination D are special nodes in Algorithm B_V. The destination D should only perform lines 02 and 03 of Algorithm B_V; the source S should only perform lines 00 and 01 of Algorithm B_V. Both Algorithm B_V and B_V^{ACK} are for an actual RREP forwarder V, who can only receive two types of packets, RREP and ACK, during the RREP phase.

Let's use Figure 1 to describe a simple example of self-healing route discovery. If B is a malicious forwarder, B can use rushing attack to make C believe that the best path between source A and destination C goes through B. Therefore, C will unicast back an RREP packet to B. Fortunately, even though the malicious B will drop the RREP packet or send a corrupted RREP packet, the other cooperative nodes in the community area will be able to identify the situation and try to take over as the RREP forwarder.

- First, during RREQ phase any cooperative node B_c in the community area already remembered $V = B$ as its one-hop neighbor and $U = A$ as V's upstream node.
- Second, during RREP phase any such cooperative B_c can detect $V = B$ is not willing to forward the packet. In CSMA/CA, this can be achieved by monitoring the channel idle time and estimating B's exponential backoff window size. If B_c is very near B and hears all B's receptions, then the initial backoff windows size is 2, or 2^{n+1} after the n-th collision. However, some of B's receptions cannot be heard by a slightly distant B_c due to hidden terminals. To count B's deferring, B_c must add the estimated defer time $\tau_{defer} = \frac{l}{w}$ to the estimation where l is the estimated packet size (e.g. l =1500bytes) and w is the link capacity (e.g. 11Mbps for 802.11b). If the channel is idle for more than the upperbound of the estimated backoff window size, then B_c concludes B has dropped the RREP packet due to whatever reason (e.g., selfishness, maliciousness, route outage due to mobility, etc.). So B_c initiates its take-over action.
- Third, multiple B_c nodes may compete to forward the RREP packet. Similar to the random delay imposed in the DSR and AODV's RREQ forwarding design, each node uses an autonomous random delay to alleviate the chance of collision. Even if collision does happen, the node $W = A$ determines who wins by sending back a unicast ACK—the one who successfully receives the unicast ACK from A is the one who takes over.

If S and D are more than two hops away, then the single-hop self-healing procedure described above is executed from D to S inductively. When the underlying on-demand route discovery selects at least one non-cooperative RREP forwarder, then like what illustrated in Figure 5 and 6, each actual RREP forwarder depicted is the one who is ACKed (i.e., the one who is selected by the next RREP forwarder if there are take-over competitors). Figure 6 clearly shows a community is the intersection area of three consecutive transmission circles. Finally it is guaranteed a valid RREP comes back to S if at least one cooperative node physically presents in every community area en route.

Discussion: ACK ACKs to the unicast control packets play an important role in community-based secure routing. At the link layer, an 802.11 unicast is always ACKed. To make our design applicable to non-802.11 MAC schemes, at the routing layer we implement dedicated short ACKs for RREP packets (also for other unicast control packets, i.e., PROBE, PROBE_REP and data packets piggybacked with probing message described in Section III-D.4).

The ACK design of Algorithm B, B_V and B_V^{ACK} is justified below:

- If channel error and hidden terminal effect are strong (e.g., when node density is large and traffic load is heavy), the chance of not overhearing the ACK transmission increases. A node in the community area who

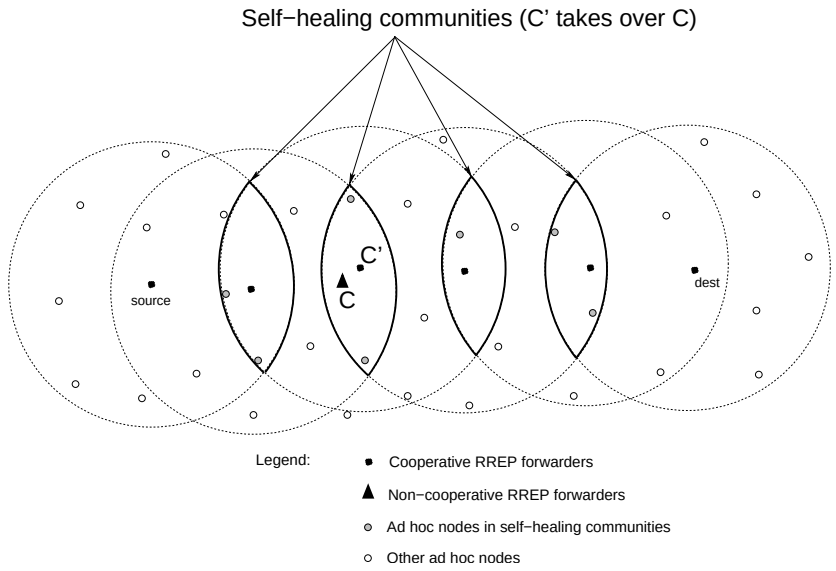

Fig. 5. **The take-over design causes no ambiguity in community configuration along a multi-hop path**

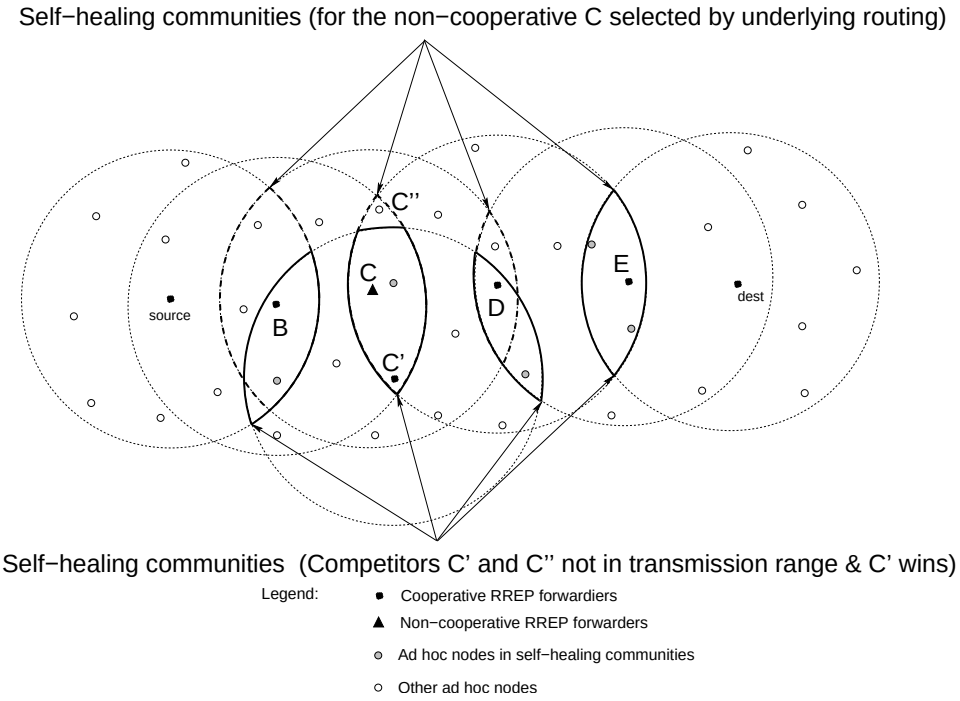

Fig. 6. **A special case in take-over (multiple competitors C' and C'' causes no trouble because of the ACK design)**

missed the ACK transmission will try to take over after its own random latency. The action on line 09 of Algorithm B_V^{ACK} is an omni-directional transmission to notify all such nodes about the previous ACK.

- When multiple take-overs are initiated by different nodes around a non-cooperative RREP forwarder. These taking-over nodes may be out of one-hop transmission range (e.g., using Figure 6 as the example, two nodes C' and C'' at the upper corner and the lower corner of the moon-shape community area cannot hear each other). It is the ACK from the RREP downstream node (line 19–23, Algorithm B) that serves as the critical notification to the competing nodes.
- As we discussed in the "Discussions: ACK again" below, the formation of a self-healing community is determined by three consecutive ACK packets (rather than the unicast control packets being ACKed, which will cause ambiguity in self-healing community configuration). This way, since our design implements ACKed forwarders at each hop, all self-healing communities en route are henceforth configured without any ambiguity (as Figure 5 and 6 illustrate).

Discussion: Fairness in taking over The current forwarder may take over for malicious purposes:

- Correct RREP unicast: If it sends out a correct RREP unicast and the take-over succeeds, then this is a cooperative action. Even though the node is "malicious" by out-of-band evaluation criteria, the take-over action fortunately heals the current round forwarding. Certainly this is not a permanent decision. If the forwarder stops forwarding in the future, then another community member will take over at that moment.
- Intentionally corrupted RREP unicast: A malicious community member B_m may also try to prevent anybody from taking over (Obviously a selfish community member won't bother to commit this crime because this attack consumes its system resource). It can skip Algorithm B (line 5 or 16) to "rush" its take-over unicast, then to send out an intentionally corrupted take-over unicast. This take-over rushing attack is addressed by requiring correct ACK contents (line 4 or line 15 of Algorithm B). In addition, a node who has just sent out a take-over unicast should wait for a well-known timeout (e.g., 1 second) before its next take-over trial. In other words, if a node B_c finds out another node B_m sends out RREP unicasts too frequently or always corrupted, then it concludes B_m is a take-over rushing attacker. From now on, channel access requests (e.g., RTS) from B_m are considered as jamming efforts.

3) Configuration of self-healing communities: A chain of self-healing communities is configured during the self-healing RREP phase. Each node must maintain a 2-bit membership flag in its on-demand soft-state for an S-D connection. Each RREP forwarder sets the membership flag to 2. A node overhearing three consecutive RREP unicast packets sets the membership flag to 1. This is because a self-healing community member must be in the transmission range of exactly three RREP forwarders: the immediate upstream forwarder, the forwarder in the same community, and the immediate downstream forwarder. As a result, a new field is added to the existing RREP packet format:

$$\langle RREP, \underline{hop\_count}, \dots \rangle$$

where the underlined part is a counter[4] for the purpose of community indexing. It is set to 0 by the destination D, then increased by one by each RREP forwarder. From the three consecutive hop count values, any community member can identify the index corresponding to its own community (i.e., the middle one). For example, if a mobile node overhears three RREP packets (of the same connection) with consecutive hop_count values 2, 3, and 4 in the strict order specified, then it can conclude it is in the community indexed by 3. Finally, to correctly maintain the communities immediately next to the source S and destination D, a community member only need to hear two consecutive RREP packets and check whether S or D is involved in the RREP packet transmissions.

Discussion: ACK again For the ease of presentation, we lied a little in the above description. We said "hearing three consecutive RREP unicast packets" will make a node set its membership flag. This is not true. Actually after a node overhears the first RREP ACK (the ACK implies the unicast happened but could be missed by the listening node due to network dynamics), it will set its membership flag only after hearing the next two RREP ACKs (plus the corresponding RREP unicasts) instead of RREP unicasts only. As we explained in "Discussion: ACK", an ACK uniquely identifies the RREP forwarder at each hop when there are take-over competitions (here obviously an un-ACKed RREP unicast is just a noise).

4) Reconfiguration of self-healing communities: The self-healing communities lose shape due to mobility and other network dynamics. For each S-D connection, we use periodic end-to-end probing to reconfigure self-healing communities. The probing interval T_{probe} is adapted with respect to network dynamics. Figure 7 illustrates the reason why self-healing routing is robust against node mobility, which translates existing communities into amorphous shapes over time. Ideally, given a maximum mobility speed and assume T_{probe} is adaptively adjusted to an appropriate value, then the probing can successfully re-configure the self-healing communities en route if at each hop there is at least one old community member (e.g., $newF$ in the figure) still stay in the forwarding area. At the hop such a community member bridges the gap between the old amorphous community and the new community.

[4]This counter is added only for the purpose of identifying the sequence in RREP forwarding. In a specific regular on-demand routing protocol, this field again could be spared and no change is made to RREP. For example, in DSR the sequence of RREP packets can be identified by looking at the decreasing source routing list.

Therefore, upon an ideally adapted T_{probe} there is no need to flood the network with RREQ for the purpose of route maintenance.

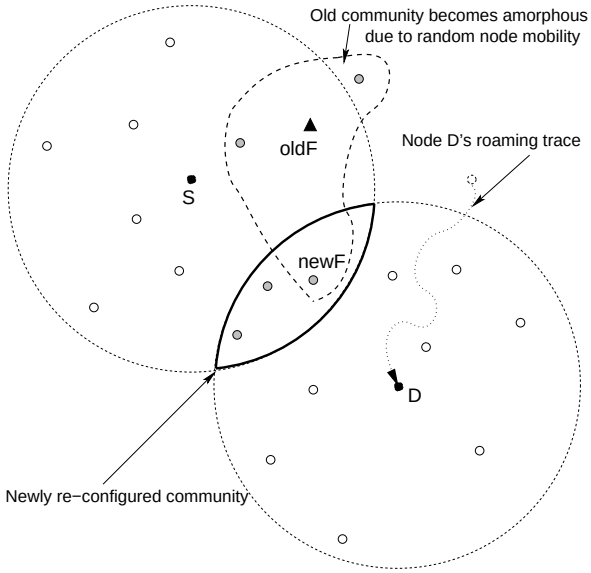

Fig. 7. **Reconfigurable self-healing communities are robust to node mobility**

We firstly describe how T_{probe} is selected in practice following a heuristic design. Whenever a take-over action happens, the taking-over node B_c also sends a short report to the source S

$$\langle TAKE\_OVER\_REPORT, (S, D, seq\#), B_c, B\rangle$$

where $(S, D, seq\#)$ identifies the end-to-end connection and B is the forwarding node being taken over. T_{probe} is initialized to be $\frac{R}{v}$ where R is the well-known one-hop transmission range and v is the estimated average node mobility speed. This initialization value approximates the time needed to roam out of one-hop transmission range. Then the source decreases its T_{probe} by $\tau_{dec} = 100$ms upon receiving such a take-over report, and increases T_{probe} by $\tau_{inc} = 10$ms if no take-over report is received in the most recent second.

As frequent take-over actions indicate more network dynamics or more non-cooperative behaviors, the heuristic scheme seeks to maintain fresher self-healing communities by issuing more probing requests. Meanwhile it also seeks to decrease probing overhead when the self-healing communities en route are relatively stable. As a result, even if the number of RREQ flooding for each connection is not 1 (which is the ideal case), this heuristic scheme significantly reduces the flooding frequency. This implies RREQ rate-limit proposals [13][23] are practical in community-based security.

We then describe the probing protocol details. The source S is responsible to keeping the on-demand route alive because it knows whether there is further data transmission. For every T_{probe} timer, the source S sends out a PROBE packet.

$$\langle PROBE, (S, D, seq\#), round\#, upstream, hop\_count\rangle.$$

where $round\#$ is increased from 0 to $2^{16} - 1$ by 1 per probe (and wrap around since then), the meaning of $upstream$ or hop_count is similar to the one used in RREQ/RREP packet format.

Upon receiving a PROBE message, the destination D replies with a PROBE_REP packet (with $D, S, seq\#$ and $round\#$ copied from the corresponding PROBE).

$$\langle PROBE\_REP, (D, S, seq\#), round\#, hop\_count\rangle.$$

In general, like the RREP phase, probing packet forwarding follows a self-healing procedure which tolerates malicious and selfish behaviors. Then self-healing communities en route are reconfigured by monitoring the forwarding chain. That is, a node who forwards the PROBE or PROBE_REP message sets its membership flag to 2 (i.e., the forwarding member), and any node overhearing three consecutive ACKs (of PROBE or PROBE_REP messages) should set its membership flag to 1 (i.e., the non-forwarding member). The hop_count field, which is increased by 1 at each stop, is similar to the same field in RREP packets to identify the indexes of self-healing communities.

Example 1: **(Analogical example to show the reason why *L'Hospital*'s community design can cope with mobility)** To better understand the seemingly complex algorithms described below, let's firstly present an intuitive explanation. Intuitively, let's say we do PROBE using a special RREQ flooding rather than unicasts. The special RREQ packets are flooded in the network just like the initial RREQ flood except with one catch: *the special RREQ (of an $S-D$ connection χ) is only forwarded by the current community members (of χ)*, which have been set up previously in RREP phase or previous probing rounds. This way, the special RREQ flood does not incur *network-wide* overhead. It *only* incurs forwarding overhead in the community areas. When T_{probe} is small enough, it is clear that the constrained RREQ floods can maintain ad hoc routes just like network-wide floods, but with much less RREQ forwarding overhead. □

Nevertheless, any flooding can be explored by the adversary to launch denial-of-service attacks. By changing the constrained RREQ flooding into more efficient PROBE unicasts (like RREP, it is with self-healing take-over protections), we obtain Algorithms P, P_V, and P_V^{ACK} described below[5].

Algorithm P: A node $V'(\neq V)$ just overheard the first PROBE packet $E \rightarrow V$ for the current round PROBE:

```
00  Insert E in my soft-state neighbor set; Remember E as V's upstream node.
01  IF (V is not in my soft-state neighbor set)
02      End of the algorithm execution, no need to continue.

/* V must correctly ACK E */
03  WHILE {(My soft state for connection S − D is still alive) AND
04          (V didn't correctly ACK E within the bounded 802.11 backoff window)}
05      Wait for an autonomously decided random time;
06      IF (During the waiting period nobody has taken over)
07          I take over; send a take-over ACK to E: V'→E replacing V;
08          Quit Algorithm P; run line 02 and 03 of Algorithm P_V.
09      ELSE
10          V := the node who is ACKing to E. Insert V in my soft-state neighbor set;

/* V must correctly forward and receive X's ACK, or I take over V via randomized competition */
11  U := the upstream_node field in the PROBE packet;
12  WHILE {(My community membership flag for the connection S − D is set) AND
13          (U is not in my soft-state neighbor set) AND
14          (V is in my soft-state neighbor set)}
        /* I am a potential member to take over V */
15      IF {(V didn't correctly forward within the bounded 802.11 backoff window) OR
16          ( (Heard a correct V→X unicast) AND
17            ( (X is in not my soft-state neighbor set) OR
18              ((X is in my soft-state neighbor set) AND (X didn't ACK V within the bounded 802.11 backoff window))
              )
           )}
19          Wait for an autonomously decided random time;
20          IF (During the waiting period nobody has taken over)
                /* I take over */
21              Change the PROBE packet's upstream_node field to E.
22              Unicast the PROBE packet to my next stop X (according to the underlying routing protocol).
23          ELSE
24              V := the neighbor node who is taking over. Insert V in my soft-state neighbor set.
25      ELSE
            /* V→X is correctly forwarded and ACKed. Over. */
26          Break the loop.
```

If the node is V, then it follows the protocol specified in Algorithm P_V and P_V^{ACK}.

[5]Note that the current community indexes derived from hop_count do *not* affect Algorithms P, P_V, and P_V^{ACK}. In *L'Hospital*, the hop_count field is merely used for setting up community membership flags.

Algorithm P_V: A node V just received the first PROBE packet $E \rightarrow V$ for the current round PROBE:

00 Remember E as my PROBE upstream. Insert E in my soft-state neighbor set;
/* ACK back to E */
01 Transmit unicast ACK: $V \rightarrow E$;
/* Forward the PROBE */
02 X := my next stop (according to the underlying routing protocol);
03 Forward the PROBE packet: $V \rightarrow X$;

Algorithm P_V^{ACK}: A node V just received an ACK packet $X' \rightarrow V$ for the current round PROBE:

00 Insert X' in my soft-state neighbor set;
01 X := my next stop (according to the underlying routing protocol);
02 IF ($X = X'$)
03 Immediately record that X has ACKed me;
04 ELSE
05 Wait a specific timeout (e.g., 100ms) for X's reply (this is for countering a rushing attacker X');
06 IF (My soft-state shows that X has not ACKed me)
07 Replace X by X' in my soft-state's PROBE upstream record;
08 ELSE /* X has already ACKed me */
09 The ACK is redundant due to competition (line 07 of Algorithm P); re-transmit PROBE: $V \rightarrow X$.

The source S should only perform line 02 and 03 of Algorithm P_V; the destination D should only perform line 00 and 01 of Algorithm P_V. Line 17 of Algorithm P is added for countering a malicious node V who lies about its next stop (e.g., X is a non-existing node). Adding this line potentially causes a redundant transmission from V'. But this overhead is needed to counter the liars.

For processing probing replies, we specify Algorithm PR as the probing counterpart of Algorithm B, where RREP is replaced with PROBE_REP. Algorithms PR_V and PR_V^{ACK} are similarly specified. It is straight-forward to treat the initial RREP as a special round 0 of probing reply, and the successive rounds of probing replies as round 1, 2, and so on. The specifications of Algorithms PR, PR_V and PR_V^{ACK} are spared in this report because they are identical to Algorithms B, B_V, and B_V^{ACK} except RREP is replaced by PROBE_REP.

Discussion: Piggybacked probing for reducing overhead Since both PROBE and PROBE_REP are short messages, they can be piggybacked to ongoing data traffic (it is easy to see that the connection identifier field $(S, D, seq\#)$ in piggybacked data packets is redundant and hence removed).

Discussion: Inconsistent states Due to wireless channel contentions and errors, it is possible that a *de facto* non-forwarding community member fails to overhear at least one of the three consecutive ACKs (of RREP or PROBE or PROBE_REP or piggybacked data packets) of the current round. Fortunately, this unlucky node has the chance to rectify its incorrect membership flag at the time of next probing round.

5) Self-healing data delivery: Community-based data delivery is a combination of conventional node-based data forwarding plus community-based healing. At the source, the source node is unambiguously the current forwarder. At each intermediate stop, the most recent RREP (or PROBE or PROBE_REP or piggybacked data packet) forwarder is supposed to be the current forwarder. Such a current forwarder plays the role of *"core"* in the associated self-healing community. However, if this node fails to forward data packet due to maliciousness, selfishness, or network dynamics, members in the associated self-healing community will make up.

Algorithm C: During data delivery, a node ($\neq V$) just overheard a unicast data packet $E \rightarrow V$ for an $S - D$ end-to-end connection:

0 Insert E in my soft-state neighbor set;
1 W := my next stop (according to the underlying routing protocol);
2 WHILE {(My soft state for connection $S - D$ is still alive) AND
3 (My community flag in the soft state is set) AND
4 (V didn't correctly forward within the bounded 802.11 backoff window)}
5 Waits for an autonomously decided random time;
6 IF (During the waiting period nobody has forwarded correctly)
7 Unicast the data packet to W.

If the node is V, then it immediately performs lines 0,1 and 7 of Algorithm C.

Note that Algorithm C requires make-up but no take-over and no network layer ACKs for unicast data packets. Another design choice is to follow Algorithm B so that each unicast data packet becomes a unicast control packet. Although this ensures per-hop reliability and thus significantly changes the network's data forwarding behavior, it may be a good choice when per-hop data packet loss ratio is huge (e.g., when either the channel error rate or the ratio of compromised nodes approaches 1).

Discussion: Why "core"? Multi-path routing [26][18] is an alternative choice of community-based routing. A significant difference between multi-path routing and community-based routing is: when route healing occurs, no coordination is considered in these designs, so the healed path could be separated and scattered. But in our design the damaged path is healed towards the core node, i.e., the intended forwarders selected by the underlying routing protocol. This achieves backward compatibility.

Discussion: Cryptography for community-based secure routing *L'Hospital*'s key management is straightforward based on conventional key management schemes: *a cryptographic key is shared between two neighboring communities rather than two neighboring nodes*. The key is used in encryption and message authentication to protect message privacy and integrity against external adversary. In particular, every packet that is transmitted to a community member must be seen/decrypted by other community members so that they can monitor misbehaviors.

Some simple solutions for community-based key management are available now. Recently Deng et al. [9] proposed a cluster key design based on a global key and a distributed echo-back scheme. This key management design can be used to protect our self-healing communities—the RREQ/RREP route discovery phases are protected by the global key, then each RREP forwarder treats its community as a cluster in [9]. Currently we are investigating more resilient methods to avoid the use of global key and more efficient methods to reduce the incurred key management overhead.

E. General discussions on the protocol design

Soft-state design In MANET routing an adaptive soft-state strategy is used to cope with the highly dynamic network. For example, routing states are maintained using timeouts, and all unicasts and ACKs are tried for a threshold number of time (then errors will be reported to upstream if necessary). In the community-based secure routing, various new states are added to the underlying routing protocol's soft-state. These include set of current neighbors, community membership flags and probing interval T_{probe}. These records are maintained in the same way that DSR and AODV maintain their routing states.

Wormhole attack In [14] Hu *et al.* proposed to use "wormholes" to attack ad hoc routing schemes. Compared to active jamming, wormhole attack [14] is more "covert" in nature and harder to detect. A wormhole attacker tunnels messages received in one location in the network over a low latency link and replays them in a different location. This typically requires at least two adversarial devices colluding to relay packets along a fast channel available only to the attackers, so that a temporal-spatial "wormhole" is realized with respect to multi-hop routing. The "wormholes" nodes can selectively let routing messages get through. Then the "wormhole" link has higher probability to be chosen as part of multi-hop routes due to its excellent packet delivery capability. Once the attacking nodes know they are en route, they can launch "black hole" attack to drop all data packets or "gray hole" attack to selectively drop some critical packets.

In MANET, a typical countermeasure against wormhole attackers is to verify neighbor relation. This is due to the fact that radio propagation speed is the maximum in physics. Hence wormholes shorter than one-hop transmission range impose little threat as the original transmission (which is to be replayed by the short-range wormhole devices) features better routing metrics. (1) Physical layer countermeasures, such as RF watermarking, seek to prevent wormholes by increasing the difficulties to capture the signal patterns. The data bits are transferred in some special modulating method known only to the neighbor nodes. (2) Packet leash is a solution proposed by Hu, Perrig and Johnson for wormhole detection [14]. The leash is the information added into a packet to restrict its transmission distance. It requires either geographical location service support, or time synchronization amongst neighboring nodes. In the geographical leashes, the location information and loosely synchronized clocks together verify the neighbor relation. In the temporal leashes, the TIK protocol efficiently bounds a packet's transmission distance given tightly synchronized clocks. (3) An approach to detect wormholes without clock synchronization is proposed

by Capkun et al. [32]. Every node is assumed to be equipped with a nano-second hardware that can use variants of Brands-Chaum protocol [7] to securely measure one-hop distance bound. (4) Another approach is based on the use of directional antennas. In [12], neighboring nodes examine the directions of the received signals from each other and a shared witness. Only when the directions of both pairs match, the neighbor relation is confirmed.

In this paper we assume that the network is already protected by either packet leashes or variants of Brands-Chaum protocol. This way, any pair of topological neighbors in ad hoc routing are indeed physical neighbors.

Directional and variable-power transmissions As described in [1], malicious nodes may use directional antenna and variable-power transmissions to attack ad hoc routing. In the context of community-based secure routing, the essence of such attacks is to break our design assumption that all nodes use omnidirectional radio with (nearly) identical transmission range.

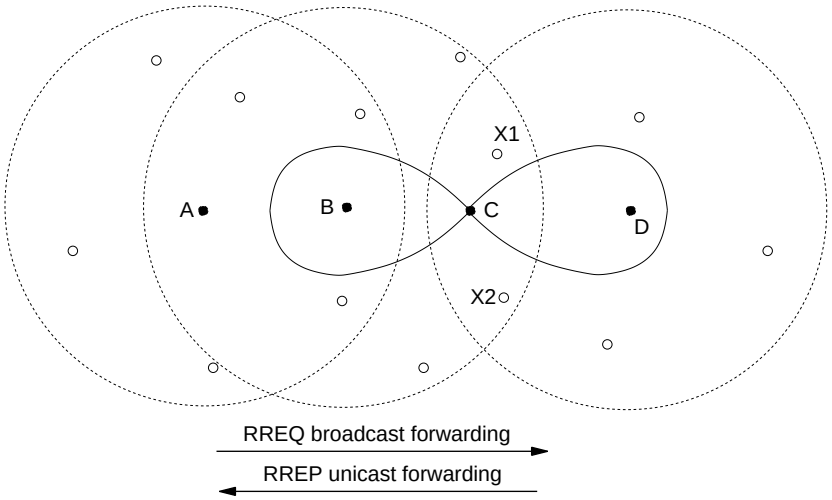

Fig. 8. **Attackers using directional or variable-power transmissions**

Fortunately, such misbehavior can be fixed in community-based secure routing. We divide the discussion into two major cases: in CASE I the malicious nodes do not use directional or variable power transmission during the initial RREQ flooding procedure; and in CASE II the malicious nodes do.

In CASE I, the initial RREQ flood is done as described in previous sections. As depicted in Figure 8, the destination D is cooperative with its own connections so it uses standard omni-directional radio in its transmissions. Then every cooperative forwarder en route follows the same rule, until the moment the RREP packet meets an adversarial forwarder. Without loss of generality, let's assume the adversarial node is C, who uses a directional (or variable-power) transmission to unicast back ACK to D.

- Algorithm B is triggered on the *de facto* community members $X1$ and $X2$ once they receive the RREP transmission $D{\rightarrow}C$. Algorithm B (line 04) ensures that $X1, X2$ will invoke their take-over actions as they cannot hear C's ACK.
- If C has ever ACKed D by directional ACK transmission, D will resend $D{\rightarrow}C$ due to Algorithm B$_V^{ACK}$ (line 09). C must eventually let $X1$ and $X2$ hear its ACK to D, or is trivially detected as a malicious node.

Or C uses a directional (or variable-power) transmission to forward the RREP packet to its RREQ upstream node B.

- Algorithm B is triggered on the *de facto* community members $X1$ and $X2$ once they receive the RREP transmission $D{\rightarrow}C$.
- Now that both C (the receiver in RREP transmission $D{\rightarrow}C$) and B (recorded RREQ upstream node of C) are in $X1, X2$'s neighbor lists (e.g., the list is maintained during the initial RREQ flood), Algorithm B (line 13) ensures that $X1, X2$ will be invoked no matter what C does. If B has ever ACKed C, B will notify $X1, X2$ about its decision. Therefore, the case is same as the one when $X1$ or $X2$ missed $C{\rightarrow}B$ transmission due to channel error or hidden terminal contention (see "Discussion: ACK" in Section III-D.2).

In a nutshell, in Algorithm B we have already considered CASE I.

In CASE II, a malicious node V uses directional antenna in RREQ forwarding. Such misbehavior can be countered by secure neighbor detection protocols, which have been studied in many secure routing literatures [15][20]. In

the example depicted in Figure 8, C has to authenticate itself to its one-hop neighbors using a secure neighbor detection protocol when it roams into a new neighborhood, otherwise nobody will forward its packets (including RREQ packets). Although C uses directional transmission in its RREQ forwarding, by changing Algorithm B line 13 to "V is in my soft-state neighbor set", it is guaranteed that the *de facto* community members $X1$ and $X2$ will initiate the take-over process as usual. Now that $X1$ and $X2$ do not know B is C's upstream node because they did not hear C's directional RREQ forwarding. In this case each of them will forward to its own upstream node recorded for the end-to-end connection. This results in a furcation and eventually multiple healed paths to the source. The source has the choice to use the best one (e.g., the one with first coming-back RREP) in unicast routing.

Packet modification attacks It is possible that a malicious network member attacks *L'Hospital* by using a random node in the upstream field. We add a simple enhancement in secure neighborhood discovery to detect such forged upstream nodes: the secure neighborhood discovery is rendered for 2-hop neighborhood rather than 1-hop neighborhood. This requirement incurs more neighborhood discovery overhead, but is needed to counter a more insidious adversary.

For another field hop_count used in *L'Hospital*, a malicious network member can launch attacks by not following the "increment-by-1" rule during unicast control packet forwarding. This anomaly is detectable by a simple intrusion detection system monitoring the field.

Collaborative adversarial RREP forwarders It is possible that there are collaborative adversarial nodes in the network. All algorithms specified in this report are unaffected if the adversarial nodes are not consecutive RREP forwarders (i.e., at most every other RREP forwarder is non-cooperative). If two or more consecutive RREP forwarders are adversarial, then the situation is similar to the one of directional RREQ forwarding. Again we use Figure 8 as the example. When $X1$ is taking over and B is adversarial, B does not correctly ACK $X1$ (otherwise $X1$'s take-over succeeds). Right now $X1$ uses its own upstream node recorded for the connection. Similarly, $X2$ will do the same thing. This results in a route furcation and eventually multiple healed paths to the source, who will make the final decision on how to use the result.

Enforcing end-to-end route discovery (or ensuring RREP is from the destination) In *L'Hospital*, topological neighbors are indeed physical neighbors assuming an underlying secure distance bounding protocol. Therefore, if RREP is indeed from the destination, then the first RREP received must be the valid RREP used in the underlying ad hoc routing protocol because

- RREP is unicast. There is no competition if an RREP forwarder behaves properly. Rushing attack [15] is not applicable here.
- By induction, for the base case the destination is cooperative with its own connections, so the first RREP is the valid RREP to the next cooperative RREP forwarder; if this is also true for the i-th cooperative RREP forwarder, then the $(i+1)$-th RREP forwarder will receive the valid RREP as its first RREP (of the current route discovery).

The following problem statement and cryptographic countermeasure can be used to protect route discovery.

Given two ends S and D of a connection, how does S ensure that an RREP packet must be originated from D? The answer is that S adds a commitment field $commit$ in its RREQ packets,

$$\langle RREQ, upstream\_node, \underline{K_D(K_{reveal}), K_{reveal}(\alpha)}, \dots \rangle$$

where $K(M)$ means using key K to encrypt message M, K_D is the public key of D (or an end-to-end symmetric key shared between S and D if available), α can be fixed to any well-known plaintext (e.g., "ad hoc routing"), and K_{reveal} is a random nonce selected for each route discovery.

Then D presents a de-commitment field in its reply. In the previous case, the de-commitment field $K_{decommit} = K_{reveal}$. In other words, RREP packet has a newly added field

$$\langle RREP, hop\_count, \underline{K_{decommit}}, \dots \rangle$$

Any community node can verify whether

$$K_{reveal}(\alpha) \stackrel{?}{=} K_{decommit}(\alpha).$$

If it is a match, then the RREQ reply does originate from the destination D. Otherwise, the RREP packet must be ignored.

Similar technique is also applicable to PROBE and PROBE_REP packets, so that each probe is rendered end-to-end. The source can also set up a commitment in the current PROBE round, then open the decommitment in the next PROBE round. An exemplary protocol to commit next round in the current round is Guy-Fawkes protocol [2]. In particular, the RREQ phase is 0-th round probing, and the RREP phase is 0-th round probing reply. In i-th round probing, the source selects a random nonce X_i, computes its hash $h(X_i)$ using a cryptographic one-way hash function, appends $h(X_i)$ and X_{i-1} to the the current probing packet M_i, finally computes $Z_i = h(M_i, h(X_i), X_{i-1})$ and appends Z_i to the current probing packet. Each verifying node only needs to cache or obtain X_i to authenticate the probing packet (by comparing the computed Z_i and the embedded Z_i in the packet). The first nonce X_{-1} used in RREQ needs to be bootstrapped by some external mechanisms, for example, a conventional digital signature.

Energy efficiency Community-based security requires each ad hoc node to constantly monitor its neighborhood (including configured communities if there is any). This implies the network interface must be ready for packet reception all the time. Real measurements [31][11] have shown that various network interfaces consume much less energy in the "receive mode" than in the "transmit mode", and on some popular interface cards the energy consumed in the "receive mode" is comparable to the "idle mode" (though there is no standard definition, "idle mode" typically refers to an energy efficient quasi-active mode so that the device's energy consumption is minimal and the device can be made active in minimal latency). Various schemes [30] have also been proposed to significantly decrease energy expense for the "receive mode" without affecting wireless packet reception guarantee.

IV. Analytic study on *L'Hospital*'s effectiveness

In this section we use an analytic model to verify the effectiveness of community-based secure routing. We define a quantity "*effectiveness gain*" EG to quantify the gained routing effectiveness.

A. Underlying spatial model

We divide the network area into a large amount of small (virtual) grids, so that the grid size is even smaller than the physical size of the smallest network member. This way, each grid is either empty, or is occupied by a single node. Also because the network area is much larger than the sum of all mobile nodes' physical size, the probability that a grid is occupied by a mobile node is very small.

Now a bionomial distribution $B(n, p)$ defines the probabilistic distribution of how these grids are occupied by each mobile ad hoc node. Here n, the total number of grids, is very large; and p, the probability that a grid is occupied by the single node, is very small. When n is large and p is small, it is well-known that a bionomial distribution $B(n, p)$ approaches Poisson distribution with parameter $\lambda = n{\cdot}p$. Hence this bionomial spatial distribution is translated into a *spatial Poisson point process* [8] to model the random presence of the network nodes. In other words, suppose that L events occur in area $\mathcal{A}$ (here an event is an ad hoc node's physical presence). If the node density $\rho_{_L} = \frac{|L|}{\mathcal{A}}$ (where $|\cdot|$ denotes the cardnality of a set, and $\rho_{_L} = |L| \cdot \rho_1$ if nodes roam independently) is equivalent to a random sampling of $\mathcal{A}$ with rate $\rho_{_L}$. Let x denote the random variable of number of related network member nodes. Then the probability that there are exactly k nodes in a specific area $\mathcal{A}$ is

$$Pr[x = k] \quad = \quad \frac{(\rho_{_L}\mathcal{A})^k}{k!} \cdot e^{-\rho_{_L}\mathcal{A}} \tag{1}$$

The choice of ρ_1 depends on the underlying mobility model. Some stochastic mobility models which directly choose a destination direction rather than a destination point and allow a bound back or wrap-around behavior at the border of the system area are able to achieve a uniform spatial distribution [4]. However, the others are not. Let's use random way point (RWP) model, the most popular one currently used in simulation studies, as the underlying mobility model. The probability of mobile node's spatial distribution in RWP model has been extensively analyzed in various literatures [5] [6] [25]. For a network deployed in a bounded system area, let the random variable $\Omega = (X, Y)$ denote the Cartesian location of a mobile node in the network area at an arbitrary time instant t. The spatial distribution of a node is expressed in terms of the probability density function

$$\begin{aligned} \rho_1 \;&=\; f_{XY}(x, y) \\ &= \lim_{\delta \to 0} \frac{Pr[(x - \frac{\delta}{2} < X \leq x + \frac{\delta}{2}) \wedge (y - \frac{\delta}{2} < Y \leq y + \frac{\delta}{2})]}{\delta^2} \end{aligned}$$

The probability that a given node is located in a subarea $\mathcal{A}'$ of the system area $\mathcal{A}$ can be computed by integrating ρ_1 over this subarea

$$Pr[\text{node in } \mathcal{A}'] \;=\; Pr[(X,Y)\in\mathcal{A}'] \;=\; \iint_{\mathcal{A}'} f_{XY}(x,y)d\mathcal{A}$$

where $f_{XY}(x,y)$ can be computed given geometric properties of the network. For example, as suggested in [6], we can use the analytical expression

$$\rho_1 \;=\; f_{XY}(x,y) \;\approx\; \frac{36}{a^6}\left(x^2-\frac{a^2}{4}\right)\left(y^2-\frac{a^2}{4}\right)$$

for a square network area of size $a\times a$ defi ned by $-a/2 \le x \le a/2$ and $-a/2 \le y \le a/2$.

Therefore, the node density $\rho_{_L}$ is a *location dependent* variable. In particular for the random waypoint model, $\rho_{_L}$ is higher at the central area and lower at the boundary area [5][6]. For location dependent distributions, the probability of (1) that there are exactly k nodes in a subarea $\mathcal{A}'$ of the system area $\mathcal{A}$ (with respect to a tiny unit area) is changed to

$$Pr[x=k] \;=\; \iint_{\mathcal{A}'} \left(\frac{\rho_{_L}^k}{k!}\cdot e^{-\rho_L}\right)\, d\mathcal{A}$$

where $\rho_{_L}$ is the node's spatial distribution function with respect to the underlying mobility model.

B. Geometric properties of self-healing community

In Section III we enforce the policy that the minimal distance between two 2-hop forwarders cannot overhear each other, that is, their distance is larger than 1-hop transmission range. Figure 9 depicts the maximum case when the distance between two 2-hop forwarders is $(1+\epsilon)\cdot R$ (where ϵ is a negligible quantity). On the other hand, Figure 10 depicts the minimum case when the distance between two 2-hop forwarders is $(2-\epsilon)\cdot R$.

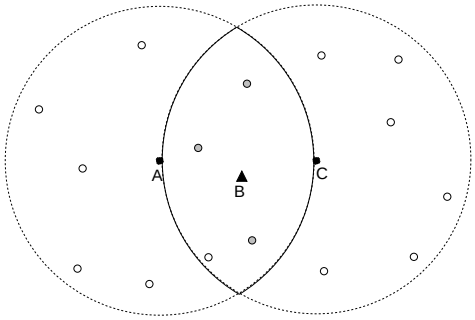

Fig. 9. **Self-healing community: maximum case**

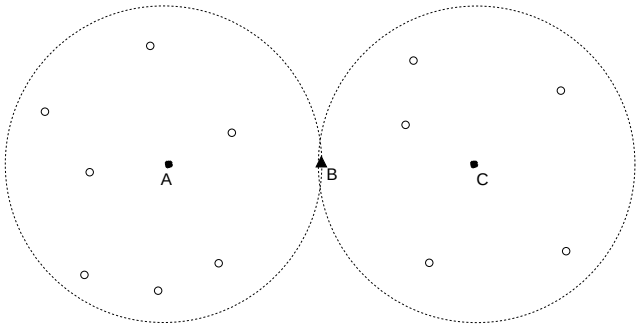

Fig. 10. **Self-healing community: minimum case**

In the minimum case the area is approximately 0. And in the maximum case the area occupied by the self-healing community is approaching

$$\mathcal{A}_{heal} \;=\; \left(\frac{2\pi}{3}-\sqrt{3}\right)R^2.$$

Suppose the RREQ procedure is a truly random process where the distance between 2-hop forwarders randomly distributes over the range between $(1+\epsilon)\cdot R$ and $(2-\epsilon)\cdot R$. We expect the size of a self-healing community is the average case:

$$E(\mathcal{A}_{heal}) \;\approx\; \left(\frac{\pi}{3}-\frac{\sqrt{3}}{2}\right)R^2.$$

Therefore, the probability that the expected self-healing community area $E(\mathcal{A}_{heal})$ has exactly k nodes is

$$Pr[x=k] \;=\; \iint_{E(\mathcal{A}_{heal})} \left(\frac{\rho_{_L}^k}{k!}\cdot e^{-\rho_L}\right)\, d\mathcal{A}.$$

C. Spatial model with adversarial presence

We adopt a probabilistic adversarial model. Amongst L authenticated network members, there are $\theta \cdot L$ non-cooperative nodes and $(1-\theta)\cdot L$ cooperative nodes. If the network is protected by cryptographic authentication schemes (e.g., by KDC in Ariadne [13] or by certification in ARAN [27]), non-network member nodes cannot join the network to be forwarders. Here θ becomes the *non-cooperative ratio* that quantifies the number of compromised or selfish network members.

Let y denote the random variable of number of cooperative network members in the expected self-healing community area. The probability that the expected area has k cooperative nodes is

$$\mathit{Pr}[y=k] \;=\; \iint_{E(\mathcal{A}_{heal})} \frac{\left((1-\theta)\cdot\rho_{_L}\right)^k}{k!}\cdot e^{-(1-\theta)\cdot\rho_{_L}}\; d\mathcal{A}$$

In a community-based secure routing scheme, the per-hop route discovery success ratio is

$$\begin{aligned} P_{community} &= \mathit{Pr}[y{\geq}1] \;=\; 1-\mathit{Pr}[y=0] \\ &= \iint_{E(\mathcal{A}_{heal})} \left(1-e^{-(1-\theta)\rho_{_L}}\right)\; d\mathcal{A}. \end{aligned} \tag{2}$$

D. Effectiveness gain

In regular on-demand routing, if (at least) one non-cooperative "bad" node presents in the community area and launches rushing attack [15] in route discovery, then with a high probability p_{rush} the bad node will be selected as an RREP forwarder. For simplicity of analysis, let's assume $p_{rush}=1$ and the bad forwarder drops its RREP packet. Let z denote the random variable of number of non-cooperative network members in the expected self-healing community area. The probability that the expected area has k non-cooperative nodes is

$$\mathit{Pr}[z=k] \;=\; \iint_{E(\mathcal{A}_{heal})} \frac{\left(-\theta\cdot\rho_{_L}\right)^k}{k!}\cdot e^{-\theta\cdot\rho_{_L}}\; d\mathcal{A}$$

In a regular on-demand routing scheme without protection, the per-hop route discovery success ratio is computed from knowing all nodes in the forwarding area are cooperative. The ratio is only

$$\begin{aligned} P_{regular} &= \sum_{k=1}^{\infty} \mathit{Pr}[y=k|x=k] \\ &= \sum_{k=1}^{\infty} \frac{\mathit{Pr}[x=k|y=k]}{\mathit{Pr}[x=k|y\neq k]\mathit{Pr}[y\neq k]+\mathit{Pr}[x=k|y=k]\mathit{Pr}[y=k]} \\ &= \sum_{k=1}^{\infty} \frac{\mathit{Pr}[z=0,y=k]}{\mathit{Pr}[x=k|y<k]\mathit{Pr}[y<k]+\mathit{Pr}[z=0,y=k]\mathit{Pr}[y=k]} \\ &= \sum_{k=1}^{\infty} \frac{\mathit{Pr}[z=0,y=k]}{\left(\sum_{i=0}^{k-1}\mathit{Pr}[z=k-i,y=i]\right)\mathit{Pr}[y<k]+\mathit{Pr}[z=0,y=k]\mathit{Pr}[y=k]} \end{aligned}$$

Since random variables y and z are independent, it is true that $\mathit{Pr}[y=a,z=b]=\mathit{Pr}[y=a]\cdot\mathit{Pr}[z=b]$. Now we can compute $P_{regular}$. However, it is a complex formula and we use the following equation to approximate the result.

$$\begin{aligned} P_{regular} &\approx \sum_{k=1}^{\infty} \mathit{Pr}[x{\geq}1]\cdot\mathit{Pr}[z=0] \\ &= \iint_{E(\mathcal{A}_{heal})} \left((1-e^{-\rho_{_L}})\cdot e^{-\theta\rho_{_L}}\right)\; d\mathcal{A}. \end{aligned} \tag{3}$$

The per-hop routing *effectiveness gain* for the community-based secure routing is defined as the ratio between the two routing probabilities: the self-healing one with community-based security, and the regular one without the protection.

$$EG \;=\; \frac{P_{community}}{P_{regular}} \;\approx\; \frac{1-e^{-(1-\theta)\rho_{_L}}}{(1-e^{-\rho_{_L}})\cdot e^{-\theta\rho_{_L}}}$$

EG is a simple metric that does not depend on the size of self-healing community and the number of hops. Figure 11 and 12 illustrate EG for a very small non-cooperative ratio in a scalable network. The effectiveness gain is huge. It is even more tremendous when either network scale or non-cooperative ratio increases.

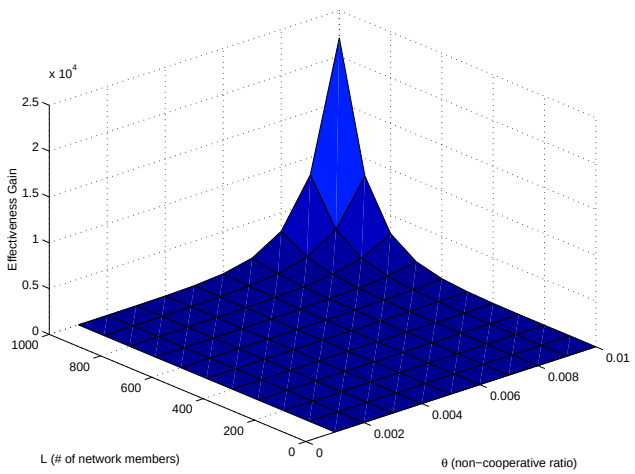

Fig. 11. **Effectiveness gain** EG (with normalized ρ_1)

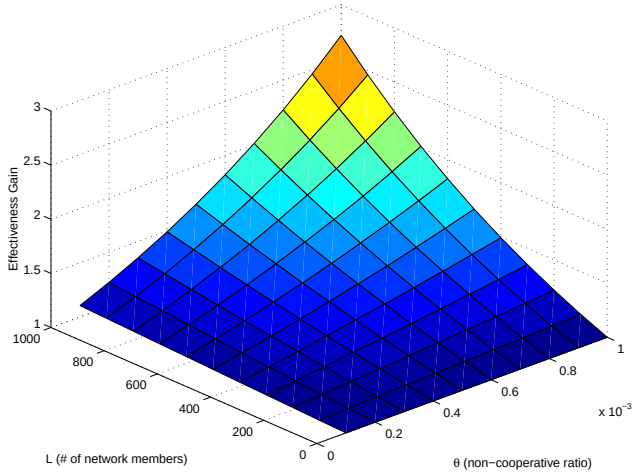

Fig. 12. **More details for Figure 11**

V. Simulation Study

A. Simulation Environment

We implement community-based security routing scheme in QualNet [28], a detailed packet- level network simulator. For our simulations, we use CBR (Constant Bit Rate) application, UDP/IP (User Datagram Protocol/Internet Protocol), IEEE 802.11 MAC and physical channel based on two-ray ground propagation model. For network device parameters, we use 2Mbits/sec channel capacity (i.e., bandwidth) and 250 meter power range. Random waypoint model [16] is used for scenarios with node mobility. The results are averaged over several simulation runs conducted with various random seeds. In each simulation scenario, 150 nodes are randomly placed within a 2400m$\times$600m field. For each CBR session, data packets of 512 bytes are generated in a rate of 4 packets per second for 2 minutes. The source-destination pairs are chosen randomly from all the nodes. During total 15 minutes simulation time, in average, five short-lived randomly selected CBR sessions are maintained.

We evaluate community-based security routing that builds on top of AODV (denoted as CBS-AODV in the results) with comparison to original AODV protocol. The following metrics are used for measurement. *(i) packet delivery ratio*: the ratio between the number of data packets received and those originated by the sources. *(ii) routing overhead*: total bytes of routing control packets. Each hop-wise transmission of a routing packet is counted as one transmission and its size is counted. For CBS-AODV, new types of control packets are all calculated. *(iii) average end-to-end packet latency*: the average time from when the source generates the data packet to when the destination receives it. This includes: route acquisition latency, processing delays at various layers of each node, queueing at the interface queue, retransmission delays at the MAC, propagation and transfer delay. Especially community makeup back off delay is included for CBS-AODV. *(iv) average route acquisition latency*: the average latency for discovering a route, i.e., the time elapsed between the first transmission of a route request and the first reception of the corresponding reply. *(v) number of triggered route request flooding*: the number of route search flooding initiated by the sources. This metric is used to show that using the community forwarding and self-healing community maintenance, recourse depletion attack through excessive control packet flooding can be limited,

Our simulation will investigate (1) impact of internal adversaries on the performance and the resilience of community forwarding against rushing attack and black hole attack; and (2) impact of node mobility on community-forwarding scheme under these attacks. In the first simulation study (Figure 13 to 15), we use static network scenarios to emphasize only the impact of these attacks. We vary the percentage (p) of internal adversaries from 0 to 10% (e.g., if p = 10, 15 nodes (0.1 * 150 nodes) are adversaries). In our second experiment (Figure 19 to 21), we fix the attacker ratio to 1 percent, and vary the node mobility from static to a speed of 10 m/s while the minimum and maximum speeds are the same. The pause time is set to 30s.

B. Simulation Results

Figure 13 illustrates the delivery ratio as function of increasing attacker ratio in the network. With the increase of the number of attackers in the network, more attackers will place themselves on the routing paths through rushing

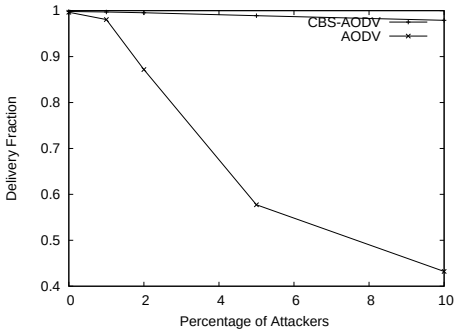

Fig. 13. **Data Packet Delivery Ratio**

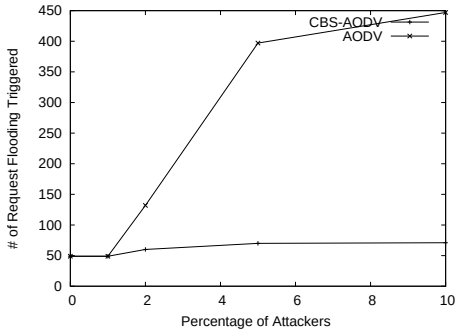

Fig. 14. **Number of Triggered Route Request Flooding**

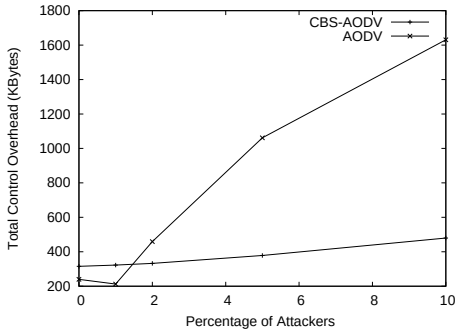

Fig. 15. **Control Overhead (Kbytes)**

attacks and hence to perform black hole attacks on data transmission and also to perform resource depletion attacks. The figure clearly shows that AODV is impaired by the attacks while CBS-AODV suffers little. The results demonstrated in Figure 14 and Figure 15 further explains this observation. As shown in Figure 14, without a remedy of community forwarding, in AODV the attacks successfully cause communication sources to issue more route search-flood as a mean of launching resource depletion attacks. Figure 15 verifies that AODV generates very high routing overhead when more attackers appear in the network.

Note that Figure 15 also shows potentially higher overhead of CBS-AODV compared to AODV in an ideal scenario, e.g., static network without attackers in order to maintain the self-healing community chains. However, such extra overhead is quite limited so the packet delivery is not damaged. Recall that the probing frequency is adjustable so that one can maintain the probing overhead to be minimized in the ideal network scenario. Notably, the community maintenance overhead is not however affected by the percentage of attackers as illustrated in Figure 14.

Figure 16 shows the portion of forwarder and non-forwarder that actually performs packet forwarding. In CBS-AODV, it is clear that with increasing attacker ratio, the forwarders fail more in forwarding, while the community nodes forward more packets to make up for packet losses at attackers. The figure shows both cases for piggybacking and not piggybacking probing message on data packets.

Figure 17 and Figure 18 collectively illustrate the delay performance of the community-based security scheme. The impacts are two folds. On one hand, with the community support, initial route acquisition latency is small for CBS-AODV due to the fact that any dropping of route reply packets will be backed up by community nodes. While for AODV, sources have to re-send RREQ packets when replies are not received or not received in time. This increases the route acquisition time (Figure 17). When attackers increase, the chances of losing RREPs are higher. On the other hand, in the presence of packet losses, community nodes back up transmissions after a time period in CBS-AODV. This mechanism produces very high packet delivery ratio in the cost of prolonged end-to-end delay. The pace of increasing in delay is not fast though when attacker ratio increases. In contrast, Figure 18 shows rapid reducing in end-to-end delay for AODV. This is caused by rapid reduction in packet delivery while the successfully delivered packets are the ones with closer destinations on average. Our results on path lengths validates this reasoning (not shown here due to page limit) by showing that AODV reduces path length from 4.36 to 3.61 with the same x axis for this simulation configuration while CBS-AODV remains changes within 0.2 on average.

Figure 19 to 21 show the impact of node mobility on community based security schemes. The attacker ratio is set low at one percent so AODV works well when there is no mobility. As we observed from our simulation, the members of community change frequently due to node mobility. In some extreme cases, a source has to reinitiate a route request flooding to rebuild the chain.

Figure 19 shows the delivery ratio when only one percent of nodes are attackers. It shows that CBS-AODV slightly degrades the delivery ratio when mobility increases. As expected, community nodes have forwarded many packets when primary nodes failed to do so. The reduction in delivery ratio is due to the fact that CBS-AODV has to rebuild the communities when mobility causes the chain to break. Unlike AODV, CBS-AODV does not have packet buffering mechanism, thus during the reconstruction period, packets are lost. However, in this low threaten scenario, AODV degrades quickly because mobility increases chances for link breakage and for attackers to rushing to the routes.

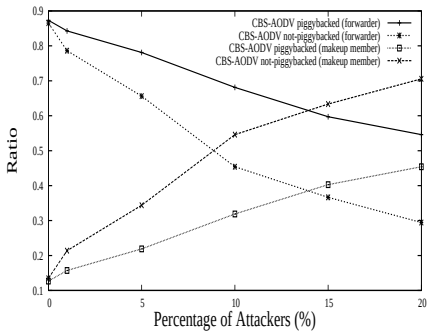

Fig. 16. **Forwarding by Intermediate or Community Nodes**

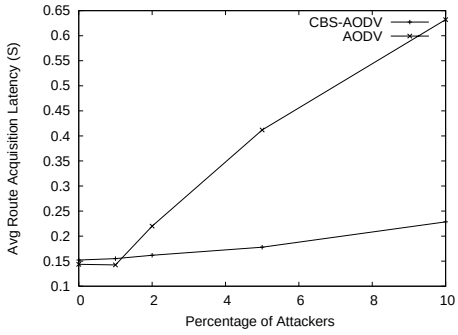

Fig. 17. **Average Route Acquisition Latency (S)**

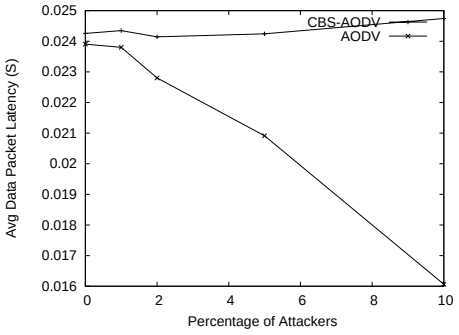

Fig. 18. **Average End-to-end Latency**

Figure 20 demonstrates that when there is mobility, both AODV and CBS-AODV initiated more route discovery than in a static case. However CBS-AODV generates less flooding than AODV as the probe mechanism remedied a lot of link breakage on the data paths. The sources are able to continue transmitting data using the new paths repaired through take over messages. The figure shows that high mobility also increases the change of breakage of the community chains, which leads to the increase of source initiated flooding.

Figure 21 shows the overall control overhead generated by both protocols when mobility increases. The increasing trend in control overhead of CBS-ADOV is due to the fact that CBS-AODV adapts its probing interval to a shorter period when mobility increases. In the current setting the interval changes from 2 second to 0.8 second when mobility increases from zero to 10 m/s. In the meantime, more TAKEOVER messages are issued in higher mobility for route repairs. In some cases, such repairs are performed in consecutive hops. After all, the figure shows that CBS-AODV incurs less control overhead than AODV under the simulation parameters.

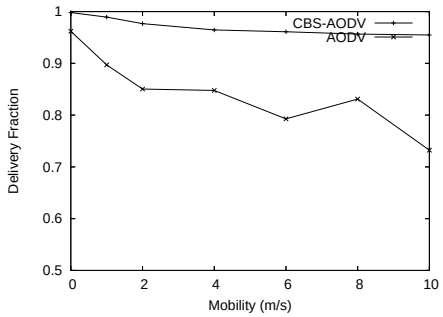

Fig. 19. **Packet Delivery Ratio**

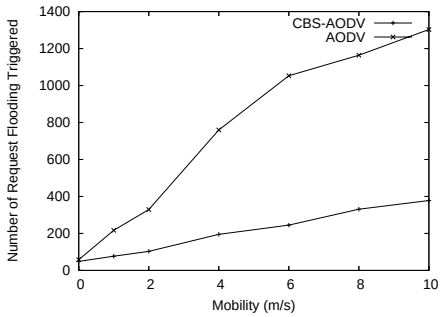

Fig. 20. **Number of Triggered Route Request Flooding**

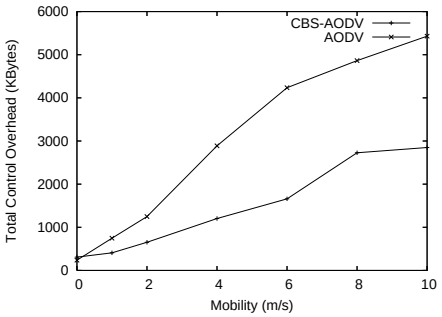

Fig. 21. **Control Overhead (Kbytes)**

VI. Comparison to related work

Recently many solutions are proposed for ad hoc routing schemes to mitigate the problem of routing disruption. To resist attacks from non-network members, either public key based digital signatures [27] or symmetric key based protocol (e.g., TESLA [24])[13] is used to differentiate legitimate members from external adversaries. Afterwards network members refuse to accept or forward any unauthenticated packet. However, such cryptographic countermeasures cannot fully answer the routing disruption challenge. As demonstrated in "wormhole attack" [14], "rushing attack" [15] and the resource depletion attacks studied in this paper, malicious nodes can easily disrupt ad hoc routing without breaking the cryptosystems in use. A wormhole attacker tunnels messages received in one location in the network over a low latency link and replays them in a different location. The attacking nodes can selectively let routing messages get through. Then the "wormhole" link has higher probability to be chosen as part of multi-hop routes due to its excellent packet delivery capability. Once the attacking nodes know they are en route, they can launch various attack against data delivery. In rushing attack, malicious nodes increase the chance to be forwarder by rushing RREQ forwarding. Then they can launch similar attacks used by wormhole attackers.

Network-based countermeasures must be devised to answer the new challenges. In Packet Leashes, Hu et al. [14] explored geographical distance and synchronized timing to limit a node's data delivery capability in terms of temporal and spatial aspects. This mitigates route disruptions caused by "wormhole" attacks. An approach based on secure distance bounding [7] is proposed by Capkun et al. [32] to detect wormholes without clock synchronization. Hu and Evans [12] proposed another approach using directional antennas. Neighboring nodes examine the directions of the received signals from each other and a shared witness. Only when the directions of both pairs match, the neighbor relation is confirmed. To defeat rushing attackers, Hu et al. [15] proposed to form local communities by a secure neighborhood discovery protocol. In a local community, RREQ forwarding is delayed and randomized so that an RREQ rushing attacker cannot dominate other members during the RREQ phase. Route disruption is mitigated because the chance of selecting a rush attacker on a path equals the chance of selecting a good member. Our self-healing design adopts a different approach. We implement a faster RREQ phase, then in the RREP phase we explore the presence of good network members to heal a damaged ad hoc route on the fly. Such self-healing feature has not been explored in previous secure routing research to counter malicious nodes.

Multi-path routing [26][18] and route fix using local recovery query [29] are alternative choices of community-based routing. In multi-path routing, more paths parallel (albeit some of them are near) to the optimal path are maintained, a damaged path is replaced by another path rather than fixed locally. It incurs extra overheads to maintain paths other than the optimal path and to deliver data on those non-optimal paths. In local recovery query, the forwarders need to cooperatively query a larger recovery area to fix a damaged link. This cooperative assumption does not apply to non-cooperative members studied in this work. In general, we adopt a very different approach to fight against non-cooperative members–we build localized self-healing communities *on* the optimal path to counter non-cooperative nodes. In the context of secure routing, Papadimitratos and Haas [21] studied a multi-path approach to mitigate route disruption attacks. By encoding data packets into erasure codes, the destination is able to recover the source's data upon receiving a threshold subset of encoding symbols that have been delivered along the multiple paths. Awerbuch et al. [3] proposed a multi-path evaluation and probing scheme to detect malicious packet forwarders. If a malicious forwarder cannot differentiate the data packets without probing piggybacks from those with, then the source can pinpoint the range of failure on a path. Nevertheless, none of the related work adopts our localized approach to secure the optimal path discovered by the underlying ad hoc routing protocol.

Local monitoring also improves ad hoc routing security. In secure neighbor detection schemes [15][20], mobile nodes constantly gather knowledge about its current neighborhood. Each node must prove its network membership as well as its local presence. Control and data packets are only forwarded for verified neighbors. As we mentioned in Section III-E, these secure neighborhood detection schemes help community-based routing to subdue attackers with directional transmission capability. In neighbor monitoring, any wireless node can use "watchdog" [19] or passive acknowledgement [17] to detect its neighbors' forwarding misbehaviors. Inside a self-healing community, members monitor each other's behavior. We devised autonomous algorithms to guide each member's actions and reactions upon detecting non-cooperative events. Routing integrity is achieved if at least one cooperative member is monitoring when a routing misbehavior occurs.

VII. Conclusions

In this paper we have studied how non-cooperative network members can threat the secure routing protocols by various means. In particular, they can deplete network resource and reduce the routing performance to minimum. These security threats have not been fully addressed in previous research. We propose the concept of *"self-healing community*" and show how to use this concept to defend against the new security threats. Our design explores redundancy in deployment, an inherent feature of ad hoc networking, to let nearby cooperative "good" network members counter the attacks launched by the non-cooperative nodes.

We rely on localized simple schemes and end-to-end probing to configure and reconfigure "self-healing communities". The localized design realizes a *"localized hospital*" (L'Hospital) to save the (optimal) route discovered by the underlying routing protocol. Ad hoc routes are healed locally within minimal latency. In the ideal case, only a single initial RREQ flood is needed for each end-to-end connection. In practice, even though this ideal case is impractical, the RREQ flooding frequency is minimized. By an analytic model we show the effectiveness gain of community-based secure routing is tremendous. Then we design and simulate secure ad hoc routing protocols to verify the cost and overhead incurred by reconfigurable self-healing communities. Our study shows that it is effective and efficient to use the new paradigm to secure common ad hoc routing protocols.

REFERENCES

[1] I. Aad, J.-P. Hubaux, and E. W. Knightly. Denial of Service Resilience in Ad Hoc Networks. In *ACM MOBICOM*, pages 202–215, 2004.

[2] R. J. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. M. Needham. A New Family of Authentication Protocols. *Operating Systems Review*, 32(4):9–20, 1998.

[3] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *First ACM Workshop on Wireless Security (WiSe)*, pages 21–30, 2002.

[4] C. Bettstetter. Mobility Modeling in Wireless Networks: Categorization, Smooth Movement, and Border Effects. *ACM Mobile Computing and Communication Review*, 5(3):55–67, 2001.

[5] C. Bettstetter, H. Hartenstein, and X. Perez-Costa. Stochastic Properties of the Random Waypoint Mobility Model. *ACM/Kluwer Wireless Networks, Special Issue on Modeling and Analysis of Mobile Networks*, 10(5):555–567, 2004.

[6] C. Bettstetter and C. Wagner. The Spatial Node Distribution of the Random Waypoint Mobility Model. In *German Workshop on Mobile Ad Hoc Networks (WMAN)*, pages 41–58, 2002.

[7] S. Brands and D. Chaum. Distance-Bounding Protocols (Extended Abstract). In T. Helleseth, editor, *EUROCRYPT'93, Lecture Notes in Computer Science 765*, pages 344–359, 1993.

[8] N. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, 1993.

[9] J. Deng, R. Han, and S. Mishra. Intrusion Tolerance and Anti-Traffic Analysis Strategies for Wireless Sensor Networks. In *IEEE International Conference on Dependable Systems and Networks (DSN)*, pages 594–603, 2004.

[10] J. Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, 2002.

[11] L. M. Feeney and M. Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *IEEE INFOCOM*, 2001.

[12] L. Hu and D. Evans. Using Directional Antennas to Prevent Wormhole Attacks. In *Network and Distributed System Security Symposium (NDSS)*, 2004.

[13] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. In *ACM MOBICOM*, pages 12–23, 2002.

[14] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. In *IEEE INFOCOM*, 2003.

[15] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols. In *ACM WiSe'03 in conjunction with MOBICOM'03*, pages 30–40, 2003.

[16] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[17] D. B. Johnson and D. A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), April 2003.

[18] M. K. Marina and S. R. Das. Ad Hoc On-demand Multipath Distance Vector Routing. In *IEEE ICNP*, pages 14–23, 2001.

[19] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *ACM MOBICOM*, 2000.

[20] P. Papadimitratos and Z. J. Haas. Secure Routing for Mobile Ad Hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, 2002.

[21] P. Papadimitratos and Z. J. Haas. Secure Data Transmission in Mobile Ad Hoc Networks. In *Second ACM Workshop on Wireless Security (WiSe)*, pages 41–50, 2003.

[22] C. E. Perkins and E. M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *IEEE WMCSA'99*, pages 90–100, 1999.

[23] C. E. Perkins, E. M. Royer, and S. Das. Ad-hoc On Demand Distance Vector (AODV) Routing. `http://www.ietf.org/rfc/rfc3561.txt`, July 2003.

[24] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes*, 5(2):2–13, 2002.

[25] G. Resta and P. Santi. An Analysis of the Node Spatial Distribution of the Random Waypoint Model for Ad Hoc Networks. In *ACM Workshop on Principles of Mobile Computing (POMC)*, pages 44–50, 2002.

[26] P. Sambasivam, A. Murthy, and E. M. Belding-Royer. Dynamically Adaptive Multipath Routing based on AODV. In *Med-Hoc-Net*, 2004.

[27] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. Royer. A Secure Routing Protocol for Ad Hoc Networks. In *10th International Conference on Network Protocols (IEEE ICNP'02)*, 2002.

[28] Scalable Network Technologies (SNT). QualNet. `http://www.qualnet.com/`.

[29] C. Sengul and R. Kravets. Bypass Routing: An On-Demand Local Recovery Protocol for Ad Hoc Networks. In *Med-Hoc-Net*, 2004.

[30] E. Shih, P. Bahl, and M. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *ACM MOBICOM*, pages 160–171, 2002.

[31] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, 1997.

[32] S. Čapkun, L. Buttyán, and J.-P. Hubaux. SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 21–32, 2003.

[33] M. G. Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *First ACM Workshop on Wireless Security (WiSe)*, pages 1–10, 2002.