Impact of Denial of Service Attacks on Ad Hoc Networks

Imad Aad
DoCoMo Euro-Labs
Munich, Germany

Jean-Pierre Hubaux
EPFL
Lausanne, Switzerland

Edward W. Knightly
Rice University
Houston, TX

***Abstract*—Significant progress has been made towards making ad hoc networks secure and DoS resilient. However, little attention has been focused on quantifying DoS resilience: Do ad hoc networks have sufficiently redundant paths and counter-DoS mechanisms to make DoS attacks largely ineffective? Or are there attack and system factors that can lead to devastating effects? In this paper, we design and study DoS attacks in order to assess the damage that difficult-to-detect attackers can cause. The first attack we study, called the JellyFish attack, is targeted against closed-loop flows such as TCP; although protocol compliant, it has devastating effects. The second is the Black Hole attack, which has effects similar to the JellyFish, but on open-loop flows. We quantify via simulations and analytical modeling the scalability of DoS attacks as a function of key performance parameters such as mobility, system size, node density, and counter-DoS strategy. One perhaps surprising result is that such DoS attacks can *increase* the capacity of ad hoc networks, as they starve multi-hop flows and only allow one-hop communication, a capacity-maximizing, yet clearly undesirable situation.**

I. Introduction

Significant progress has been made in securing ad hoc networks via the development of secure routing protocols [1], [2], [3], [4], [5]. Moreover, ensuring resilience to misbehavior and denial-of-service attacks has also been the focus of significant research efforts as such resilience is a critical component of a secure system: examples include "watch-dog" mechanisms designed to detect and circumvent misbehaving nodes [6]; rate-limiting of route-request messages to prevent route query-flood attacks [4]; and "rushing attack prevention" that seeks to inhibit malicious nodes from attracting an excessive number of routes, which would increase their ability to inflict damage [7].

Yet, there remains an indefinite "arms race" in system and protocol design: attackers (or researchers anticipating the moves of attackers) will continually introduce increasingly sophisticated attacks, and protocol designers will continually design protocol mechanisms designed to thwart the new attacks.

The goal of this paper is to quantify via analytical models and simulation experiments the damage that a *successful* attacker can have on the performance of an ad hoc network. In particular, we recognize that successful attacks are inevitable (at least until the corresponding counter-DoS protocol modification is deployed), and our objective is to characterize the relationship between the resources that must be commandeered by the attacker (the percentage of nodes in an ad hoc network used in the attack) and the impact on performance of non-attacking nodes, where performance refers to per-flow goodput and system-wide fairness. In this way, we study the scalability of DoS attacks and identify the key mechanisms and factors of both attacks and protocols that affect a system's DoS resilience.

Our methodology is to study DoS resilience via a new and general class of *protocol compliant* denial-of-service attacks, which we refer to as *JellyFish* (JF). Previously studied attackers *disobey* protocol rules; on the contrary, JellyFish conform to all routing and forwarding protocol specifications, and moreover, as implied by the name, are passive and difficult to detect until after the "sting." JellyFish target *closed-loop* flows that are responsive to network conditions such as delay and loss. Examples include TCP flows and congestion-controlled UDP flows employing a TFRC-like algorithm [8].

The goal of JF nodes is to reduce the goodput of all traversing flows to near-zero while dropping zero or a small fraction of packets. In particular, JF nodes employ one of three mechanisms. The first JF variant is a packet reordering attack. TCP has a well-known vulnerability to reordered packets due to factors such as route changes or the use of multi-path routing, and a number of TCP modifications have been proposed to improve robustness to misordering [9], [10], [11], [12].

However, *no* TCP variant is robust to *malicious* and persistent reordering as employed by the JF misordering attack. The second JF mechanism is periodic dropping according to a maliciously chosen period. This attack is inspired by the Shrew attack [13] in which an endpoint sends maliciously spaced periodic pulses in order to force flows into repeated timeout phases [13]. The JF periodic dropping attack utilizes the same principles but realizes the attack via periodic dropping at relay nodes. In particular, suppose that congestion losses force a node to drop x% of packets. As shown in [13], if these losses occur periodically at the retransmission-time-out timescale (approximately 1 second), TCP throughput is reduced to near zero even for small values of x. Thus, a JF periodic-dropping node can drop no more packets than neighboring congested nodes, but inflict near-zero throughput on all TCP flows traversing it. Third, we consider a delay-variance attack in which the attacker delays packets (preserving order) in order to thwart TCP's timers and congestion inferences. This attack not only thwarts widely deployed TCP variants, but also can disrupt rate-based congestion control algorithms such as [14], [15]. Notice that JF nodes are *protocol compliant* in that IP's datagram service does not mandate loss-free service, in-order delivery, or bounded delay jitter.

Finally, in addition to the JF attack, we also study the "black hole" attack as described in [4]. This attack is relevant for *open-loop* flows that do not respond to congestion, loss, or delay information, and hence cannot be thwarted by JellyFish. Black Hole nodes participate in the routing protocol to establish routes through themselves, yet drop all packets after correctly receiving them at the MAC layer.

With these attacks (three JF variants and Black Holes), we use a combination of analytical modeling and simulation experiments to study the key performance factors that determine a network's DoS resilience and equivalently, the attack's scalability.

Throughout, we consider that victims will diagnose and react to DoS attacks. Thus, we quantify the relationship between the timescale of diagnosis and reaction to the attacker as compared to the route lifetime. Intuitively, if a system has no mobility (and infinite route lifetimes), JF will have little effect as nodes will eventually discover routes without JF if such routes exist. However, as mobility increases, the route lifetime shortens and the effects of JF become increasingly pronounced as the time spent uselessly transmitting on JF paths and re-establishing routes becomes an increasing fraction of a flow's lifetime. Thus, we derive an analytical and experimental relationship that characterizes the impact of these timescales on flow goodput.

Several reputation systems have been developed to help traffic sources and forwarders avoid misbehaving nodes and route breaks. Using multipath routing is another method to make end-to-end throughput more robust against route breaks. We model the impact of reputation mechanisms and multipath routing on end-to-end throughputs and we show that the enhancement is negligible whenever we consider realistic system parameters.

Finally, we study a number of system factors that affect a network's DoS resilience and obtain the following findings. (i) JF have a network partitioning effect that severely degrades or altogether prevents long-range communication. Consequently, an increased number of JF reduce the system's fairness index but may *increase* network capacity, as capacity can be increased by starving long-range flows and serving only one-hop flows. (ii) The mean and distribution of path length have a significant effect on attack scalability as higher path length flows are highly vulnerable. (iii) JF are most devastating in a system with a well balanced offered load. If a system is heavily overloaded, system performance is already so poor (high path length flows are already starved), that JF have little marginal impact. (iv) Random or mobile JF placement performs nearly identical to optimal-coverage JF placement in systems with even a small number of JF. (v) JF are most effective in moderate to high density networks as excessively low density networks may already be partitioned and JF can do little marginal damage. (vi) The scaling of the attack with the percentage of JF remains largely unaffected for large vs. small scale networks. Yet, the absolute performance is quite different, as without attack, small scale network performance is significantly better than large scale network performance.

Thus, our goal is not to advance the aforementioned "arms race" by developing attacks, victim counter strategies, counter attacks, etc., but rather to explore the impact of a class of attacks that are difficult and time consuming to detect due to their compliance to all protocol rules. Yet, we do consider that bad paths will indeed be diagnosed by victims and routed around (as will be the case with the JF attack or other yet-to-be-invented attacks) and we study the key performance factors for attack scalability.

The remainder of this paper is organized as follows. In Section II we present the JF attacks and an example of their effects on throughput. In Section III we present a simple analytical model that relates system properties such as mean-path-duration and mean-path-length to the victim's throughput. In Section IV we perform extensive simulation experiments to quantify the factors that control an attack's scalability. Finally, in Section V we review related work and in Section VI we conclude.

II. JellyFish and Black Hole DoS Attacks

A. System Model

Unless otherwise specified, we consider a general mobile ad hoc network employing a broad set of security and DoS resilience mechanisms that (i) ensure node authentication, (ii) ensure message authentication, (iii) ensure one identity per node (preventing Sybil attacks), and (iv) prevent control plane misbehavior (query floods, rushing attacks, etc.).

Examples of protocols that achieve the above objectives are discussed in Section V, but for concreteness, we can consider a secure source routing protocol as in reference [4] as well as enhancements such as [16], [7]. Throughout the paper and especially in Section III, we discuss the implications of such enhancements, as well as other counter DoS mechanisms.

The effects of the DoS attacks we describe are independent of the considered MAC layer protocol. However, in our simulations, we consider the MAC layer to be IEEE 802.11.

A fraction of nodes are malicious and seek to thwart system performance. A malicious node will always participate in route setup operations. For example, if source routing is employed, malicious nodes always relay Route Request packets in order to have as many routes as possible flowing through themselves; if distance vector routing is employed, malicious nodes will also obey all control-plane protocol specifications. However, once a route is established, attacking nodes will thwart the end-to-end throughput of the flow via a JellyFish or Black Hole attack. While packets may be encrypted at higher layers and become "unrecognizable" (e.g., TCP vs. UDP) to the network layer, the JellyFish and Black Hole attacks can still be applied irrespective of the packet types.

B. JellyFish Attack

A critical strength of the JellyFish Attack is that it maintains compliance with *all* control plane and data plane protocols in order to make detection and diagnosis costly and time consuming. The key principle that JF use to facilitate the attack is targeting end-to-end congestion control. In particular, many applications such as file transfer, messaging, and web will require reliable, congestion-controlled delivery as provided by protocols such as TCP. Moreover, TFRC-controlled real-time applications such as interactive video must also adapt their rates to available bandwidth and hence also employ end-to-end congestion control.

The dual role of hosts as routers in ad hoc networks introduces a critical vulnerability for congestion control: specifically, there are a number of *forwarding* behaviors that routers (ad hoc relay nodes) can employ that will severely degrade the end-to-end throughput of congestion-controlled traffic. We refer to these behaviors as variants of the JellyFish attack, which we describe as follows.

JF Reorder Attack. TCP's use of cumulative acknowledgements defines the message "ACK-N" to indicate that *all* segments $1, \ldots, N$ have been received. Consequently, receipt of duplicate ACKs is used to infer loss. Yet, because duplicate ACKs can also indicate an out-of-order packet receipt, TCP has a number of mechanisms to increase its robustness to out-of-order packets, including TCP Sack [17] and reorder robust TCP [12]. Yet, all such TCP variants assume that reordering events are rare, short-lived, and due to network events such as route changes.

In contrast, we consider JF nodes to maliciously reorder packets. In this attack, JF deliver *all* packets, yet after placing them in a re-ordering buffer rather than a FIFO buffer. Consequently, we will show that such persistent re-ordering of packets will result in near zero goodput, despite having all transmitted packets delivered.

JF Periodic Dropping Attack. Losses due to buffer overflow are inevitable in congested environments. Kuzmanovic and Knightly [13] show that if such losses occur periodically near the retransmission time out (RTO) timescale (in the 1s range as RTO is intended to address severe congestion), then end-to-end throughput is nearly zero. An *endpoint* attack is described in [13] in which a malicious node transmits periodic pulses into the network. As the RTO-spaced pulses can force all flows sharing the bottleneck link to enter repeated timeout phases, the attack results in all such flows obtaining near-zero throughput while the attacker has a low average transmission rate. The study showed that the impact of the attack can be quite severe whether minimum RTO values are all set to 1 second as recommended in [18], or are randomized over a wide range.

Here, we utilize the same principle for the JF periodic dropping attack in which attacking nodes drop all packets for a short duration (e.g., tens of ms) once per RTO. Thus, unlike [13], JF are passive and generate no traffic themselves; like non-malicious nodes, JF drop for only a small fraction of time; yet, with this dropping pattern during a maliciously chosen period, the following behavior results. Upon encountering the JF's first loss duration, the victim flow will enter timeout as the JF chooses

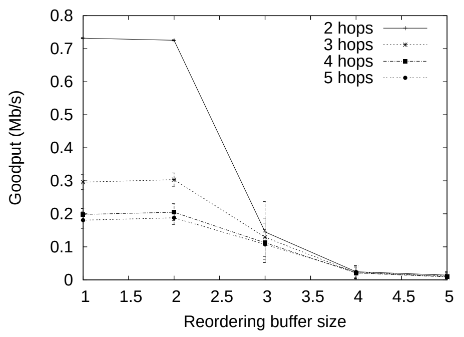

Fig. 1. JF-reorder effect on throughput

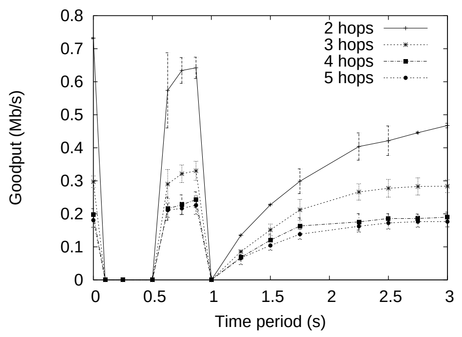

Fig. 2. JF-drop effect on throughput

the dropping duration to be sufficiently long to result in multiple losses. When the flow attempts to exit timeout RTO seconds later, the JF will immediately or soon after periodically drop again. Note that the JF knows when a flow enters timeout as the JF itself induced the loss. Thus, the JF can safely assume that by RTO seconds later, the flow will be attempting to exit and will be in the fragile slow-start state.

JF Delay Variance Attack. Variable round-trip-times due to congestion are an inevitable component of TCP's operation. Yet, ensuring high performance in the presence of random and high delay variation due to an *attacker* was clearly not incorporated into TCP's design. Such a high delay variation can (i) cause TCP to send traffic in bursts due to "self-clocking," leading to increased collisions and loss, (ii) cause mis-estimations of available bandwidth for delay-based congestion control protocols such as TCP Westwood and Vegas, and (iii) lead to an excessively high RTO value.

Indeed, enhancing TCP to combat the effects of *non-malicious* delay variation to wireless links has been the focus of intense research (see [19] for example), as has the development of tools for available bandwidth estimation. Consequently, *malicious* manipulation of packet delays by the JF delay variance attack has the potential to significantly reduce TCP throughput. Such attackers therefore wait for a variable amount of time before servicing each packet, maintaining FIFO order, but significantly increasing delay variance.

C. Impact of JF

We next present simulation experiments that illustrate the effects of JF on end-to-end goodput. To study these effects in isolation, we consider a simple "chain" scenario with a sequence of nodes between the sender and receiver, one of which is a JF. We use TCP Sack, the default IEEE 802.11 MAC at 2 Mb/s, and show the 95% confidence intervals over 10 simulation runs.

Figure 1 shows the impact of the JF-reorder attack on the TCP-Sack flow for different re-ordering. This experiment has a scheduler that is a FIFO queue, except that it selects randomly among the first k packets in the queue. The figure depicts performance as a function of the re-ordering buffer size expressed in packets.

The figure indicates that TCP is robust to moderate reordering with a reordering buffer of 2 packets. Whereas, when the reordering buffer is larger and the reordering is performed in this persistent and malicious way, TCP throughput collapses. For example, consider the curve with 3 nodes and a 2-hop chain, i.e., a source, destination, and a single relay node. Without an attack (a reordering buffer of 1), the flow obtains a throughput of 710 kb/s. Yet, with a reordering buffer of 3 or more packets, the throughput decreases to approximately 1% of the peak value indicating a successful attack and near starvation of the flow. That is, if the scheduler selects the next packet to service randomly among the first 3 or more queued, the resulting reordering cannot be overcome by TCP. We note that solutions to TCP reordering such as TCP-PR [10] use only timers to detect loss vs. duplicate ACKs. Thus, attackers would need to either use other JF variants for TCP-PR flows or use larger reorder buffers to force TCP-PR timeouts.

Figure 2 depicts the results of simulation experiments with the JF periodic dropping attack. Consider first the upper curve in which the path consists of a source, a single relay node (a JF), and a destination. A time period of 0 indicates no attack and the flow again obtains a throughput of 710 kb/s. As in [13], the degradation in throughput to the victim is highly non-linear as a function of the dropping period, with null frequencies near 0.5 and 1 second (the minimum RTO value). To obtain the null at 1 second, the JF drops packets for 90 ms every 1 second, which results in dropping 9% of the time, and forwarding 91% percent of the time, values easily incurred by a congested node.

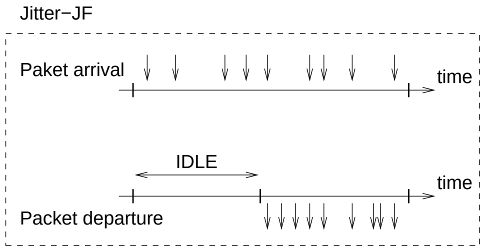

Fig. 3. The jitter implementation used

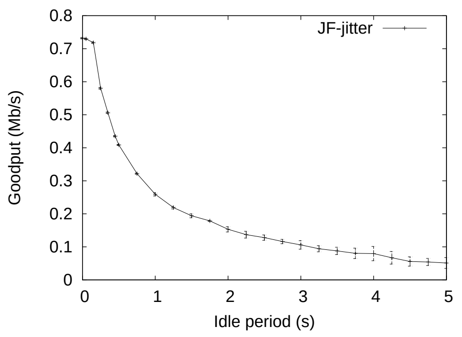

Fig. 4. JF-jitter effect on throughput

The attack is therefore successfully exploiting the slow-time-scale congestion avoidance mechanism of TCP, namely, that flows must infer that multiple packet losses within a round-trip-time are an indication of severe congestion, such that the flow must back off aggressively, and wait RTO seconds before entering slow start. Significantly reducing RTO or removing the mechanism all together would lead to significant spurious retransmissions and potentially congestion collapse [18], whereas increasing the value would make the attack even more devastating.

Finally, we show how an intermediate JF can attenuate the TCP throughput by varying the RTT.

The jitter implementation we used is shown in Fig. 3. In this scenario, the JF behaves as a server with vacations, alternating between periods of serving no packets (and queuing, but not dropping them) and serving packets at its maximum capacity. Both idle and active periods are of equal lengths. Packet departure times are proportional to their arrival times. We deploy this jitter-JF in a three node chain.

Figure 4 shows how TCP goodput decreases with increasing jitter (i.e., increasing idle and active periods). While this decreased throughput is also due to increased mean delay, the figure nonetheless indicates that the effects of this attack can be quite severe.

D. Black Hole Attacks

We also consider Black Hole attacks as described in [4]. As with JF, Black Hole nodes participate in all routing control plane operations. However, once paths are established, Black Holes simply drop *all* packets. Although refusing to forward data is *not* protocol compliant, we also study Black Holes for the following reasons. First, as demonstrated in the simulations above, JF have nearly the same *impact* as Black Holes, making end-to-end throughput collapse until the victim detects and fixes the problem. Thus, in many simulation experiments, we will consider Black Holes in place of JF for simplicity. Second, Black Holes allow us to study flows that are *not* congestion controlled and therefore are immune to JF. Thus, we can still study attack scalability for open-loop flows that ignore the delay, ordering, and loss information that JF are manipulating.

E. Misbehavior Diagnosis

The broadcast nature of the wireless medium can help detecting misbehavior or node failure. This is the case of PACK (Passive Acknowledgement) [20] which exploits the fact that an upstream neighbor can overhear a node's transmission and diagnose failure/misbehavior. Unfortunately, PACK has several key limitations that preclude its use as a general solution to attacks such as JF, as we showed in [21].

From the end-to-end point of view, victims of the attacks will measure that they have near zero throughput and will react. Likewise, a malicious node's neighbors may attempt to diagnose failed paths due to DoS behavior. Clearly, the attacker seeks to inhibit diagnosis in order to maximize damage. Thus, in [21] we explore network and endpoint mechanisms for DoS diagnosis with a focus on their practical feasibility, as well as on the timescales for successful diagnosis and repair. In Section III we quantify the effects of these timescales on flow goodput and in Section IV we experimentally explore this factor.

F. Victim Response

Once a path is diagnosed as providing zero throughput, the end points will attempt to establish an alternate path. With uni-path source routing, this will be achieved via transmission of a new route request message, typically from the source. When route replies are received, the victim should avoid paths with *any* node from the prior malfunctioning path as the victim does not know which node on the path was malicious, i.e., the victim has insufficient information to form an accurate "black list." Furthermore, note that as JF are protocol compliant, the victim is not certain whether throughput collapsed due to an attacker or simply due to congestion, fading, or other factors incurred in normal protocol operation.

An alternate solution is to employ multipath routing, and to adapt the path weights according to path goodput as proposed in [4], [22]. Even without attackers, such a protocol must overcome the impact of different paths having different delay characteristics and the corresponding impact on TCP throughput. For example, reference [23] found that TCP Sack's use of multiple paths in ad hoc networks led to a severe throughput reduction for even two paths, and near collapse for three or more paths. The authors then suggest a re-design of TCP to support multipath routing.

Other promising counter-measures would be the establishment of backup routes, e.g., caching of all route reply messages for later use if a current path fails.

In any case, even with multipath routing and TCP re-design, use of backup routes, etc., a victim flow will always encounter the issues we study next: delays to diagnose and react to the problem, and poor throughput until all forwarding paths are free of JF.

III. Analytical Model

In this section, we develop a simple model to predict the throughput of a flow traversing a network in the presence of attacking nodes.

Consider an ad hoc network with N nodes and $a < N$ attacking nodes (JellyFish or Black Holes). Denote p as the probability that a randomly selected node is an attacker such that $p = a/N$. (We also discuss other relationships between a, N, and p below.) Consider a path traversing h relay nodes. If the selected nodes represent a random sample of the N network nodes, then the path contains no attacking nodes with probability $(1-p)^h$.

We compute the throughput via a renewal argument in which time alternates between periods of successful transmission and periods of thwarted transmissions and assume that such durations are independent and identically distributed. In particular, we denote $E(T_L)$ as the expected lifetime of a route as determined by factors such as the node velocity and node density.

When a route breaks due to mobility, a number of delays are incurred in repairing the route. First, a duration T_{diag} is incurred to diagnose that the route is broken. Next, the request for a new route may be delayed by a rate-limiting duration in order to mitigate the impact of route query flood attacks. We denote this rate-limiting time as T_{RL}, which denotes the minimum inter-spacing of route requests allowed by the routing protocol. Finally, the node must wait to receive one or more route reply messages, a duration that we denote as T_{RR}.

After these three phases, a node begins transmitting data on the new path. However, the new path includes at least one attacking node with probability $1-(1-p)^h$. If this is the case, the transmission is thwarted and the node must again incur the above three delays and try again. Note that even if the victim has ensured that the new route contains no nodes in common with a failed route, the new route may again contain an attacking node. Thus, a node exits the zero-throughput phase only after it has successfully established a route without an attacking node.

In general, a protocol may change timers according to the number of attempts. Thus we denote superscript j as the attempt number such that for example T_{RL}^j denotes the rate-limiting duration waited immediately before the j^{th} attempt. Thus, we have that the total expected time of zero throughput, i.e., the time to find a new route that contains no attacking node, is given by

$$E(T_0) = T_{RR}^0 + \sum_{t=1}^{\infty} \left(\sum_{j=1}^{t} \left[E(T_{diag}^j) + E(T_{RL}^j) + E(T_{RR}^j) \right] \right) \times (1-p)^h \left(1-(1-p)^h\right)^t . \quad (1)$$

More generally, the path length can be represented by a random variable H such that $E(T_0) = \sum_{h \geq 0} E(T_0|H) Pr(H = h)$ using Equation (1) for $E(T_0|H)$ along with the distribution of H. To simplify, we consider a fixed path length ($H = h$) unless otherwise noted, and consider the further simplification $E(T_{diag}^i) = E(T_{diag}), E(T_{RL}^i) = E(T_{RL})$ and $E(T_{RR}^i) = E(T_{RR}), \forall i$ such that we have

$$E(T_0) = T_{RR} + \sum_{t=1}^{\infty} t[E(T_{diag}) + E(T_{RL}) + E(T_{RR})] \times (1-p)^h \left(1-(1-p)^h\right)^t \quad (2)$$

which under the above assumptions reduces to

$$E(T_0) = T_{RR} + [E(T_{diag}) + E(T_{RL}) + E(T_{RR})] \times [\frac{1}{(1-p)^h} - 1]. \quad (3)$$

The normalized goodput for a flow is given by

$$G = \frac{E(T_L)}{E(T_L) + E(T_0)} \quad (4)$$

We make several observations about Equations (3) and (4). First, note the corner case with p approaching 1 or high route length send goodput to 0. Another corner case is a scenario with no mobility: in this case, once a successful route is established, it is never subsequently broken and goodput approaches 1.

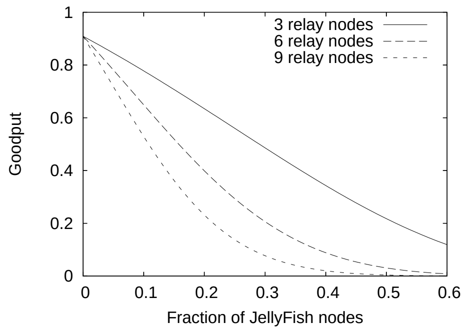

Fig. 5. Attack scalability and path length

Intermediate cases are depicted in Figure 5, which illustrates goodput (computed from Equation (4)) as a function of the percentage of attackers $p = a/N$ for three route lengths of 3, 6 and 9 relay nodes. We consider a mean route lifetime of $E(T_L) = 10$ s, which corresponds to a high node velocity of Vmax = 30 m/s as reported in [24]. Moreover, the curves depict the case that the diagnosis, rate limit, and route reply times are 2 s, 2 s, and 1 s respectively. The diagnosis time is set to two times the default retransmission timeout value for TCP: a lower value would certainly lead to false inference of broken routes. The 2 second rate limit value is the default value for DSR for the minimum spacing of route requests, which is increased in DSR to 10 seconds for subsequent requests (not considered here).

The figure indicates that without any attacking node, legitimate nodes spend approximately 90% of their time successfully transmitting, and the remaining 10% having broken routes and trying to re-establish them. Next observe the scalability of the attack for 6 relay nodes: with 10% of attacking nodes, the goodput drops to 65%, whereas with 20% of attacking nodes, the goodput drops to 40%. The impact of the attacker is even more pronounced in large-scale networks in which a longer path length is increasingly likely to include an attacking node. For example, with 9 relay nodes, the goodput decreases to 53% under 10% attacking nodes and to 23% under 20% attacking nodes.

The model also allows us to explore the impact of a "Rushing Attack" [7] as we showed in [21].

A. Performance of reputation systems against JF attacks

To help thwarting misbehavior in ad hoc networks, several reputation systems have been proposed in the literature [25], [26]. In this section we evaluate the ability of such systems to avoid the JF. In order to abstract from the technical details, we assume that the reputation system can be modeled as a black box with two performance parameters:

		Reputation	
		Good	Bad
Actual	Good	$g(1-f_p)$	gf_p
	Bad	$(1-g)f_n$	$(1-g)(1-f_n)$

TABLE I
ALL COMBINATIONS OF REPUTATION AND ACTUAL STATE OF NODES.

- *False positives* (f_p): This is the rate at which the reputation system reports well-behaved nodes as being malicious.
- *False negatives* (f_n): This is the rate at which the reputation system reports a malicious node as being well-behaved.

Assume that the system has a proportion g of well-behaved nodes. The various combinations of good/bad choices are depicted in Table I. For instance, there is a probability of gf_p to select a good node with bad reputation.

When establishing a new route, the source and the forwarding nodes try to select well-behaved nodes only (Reputation: *Good*). However, they may mistakingly choose, with probability f_n, one of the $(1-g)$ actually *Bad* nodes. Therefore $(1-g)f_n$ replaces p in (3), i.e.

$$G_{reput} = \frac{E(T_L)}{E(T_L) + E(T_0^{reput})}. \quad (5)$$

where

$$E(T_0^{reput}) = T_{RR} + [E(T_{diag}) + E(T_{RL}) + E(T_{RR})] \times [(1-(1-g)f_n)^{-h} - 1]$$

Figure 6 shows that using a reputation system with $f_n = 0.1$ (i.e. only one tenth of misbehaving nodes

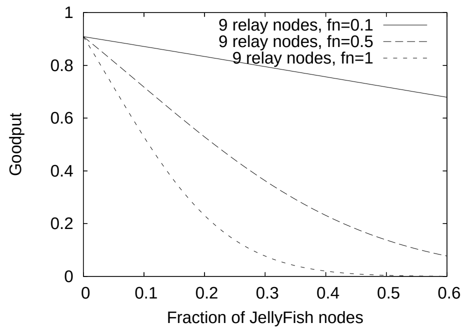

Fig. 6. Impact of false negatives of a reputation system

are likely to be selected during route establishment) enhances the goodput with respect to a system that does not rely on reputation ($f_n = 1$). In fact, when $f_n = 1$, misbehaving nodes and well-behaved nodes are selected with equal probability, i.e. the reputation system has no impact on node selection during route setup.

On the other hand, the false positives rate f_p has a negative impact, due to the fact that the source and the forwarding nodes will avoid actually good nodes (g), with a bad reputation (f_p), during route establishment. This reduces the number of possible paths by a factor of:

$$\frac{g(1-f_p)}{g} = (1-f_p) \qquad (6)$$

with respect to a system that does not use any reputation mechanism. Therefore, when $f_p = 1$ (i.e. all good nodes are judged to be bad), the factor expressed by (6) reduces to zero, and no route can be established. When $f_p = 0$, the reputation system does not mislead the route establishment procedure, therefore the factor in (6) is equal to 1, i.e., the performance is similar to the one of a system with no reputation mechanism.

B. Using multipath routing

In this section we analyze the performance of using multipath routing [27] to thwart JF attacks: Upon route establishment, the source keeps a list of possible routes, not necessarily optimal ones, to the destination. The source either uses several routes simultaneously, or changes route upon diagnosing a problem on a given path. We consider the best case scenario where there is always at least one unbroken path (no need for route re-establishment). This increases the throughput in equation (4) to:

$$G_{multipath} = \frac{E(T_L)}{E(T_L)+E(T_{RR})+E(T_{diag})\times[(1-p)^{-h}-1]}. \qquad (7)$$

That is, a single route request is issued (per flow lifetime), and therefore route request limitations are not considered.

Figure 7 shows that using multiple paths improves the flow throughput with respect to a system using single paths, however it is still weak in thwarting JF attacks. More importantly, one should also take into consideration the negative effect of multi-path routing (therefore packet reordering) on TCP (not considered in the model). The resulting overall impact of packet reordering on TCP goodput due to the use of multipath routing would be definitely negative.

IV. Assessment of Performance Under DoS Attack

In this section, we perform an extensive set of simulation experiments to quantify the impact of DoS attackers on the system performance and to identify the key factors that determine an attack's scalability. After describing our methodology, we establish a baseline case and then isolate the impact of each factor.

A. Methodology

Attackers affect performance in a number of ways. The performance metrics below allow us to evaluate the impact of JF on individual flows, as well as on the whole system performance.

- *System fairness*: To measure fairness, we use Jain's fairness index computed using long-term throughput averages and given by [28]:
$$F_J = \frac{(\Sigma_{i=1}^{m}\gamma_i)^2}{m\Sigma_{i=1}^{m}\gamma_i^2} = \frac{1}{m\Sigma_{i=1}^{m}\gamma_i^2}$$
where m is the total number of flows and γ_i is the proportion of received packets of flow i during the simulation time. F_J is equal to 1 when all flows equally share the network, and is equal to $1/m$ when a single flow monopolizes all resources (in which case $F_J \to 0$ when $m \to \infty$).
- *Number of hops for received packets*: We consider random topologies with random traffic matrices. However, JF and Black Holes can have the effect of starving multihop flows and giving all the capacity to one-hop flows that (by definition) have no relay nodes and hence do not encounter JF. This performance measure captures this effect and also characterizes network partitioning in which multihop communication becomes impossible.
- *Total system throughput*: This measure characterizes the received throughput aggregated over all network flows. Providing all capacity to one-hop flows and

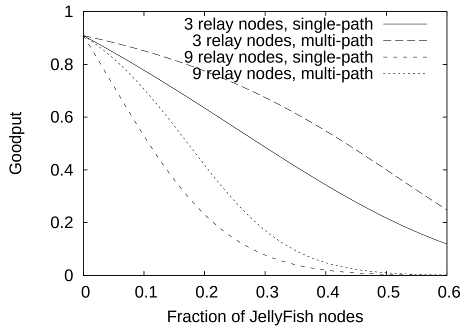

Fig. 7. Enhancement due to the use of multipath routing

starving others can be the capacity-maximizing allocation of bandwidth to flows. Thus, JF and Black Holes often increase total system throughput.

- *Probability of interception*: This characterizes the probability that a flow encounters a JF in its path. This probability depends on many factors such as the placement of JF, the traffic patterns, the percentage of JF etc. Moreover, all the previously mentioned performance metrics depend on this probability.

Experimental and simulation results showed that delays and jitters in ad hoc networks vary considerably. Therefore they provide no relevant information to be considered in our analysis.

An attack's effectiveness is a function of a number of system parameters. We consider the offered load, the congestion control protocol, and the JF placement strategy. We next assess the effect of these parameters on the performance metrics described above by varying them one at a time. We show the effect of other system parameters in [21]. We use *ns-2.27* [29] simulations and present results averaged over 50 simulation runs, using 18 different topologies / mobility scenarios (8000 simulations in total)[30]. We show the corresponding 95% confidence intervals. Each simulation is 500 s, and results are obtained after a warmup period of 100 seconds. Unless otherwise specified, we use Black Holes to emulate the effects of JellyFish on TCP, as the latter were shown in Section 2 to result in near-zero throughput, resulting in a near identical effect as Black Holes. Moreover, JF can have a slightly stronger effect: for example, with JF, re-ordered and delayed packets are still transmitted end-to-end, consuming additional capacity while not contributing to goodput. To simplify the presentation, we designate the attacking nodes as JF throughout the section; in practice, however, they are Black Hole nodes in the case of UDP flows and JellyFish nodes in case of TCP flows. Degraded channel conditions (e.g. noise, fading etc.) are harmful components to the system performance. Therefore we consider a clear non-fading channel to assess the impact of the JF attacks.

B. Baseline

For the baseline simulations, we consider a scenario in which 200 nodes move randomly (random waypoint model) in a 2000 m $\times$ 2000 m topology, at a maximum velocity of 10 m/s, pausing for 10 s on average. Nodes use the IEEE 802.11 MAC with a node receive range of 250 m. The channel capacity is 1 Mb/s. 100 of these nodes communicate with each other to create 50 flows. UDP packets are transmitted at a constant rate of 800 bits/s, corresponding to one 500-byte packet every 5 s. The other 100 nodes route packets without generating any flows, and are henceforth called "routers." JF are compromised routers among these 100. For the baseline, JF are statically placed on a grid at equal distances from each other. Without loss of generality, DSR [20] is used for ad hoc routing.

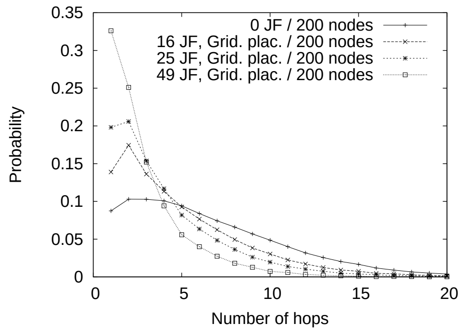

Fig. 8. Distribution of the number of hops for received packets

Figure 8 shows that in the absence of JF, one-hop flows account for approximately 8% of received packets, with the remaining packets nearly uniformly allocated to flows up to 5 hops, and then longer-path-length flows accounting for significantly less.
(Note that there is a smaller number of flows having very long paths due to the random traffic matrix.)

However, with 25 JF (12.5% of nodes), the percentage of received packets corresponding to one-hop flows increases to 20%, and with 49 JF (25% of nodes), the percentage increases to 33%. In each case, this advantage to one-hop flows comes at the cost of multihop flows. For example, under 25 JF, 5 hop flows have their throughput cut in half and 10-hop flows become nearly starved. This indicates that the attack has nearly prohibited long-range communication such that the network is in effect partitioned, allowing only short-range communication.

Figure 9 shows the impact of JF on system fairness.

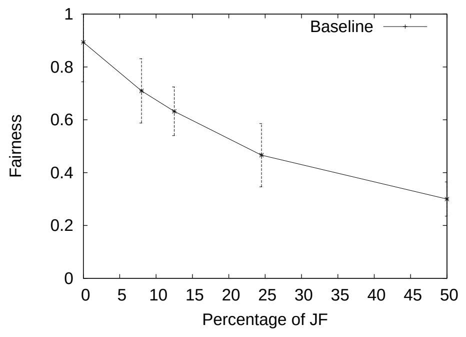

Fig. 9. Fairness index for the baseline case

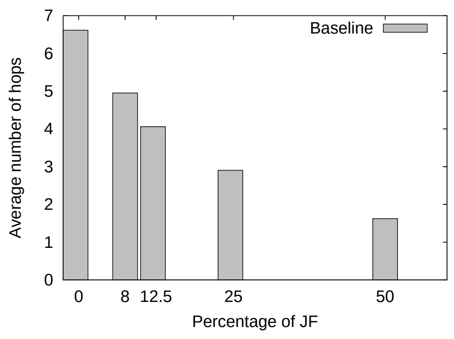

Fig. 10. Average number of hops for received packets

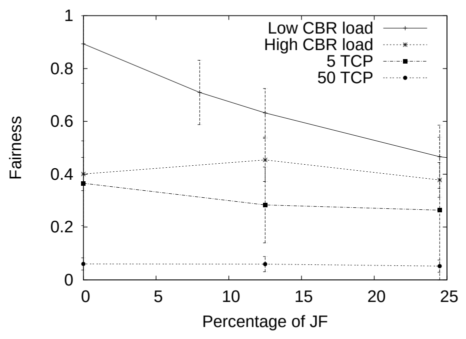

Fig. 11. Effect of offered load

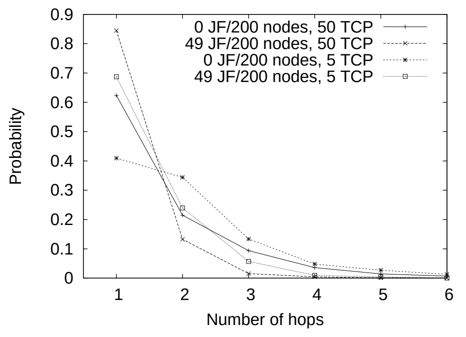

Fig. 12. Hops for received packets with different TCP loads

Observe that with no JF, the system has a relatively high fairness index of 0.9 indicating that flow rates are not significantly different. However, with an increasing proportion of JF in the network, the fairness index significantly decreases, indicating that some flows are obtaining a significantly higher throughput share at the expense of other flows.

Figure 10 explains the phenomenon. The figure depicts the mean hop length for a *received* packet. Without attack, the mean is 6.6 indicating that a significant number of packets are received on long-path-length routes. Yet, as the number of JF grows, the average path length for a received packet diminishes: fewer and fewer packets are able to traverse long routes leading to increased capacity for one-hop flows. Figure 8 illustrates the unfairness: long paths are increasingly likely to be intercepted by JF, considerably reducing their share of the system capacity, whereas the short-path flows “benefit” from the attack.

C. Offered Load and TCP

The system’s offered load is an important factor for the scalability and impact of the JF attack. At one extreme, if the offered load is very high, most packets received end-to-end will be over one hop flows even without the attack, so that JF can do little if any additional damage. At the other extreme, with a more moderate load, JF will skew the distribution of received traffic more towards that achieved in an over-load case.

To study this effect, we consider an offered load per flow of 5 times that of the baseline. Moreover, we consider the offered load that TCP will achieve for 5 and 50 TCP flows. The rest of the parameters remain the same as in the baseline scenario.

The curve in Figure 11 with an offered load of 5 times that of the baseline case illustrates that an overloaded network has a fairness index of 0.4 without any JF, even below that obtained under the baseline load with 25% JF. Thus, there are too few multihop flows for the JF to even slightly degrade the fairness index, i.e., repeated collisions and buffer overflow severely impede multihop traffic.

For TCP traffic, TCP congestion control does not attempt to provide equal throughput to all flows (which would achieve a fairness index of 1). Instead, it seeks to provide throughput that is inversely proportional to round-trip-time. However, the situation with 50 TCP flows is quite similar to that of the CBR overload case: the JF have little effect on fairness, as one-hop flows are dominating the percentage of packets received end-to-end, even without the attack.

With 5 TCP flows, Figure 12 indicates that without the attack, 40% of received packets are from one hop flows whereas with 49 JF, this percentage increases to 69% of received packets. Thus, the attack increased the number of one-hop packets by 73%, resulting in a significant impediment to multihop traffic.

Next we measure the total system throughput as a function of the percentage of JF and present the results in Figure 13. For the baseline case, the figure shows that an increasing percentage of JF results in progressively lower system throughput as an increasingly high number of flows become thwarted by the attack for the reasons discussed above. However, the results are quite different under 5$\times$ system load and for 50 TCP flows. For the case of 5$\times$ system load, the total system throughput has nearly doubled under 12.5% of attackers when compared to no attackers, thus indicating that a DoS attack can *increase* the capacity of an ad hoc network. Although initially surprising (at least to the authors), the reason is quite simple: JF prevent multi-hop communications, thus liberating significant capacity, which is used by one-hop flows. Thus, Figure 13 shows how misleading capacity can be to express the impact of DoS: communication still continues to take place, but only with one hop neighbors.

We also observe that even under high loads, the behavior is non-monotonic. The reason is that the existence of surviving flows depends on the topology and node movements: if JF happen to stop a flow that potentially interferes with others, the overall throughput will increase. Otherwise, system throughput is reduced by the thwarted flow's throughput. This dependency on the topology and movement makes the confidence intervals[1] very large, in spite of averaging over 18 different mobility scenarios of 50 runs each.

Thus, with the given topology dimensions of 2000 m $\times$ 2000 m and a high offered load, having 200 nodes with a receive range of 250 m each and a 500 m interference range, the first JF added will most likely *reduce* contention and interference, thus *increasing* system throughput. But beyond a certain number of JF, no flows can take advantage of this removal of interfering flows anymore, and the system throughput starts decreasing.

Figure 14 illustrates this issue from an alternate perspective and depicts the average number of hops for a received packet under different loads. The figure shows that the presence of JF severely diminishes the allowed path lengths for successful communication in both the baseline and the overload cases.

[1]Not shown, for clarity.

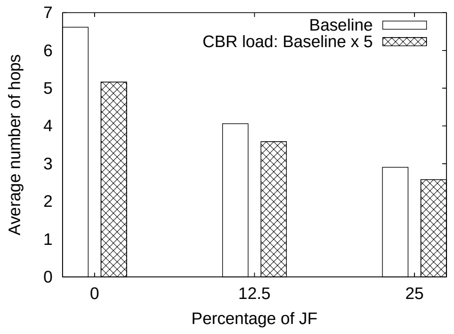

Fig. 14. Hops for received packets with different CBR loads

D. JellyFish Placement

The baseline scenario considers grid placement with the JF placed at equal distances from each other. Here we analyze the effect of different JF placement methods on the effectiveness of the JF attack and consider two additional methods: (i) random static placement in which JF are uniformly randomly placed within the geographical area, yet are non-mobile, and (ii) mobile JellyFish in which JF nodes have the same mobility characteristics as all other nodes.

Figure 15 shows the probability that an established route contains a JF node for the different placement techniques. From Section III, we have that the probability of interception is given by $P_{int} = 1 - (1 - a/N)^h$ for a fixed average number of relay nodes $h = 5.62$. As also described in Section III, this expression is easily generalized to incorporate the hop count *distribution* via

$$P_{int} = \sum_{h \geq 0} (1 - (1 - a/N)^h) Pr(H = h) \qquad (8)$$

where $Pr(H = h)$ is the probability of having $H = h$ relay nodes. The figure indicates that the simplified model which considers path length to be constant overestimates the number of intercepted flows. In contrast, by using

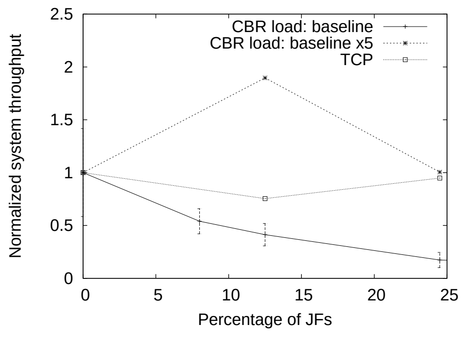

Fig. 13. Normalized system throughput with different loads

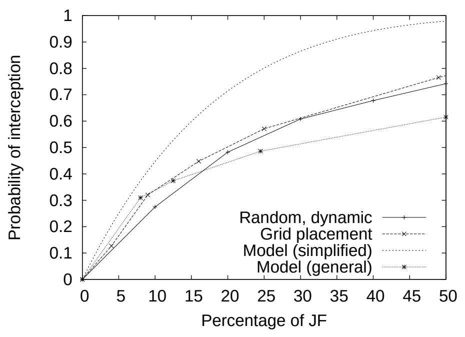

Fig. 15. Probability of interception P_{int} for different JF placement methods

the path length distribution as obtained from simulations together with Equation (8), the curve labeled “Model (general)” provides a close match with simulation.

E. System Size

Finally, we explore the effect of system size on attack scalability. In particular, as demonstrated in Section 3, the mean hop length plays a critical role in an attacks effectiveness. Here, we consider a $1000\,\text{m} \times 1000\,\text{m}$ system vs. the $2000\,\text{m} \times 2000\,\text{m}$ case of the baseline, and keep the node density constant resulting in 50 nodes.

Observe first from Figure 16 that without an attack, the mean hop length for a received packet is reduced by a factor of approximately 2 . Moreover, this factor is maintained across different percentages of JF as shown. Thus, the attack scalability remains unchanged with system size, yet the mean path length has a significant effect.

A similar trend is illustrated in Figure 17 which shows that a smaller system size results in higher initial fairness. That is, with shorter path lengths, flow throughputs are nearly identical. (Consider a small system in which all flows are within radio range: if the MAC protocol provides long term fairness, then the fairness index will be 1.) Yet, both system sizes obtain a similar scaling of a reduction in fairness with an increasing number of attackers.

V. Related Work

Significant recent research efforts have focused on the challenge of securing mobile ad hoc networks with most work targeted towards securing routing protocols. Results can be classified according to the routing protocol(s) they consider and by the assumptions they make in terms of available security mechanisms (e.g., online/offline presence of an identity and key certification center, key distribution and revocation techniques, and cryptographic computation capabilities of the nodes).

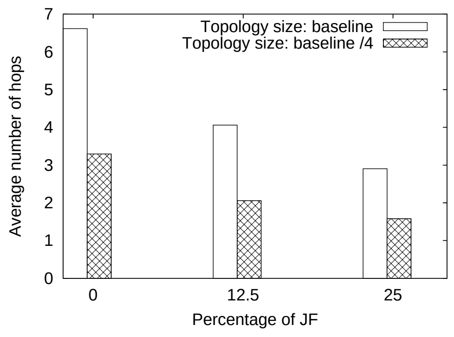

Fig. 16. Hops for received packets for different system sizes

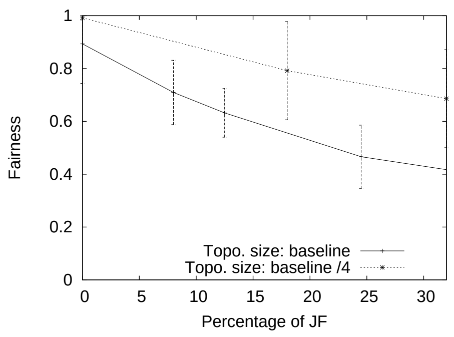

Fig. 17. Fairness for different system sizes

Other relevant parameters include the number of nodes, the mobility model, the underlying transmission protocols (MAC and physical layer), the propagation model of the radio channel, as well as the strength of the attacker (e.g., the number of controlled nodes).

In this section, we present an overview of related work in security in ad hoc networks with an emphasis on the mechanisms aiming at protecting against DoS attacks.

Securing Routing Protocols: The area which has attracted the most attention is security of the routing protocol and in particular, security of route establishment.

Ariadne [4], proposed by Hu, Perrig and Johnson, protects source routing protocols such as DSR against a number of attacks. They propose a protocol to secure the routing discovery phase and to ensure that all forwarded packets follow the secure route. As they mentioned, this protocol does not protect the network against a legitimate but malicious relay node, which silently discards all or part of the packets. The two suggested counter-measures in [4] are Passive Acknowledgement (that we discussed in [21]) and multi-path routing. Reference [4] also suggests blacklisting poorly performing nodes in order to prevent them from being included in future routes; we evaluated the resilience of multi-path routing and reputation systems in Section III.

In the same paper Hu *et al.* also consider a route-request-flooding attack, which without counter-measures can be quite devastating, as each Route Request message generates a broadcast throughout the entire network. The proposed solution consists of having every node *rate limit* the Route Requests it is asked to relay. Although such a mechanism is indeed needed to protect the system from such attacks, we showed in Section III that such rate limiting can also delay a victim’s ability to respond to an attack, and consequently will reduce the throughput of victims.

Hu *et al.* also address the problem of securing distance vector protocols and have developed a protocol termed SEAD (Secure Efficient Ad hoc Distance vector routing protocol) [3]. In order to guard against several attacks including DoS attacks, SEAD makes use of one-way-hash chains and Merkle hash trees. The purpose of these structures is to authenticate the metric (distance to the target) and the sequence numbers (which are used in distance vector to assess the freshness of the information about a given route and, if not properly protected, could be exploited to mount attacks). They conclude that distance vector protocols are more difficult to secure than those based on source routing. In any case, we note that SEAD does not consider attacks against packet forwarding, nor does it address the use of multiple routes.

Other studied attacks include the Rushing Attack [7] (discussed in Section III) and the Wormhole Attack [16]. Reference [31] provides a description of four new mechanisms as tools for securing distance vector and path vector routing protocols; however, these mechanisms aim at protecting against attacks that are different from the those considered in this paper.

Finally, other proposals about secure routing protocols focus on secure route establishment and explicitly exclude packet dropping from their field of investigation [2], [5]; we do not comment on them, as they are quite remote from our topic.

Identification of the Attacking Node(s): We have considered that once a victim has detected a DoS attack, it will establish a new route. A more sophisticated reaction would also attempt to identify the attacking node(s) on the route exhibiting the anomalous behavior. For this purpose, Awerbuch *et al.* propose a technique aiming at identifying a "Byzantine node" on a given route [1]. The technique requires that the destination acknowledge every packet to the source; when the source detects that the number of lost packets is higher than a given threshold, it performs a binary search on the path in order to identify the faulty link. For that purpose, it polls specific nodes via *probes* and asks them to reply. The protocol considers that malicious nodes are unable to distinguish between polling packets and normal ones, and are unable to know whether the source has started a probing session. Although a promising technique, this proposal has been investigated in static scenarios and its effectiveness with mobility is still unproven.

VI. Conclusion

In this paper, we studied a novel DoS attack perpetrated by JellyFish: relay nodes that stealthily misorder, delay, or periodically drop packets that they are expected to forward, in a way that leads astray end-to-end congestion control protocols. This attack is protocol-compliant and yet has a devastating impact on the throughput of closed-loop flows, such as TCP flows and congestion-controlled UDP flows. For completeness, we have also considered a well-known attack, the Black Hole attack, as its impact on open-loop flows is similar to the effect of JellyFish on closed-loop flows.

We studied these attacks in a variety of settings and have provided a quantification of the damage they can inflict. We showed that, perhaps surprisingly, such attacks can actually *increase* the capacity of ad hoc networks as they will starve all multihop flows and provide all resources to one-hop flows that cannot be intercepted by JellyFish or Black Holes. As such a partitioned system is clearly undesirable, we also considered fairness measures and the mean number of hops for a received packet, as critical performance measures for a system under attack.

We assessed the effects of various performance factors on the above metrics via a simple analytical model and a substantial number of simulation experiments. In this way, we provide a quantitative study of the performance impact and scalability of DoS attacks in ad hoc networks.

Our objective is to provide guidelines for protocol designers who are developing DoS-resilience mechanisms: with a better understanding of the key attack factors and how to evaluate the impact of an attack, protocol designers can better determine if the overhead of deploying a counter-strategy is merited given the damage that an attack can inflict.

References

[1] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *Proceedings of WiSe*, 2002.

[2] B. Dahill, K. Sanzgiri, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *Proceedings of ICNP*, 2002.

[3] Y.-C. Hu, D. B. Johnson, and A. Perrig, "Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 175–192, 2003.

[4] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings MobiCom 2002*, September 2002.

[5] M. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2002.

[6] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Mobile Computing and Networking*, 2000, pp. 255–265, http://citeseer.nj.nec.com/marti00mitigating.html.

[7] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proceedings of WiSe*, 2003.

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM*, 2000.

[9] E. Blanton and M. Allman, "On making TCP more robust to packet reordering," *ACM Computer Communications Review*, vol. 32, no. 1, pp. 20–30, 2003.

[10] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka, “TCP-PR: TCP for persistent packet reordering,” in *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems*, 2003.

[11] F. Wang and Y. Zhang, “Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response,” in *Proceedings of MobiHoc*, 2002.

[12] M. Zhang, B. Karp, S. Floyd, and L. Peterson, “RR-TCP: A reordering robust TCP with DSACK,” in *Proceedings of IEEE ICNP*, 2003.

[13] A. Kuzmanovic and E. Knightly, “Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants,” in *Proceedings of ACM SIGCOMM*, 2003.

[14] L. Brakmo, S. O’Malley, and L. Peterson, “TCP Vegas: New techniques for congestion detection and avoidance,” in *Proceedings of ACM SIGCOMM*, 1994.

[15] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, and R. Wang, “TCP Westwood: Bandwidth estimation for enhanced transport over wireless links,” in *Proceedings of ACM MobiCom*, 2001.

[16] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Packet leashes: A defense against wormhole attacks in wireless networks,” in *Proceedings of IEEE Infocom*, 2003.

[17] K. Fall and S. Floyd, “Simulation-based comparison of Tahoe, Reno and SACK TCP,” *ACM Computer Communications Review*, vol. 5, no. 3, pp. 5–21, July 1996.

[18] V. Paxson and M. Allman, “Computing TCP’s retransmission timer,” Nov. 2000, internet RFC 2988.

[19] M. Chan and R. Ramjee, “TCP/IP performance over 3G wireless links with rate and delay variation,” in *Proceedings of ACM MobiCom*, October 2002.

[20] D. B. Johnson and D. Maltz, “The dynamic source routing protocol for mobile ad hoc networks (DSR),” 2003, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt.

[21] I. Aad, J.-P. Hubaux, and E. W. Knightly, “Denial of service resilience in ad hoc networks,” in *Proceedings of Mobicom*, 2004.

[22] P. Papadimitratos and Z. Haas, “Secure data transmission in mobile ad hoc networks,” in *Proceedings of WiSe*, 2003.

[23] M. Gerla, S. Lee, and G. Pau, “TCP Westwood simulation studies in multiple-path cases,” in *Proceedings of SPECTS*, July 2002.

[24] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy, “PATHS: analysis of path duration Statistics and their impact on reactive MANET routing protocols,” in *Proceedings of Mobihoc*, 2003.

[25] S. Buchegger and J.-Y. L. Boudec, “Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes — Fairness In Dynamic Ad-hoc NeTworks,” in *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

[26] P. Michiardi and R. Molva, “Core: A Collaborative Reputation Mechanism To Enforce Node Cooperation In Mobile AD HOC Networks,” in *Proceedings of The 6th IFIP Communications and Multimedia Security Conference*, 2002.

[27] P. Papadimitratos and Z. Haas, “Secure routing for mobile ad hoc networks,” in *Proceedings of CNDS*, 2002.

[28] R. Jain, *The Art of Computer System Performance Analysis*. John Wiley and Sons, Inc., 1991.

[29] “The network simulator - ns-2,” http://www.isi.edu/nsnam/ns/.

[30] http://icapeople.epfl.ch/aad/publ/dos-ton-2007/.

[31] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Efficient security mechanisms for routing protocols,” in *Network and Distributed System Security Symposium, NDSS*, 2003.

Imad Aad (aad@docomolab-euro.com) is a senior researcher at DoCoMo Euro-Labs, Germany, working within the Future Networking Lab. He got his Electrical and Electronics engineering degree in 1998 from the Lebanese University, Beirut. He got his M.S. degree in 1999 from the University of Nice - Sophia Antipolis, then his Ph.D. degree from Joseph Fourier University, France, in 2003. He prepared his Ph.D. on quality of service in wireless LANs at INRIA, France, within the Planète team. He worked at EPFL within the LCA team as a senior researcher from 2003 to 2005 where he worked on cheating and cheating detection issues, DoS attacks and resilience and on general security aspects in wireless networks. He joined DoCoMo Euro-Labs in December 2005 where he is working on cooperative coding, anonymity, quality of service, beam antennas and overlays in wireless networks. He is chairing WiOpt 2008 workshops and served in the program committee of ESAS, WinSys, CFIP, WCNC and IWCMC. http://imad.aad.name

Jean-Pierre Hubaux (jean-pierre.hubaux@epfl.ch) joined the faculty of EPFL in 1990; he was promoted to full professor in 1996. His research activity is focused on wireless networks, with a special interest in security and cooperation issues. He has been strongly involved in the National Competence Center in Research named “Mobile Information and Communication Systems”(NCCR/MICS, a.k.a. Terminodes project), since its genesis in 1999; In this framework, he has notably defined, in close collaboration with his students, novel schemes for the security and cooperation in multi-hop wireless networks, vehicular networks, and sensor networks; He has recently written, with Levente Buttyan, a graduate textbook entitled “Security and Cooperation in Wireless Networks”. He is an Associate Editor of IEEE Transactions on Mobile Computing and Foundations and Trends in Networking. He served as the general chair for the MobiHoc 2002. He has been serving on the program committees of numerous conferences and workshops, including SIGCOMM, Infocom, Mobicom, Mobihoc, SenSys, WiSe, and VANET. He has held visiting positions at the IBM T.J. Watson Research Center and at the University of California at Berkeley. He was born in Belgium, but spent most of his childhood and youth in Northern Italy. After completing his studies in electrical engineering at Politecnico di Milano, he worked 10 years in France with Alcatel, where he was involved in R&D activities, primarily in the area of switching systems architecture and software. http://people.epfl.ch/jean-pierre.hubaux

Edward W. Knightly (knightly@rice.edu) is a professor of Electrical and Computer Engineering at Rice University. He received the B.S. degree from Auburn University in 1991 and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1992 and 1996 respectively. He is an associate editor of IEEE/ACM Transactions on Networking. He served as technical co-chair of IEEE INFOCOM 2005, general chair of ACM MobiSys 2007, and served on the program committee for numerous networking conferences including ICNP, INFOCOM, MobiCom, and SIGMETRICS. He received the National Science Foundation CAREER Award in 1997 and has been a Sloan Fellow since 2001. His research interests are in the areas of mobile and wireless networks and high-performance and denial-of-service resilient protocol design.