TTLed Random Walks for Collaborative Monitoring in Mobile and Social Networks

YANIV ALTSHULER and SHLOMI DOLEV and YUVAL ELOVICI

Abstract Complex network and complex systems research has been proven to have great implications in practice in many scopes including Social Networks, Biology, Disease Propagation, and Information Security. One can use complex network theory to optimize resource locations and optimize actions. Randomly constructed graphs and probabilistic arguments lead to important conclusions with a possible great social and financial influence. Security in online social networks has recently become a major issue for network designers and operators. Being "open" in their nature and offering users the ability to compose and share information, such networks may involuntarily be used as an infection platform by viruses and other kinds of malicious software. This is specifically true for mobile social networks, that allow their users to download millions of applications created by various individual programmers, some of which may be malicious or flawed. In order to detect that an application is malicious, monitoring its operation in a real environment for a significant period of time is often required. As the computation and power resources of mobile devices are very limited, a single device can monitor only a limited number of potentially malicious applications locally. In this work, we propose an efficient collaborative monitoring scheme that harnesses the collective resources of many mobile devices, generating a "vaccination"–like effect in the network. We suggest a new local information flooding algorithm called *Time-to-Live Probabilistic Prop-*

Yaniv Altshuler
Department of Information Systems Engineering and Deutsche Telekom Laboratories, Ben Gurion University of the Negev, POB 653, Beer Sheva 84105, Israel,
and
MIT Media Laboratory, 77 Mass. Ave., E15 Cambridge, MA 02139 USA e-mail: yanival@mit.edu

Shlomi Dolev
Computer Science Department, Ben Gurion University of the Negev, POB 653, Beer Sheva 84105, Israel, e-mail: dolev@bgu.ac.il

Yuval Elovici
Department of Information Systems Engineering and Deutsche Telekom Laboratories, Ben Gurion University of the Negev, POB 653, Beer Sheva 84105, Israel, e-mail: elovici@bgu.ac.il

agation (TPP). The algorithm is implemented in any mobile device, periodically monitors one or more applications and reports its conclusions to a small number of other mobile devices, who then propagate this information onwards, whereas each message has a predefined "Time-to-Live" (TTL) counter. The algorithm is analyzed, and is shown to outperform the existing state of the art information propagation algorithms, in terms of convergence time as well as network overhead. We then show both analytically and experimentally that implementing the proposed algorithm significantly reduces the number of infected mobile devices. Finally, we analytically prove that the algorithm is tolerant to the presence of adversarial agents that inject false information into the system.

1 Introduction

The market share of Smart-phones is rapidly increasing and is expected to increase even faster with the introduction of 4^{th} generation mobile networks, reaching from 350 million in 2009 to one billion by 2012 [13]. Companies that are distributing new mobile device operating systems had created a variety of marketplaces that motivate individuals and other companies to introduce new applications (such as Apple's *App Store* Google's *Android Market*, Nokia's *Ovi Store* and others). The main assumption behind these marketplaces is that users will prefer a mobile device based on an operating system with larger marketplace offerings. It is expected that in the future, various communities will develop additional alternative marketplaces that will not be regulated. These marketplaces will allow mobile users to download from a variety of millions of new applications. An example for such a marketplace is *GetJar*, offering 60,000 downloadable applications for over 2,000 mobile devices, counting a total of over 900 million downloads by August 2010 [27]. The content of most marketplaces is currently not verified by their operators, and thus some applications may be malicious or contain faulty code segments. Downloading a malicious application from the marketplace is not the only way that a mobile device may be infected by a malicious code. This may also happen as a result of a malicious code that manages to exploit a vulnerability in the operating systems and applications or through one of the mobile phone communication channels such as Bluetooth, Wi-Fi, etc. [19,30,32,37,58]. The number of currently active malware variants is estimated to be above 370 [31], most of which are introduced as a result of an infected application installation. McAfee's Mobile Security Report for 2008 states that nearly 14% of global mobile users were directly infected or have known someone who had been infected by a mobile virus (this number had increased in the followed year) [1,2]. In many cases, in order to detect that an application is malicious, monitoring its operation in a real environment for a significant period of time is required. The data that results from this monitoring is being processed using advanced machine learning algorithms in order to assess the maliciousness of the application. For a variety of local monitoring techniques for mobile phone applications, see [33,36,45–47,49].

In recent years, most of the prominent security and privacy threats for communication networks have relied on the use of collaborative methods (e.g., Botnets). The danger stemming from such threats is expected to significantly increase in the near future, as argued in [37, 65] and others. The amount of resources a single unit may allocate in order to defend from such threats without interfering with its routine work is very limited. Therefore, the development of efficient "collaborative defense infrastructures" is strongly required.

In this work, we propose such a collaborative application monitoring infrastructure, providing high efficiency, scalability and fault tolerance for known adversarial and Byzantine attacks.

The efficiency of the proposed algorithm stems from the generation of an implicit collaboration between a group of random walking agents who are released from different sources in the network (and at different times). This technique is new, as most related works discussing random walkers did not take into consideration that agents may be released collaboratively from different sources (e.g. [20, 21]. Hence, the analysis of such systems was limited to the probabilistic analysis of the movements of the agents.

We present a simple collaborative monitoring algorithm, called *TPP — Time-to-Live Probability Propagation*. The algorithm uses a "Time-to-Live" counter for each message, defining the number of time a message may be forwarded before it is deleted, that is logarithmic in the number of the network's devices, n. A successful completion of the mission using these short living messages is then proved (Theorem 2). In fact, the upper bounds for the algorithm's completion time and overall number of messages are $O(\log n)$ and $O(n \log n)$, respectively (Corollaries 1 and 2). The benefit factor of participating in the proposed collaborative scheme is shown to monotonically increase with the size of the network, n. Specifically, we show that by sending $O(\ln n)$ messages, the number of applications a device has to monitor locally is reduced by a factor of $O(\ln n)$ (see Corollary 2).

In addition, the algorithm is shown to be partially fault tolerant to the presence of Byzantine devices, that are capable of distributing messages concerning benign applications.[1] (see Theorem 5 in Section 6 discussing devices who distribute false messages).

Furthermore, we show that in addition to providing a mechanism for defending against adversarial abuse, the efforts the algorithm requires for overcoming such attacks grow asymptotically slower than the efforts required in order to increase the strength of such attacks, making the algorithm highly scalable (see more details in Section 6 and specifically an illustration that appears in Figure 8).

The rest of this work is organized as follows : A related work section and a comparison to state-of-the-art techniques is presented in Section 2, in which it is shown that in many cases the *TPP* algorithm achieves a completion time which is equal to the completion time of the fastest single-source information propagation algorithm (i.e. flooding), and does so using significantly lower number of messages. The formal definition of the problem appears in Section 3, and the *TPP* collaborative

[1] We assume that interference in the messages' content, or generation of messages using false identity are impossible, due to, say, the use of cryptographic means.

algorithm and its performance analysis are discussed in Section 4.1. Experimental results are presented in Section 5, whereas the algorithm's robustness with regards to attacks by adversaries is discussed in Section 6. Conclusions and suggestions for future research appear in Section 7.

2 Related Work

Flooding a network with messages intended for a large number of nodes is arguably the simplest form of information dissemination in communication networks (specifically when previous knowledge about the network topology is limited or unavailable). Since the problem of finding the minimum energy transmission scheme for broadcasting a set of messages in a given network is known to be NP-Complete [12], flooding optimization often relies on approximation algorithms. For example, in [29, 57] messages are forwarded according to a set of predefined probabilistic rules, whereas [53] discusses a deterministic algorithm, approximating the connected dominating set of each node. Alternative information propagation techniques simulate various epidemic models [25, 55, 56, 63].

In this work, we apply a different approach — instead of a probabilistic forwarding of messages, we assign a *TTL* value for each message, in order to guide the flooding process. The analysis of the algorithm is done by modeling the messages as agents practicing *random walk* in a *random graph* overlay of the network. The optimal value of this TTL is shown, guaranteeing a fast completion of the task, while minimizing the overall number of messages sent. The use of a TTL dependent algorithm was discussed in [3] (demonstrating $O(\log^2 n)$ completion time) and [44] (where no analytic result concerning the completion time was demonstrated).

In addition, the algorithm's efficiency is further improved by selecting the TTL value in a way which focuses the collaborative efforts towards threats of high penetration probabilities. This approach is inspired in part by the observation made in [65] regarding a critical phase transition point of threats' penetration rates, defining the epidemic potential of the threat.

Flooding Algorithms. The simplest information propagation technique is of course the *flooding* algorithm. It is well known that the basic flooding algorithm, assuming a single source of information, guarantees completion in a worse case cost of $O(n^2)$ messages ($O(|E|)$ in expanders, and $O(n \cdot \log(n))$ in random graphs $G(n,p)$), and time equals to the graph's diameter, which in the case of a random graph $G(n,p)$ equals $O(\frac{\log n}{\log(n \cdot p)}) \approx O(\log n)$ [34] [16, 22]. We later show that the *TPP* algorithm discussed in this work achieves in many cases the same completion time, with a lower cost of $O(n \log n)$ messages (Corollaries 1 and 2).

There are various methods used to improve the efficiency of the basic flooding algorithm. One example is a probabilistic forwarding of the messages in order to reduce the overall cost of the propagation process, optionally combined with a mechanism for recognizing and reducing repetitions of packets transitions by the same device [50]. Other methods may include area based methods [50] or neigh-

borhood knowledge methods [42, 51, 59, 60]. In many of these works, completion time is traded for a reduced overall cost, which results in a similar cost as the *TPP* algorithm proposed in this work (namely, $O(n \log n)$), but with a significantly higher completion time. Additional details on various flooding algorithms can be found in [66].

An extremely efficient flooding algorithm, in terms of completion time, is the network coded flooding algorithm, discussed in [17]. In this work, dedicated to $G(n,p)$ random graphs, a message is forwarded by any receiving vertex $\frac{k}{d(v)}$ times, where k is a parameter which depends on the network's topology [24]. Using this method, the algorithm achieves a completion time of approximately $O(\frac{n^3}{|E|^2})$. This algorithm, however, is still outperformed by the *TPP* algorithm. Specifically, the *TPP* algorithm would perform faster in graphs with average degree of less than $O\left(\sqrt{\frac{n}{\ln n}}\right)$.

Epidemic Algorithms. An alternative approach to be mentioned in this scope is the use of *epidemic algorithms* [9, 55, 63]. There exist a variety of epidemic algorithms, starting with the basic epidemic protocol [18, 28], through *neighborhood epidemics* [25] and up to *hierarchical epidemics* [56]. In general, all the various epidemic variants have a trade-off between number of messages sent, completion time, and previous knowledge required for the protocols (concerning the structure of the network). However, the most efficient results of such approaches are still outperformed by the *TPP* algorithm.

Distributed Coverage Algorithms. A different approach for a collaborative assimilation of an important piece of information throughout the network is the use of cooperative exploration algorithms (in either known or unknown environments), guaranteeing that all (or a large enough portion) of the graph is being "explored" by agents carrying the alerting messages. Planar networks can be sampled into $\mathbf{R}^2$, and then be collaboratively explored by a decentralized group of myopic agents (see additional swarm coverage examples in [5, 11, 38, 40, 40, 52, 54, 61, 67]).

In [62] a swarm of ant-like robots is used for repeatedly covering an unknown area, using a real time search method called *node counting*. Using this method, the agents are analytically shown to cover the network efficiently. Another algorithm to be mentioned in this scope is the *LRTA** algorithm [41], that was shown in [39] to guarantee coverage time of $O(n^2)$ in degree bounded undirected planar graphs. Interestingly, in such graphs, the random walk algorithm is also known to require at most $O(n^2)$ time (and at least $\Omega(n(\log n)^2)$) [35].

In [64] a collaborative algorithm that guarantees coverage of regions in the $\mathbf{Z}^2$ grid, in $O(n^{1.5})$ time, using extremely simple and myopic "ant like" mobile agents and using no direct communication by the agents is discussed. In [4] this algorithm was later shown to guarantee (under some limitations) coverage of dynamically expanding domains (namely, domains in which explored vertices may become "unexplored" after a certain time of being exposed to an unexplored neighbor) in $O(n^{2.5} \log n)$ time.

Summary. Tables 1 and 2 compare the performance of the *TPP* algorithm (in terms of convergence time and overall message overhead) to the main leading works in this field. Each table examines a different set of assumptions concerning the network. In each table, the algorithm displaying the best result is marked using the "✓" sign.

As previously discussed, the efficiency of the *TPP* algorithm is derived from the fact that participating devices form a collaborative infrastructure, tuned to focus on threats of high penetration rates. As the *TPP* algorithm uses random elements, it also requires approximately $O(\ln^2 n)$ random bits by each device.

Another interesting approach with respect to decentralized information proliferation is the *population problem*, discussed for example in [6], in which a consensus among a decentralized group of n units is generated in $O(n \log n)$ time, with tolerance to the presence of $O(\sqrt{n})$ Byzantine agents. Additional information on population protocols can be found in [8].

	Time	**Messages**
TPP	In most cases $O(\ln n)$ and at most $O(\frac{n}{\ln n})$	$O(n \ln n)$
Flooding	O (*Graph's diameter*)	$O(\|E\|)$
Network Coded Flooding [17] using $G(n,p)$ overlay	$O\left(n^{-1} \cdot p^{-2}\right)$	$O(n)$
Neighborhood Epidemics [25] using $G(n,p)$ overlay	$O(n^c)$ for some constant c	$O(c \cdot n)$ for some constant c
Hierarchical Epidemics [56] using α-tree overlay	$O(\ln n)$	$O(\alpha \cdot n \ln n)$ for branching factor α
LRTA* [41] in planar degree bounded graphs	$O(n^2)$	$O(n^2)$
SWEEP [64] in the $\mathbf{Z}^2$ grid	$O(n^{1.5})$	$O(n^{1.5})$

Table 1 Performance comparison of *TPP* and available state of the art algorithms.

	Time	**Messages**
TPP	In most cases $O(\ln n)$ ✓ and at most $O(\frac{n}{\ln n})$	$O(n \ln n)$
Flooding	$O(\ln n)$ ✓	$O(n^2 p)$
Network Coded Flooding	$O\left(n^{-1} \cdot p^{-2}\right)$	$O(n)$ ✓
Neighborhood Epidemics	$O(n^c)$ for some constant c	$O(c \cdot n)$ ✓ for some constant c
Hierarchical Epidemics using α-tree overlay	$O(\ln n)$ ✓	$O(\alpha \cdot n \ln n)$ for branching factor α

Table 2 Performance comparison for random $G(n,p)$ graphs, with $p < O((n \ln n)^{-0.5})$. The "✓" sign marks the algorithm with the best performance.

3 The Collaborative Application Monitoring Problem

Given a mobile network of n devices, let us denote the network's devices by $V = \{v_1, v_2, \ldots, v_n\}$. Note that the network's topology may be dynamic[2]. Each device may occasionally visit the marketplace, having access to N new downloadable applications every month. We assume that downloading of applications is done independently, namely — that the probability that a user downloads application a_1 and the probability that the same user downloads application a_2 are uncorrelated.

For some malicious application a_i, let p_{a_i} denote the application's *penetration probability* — the probability that given some arbitrary device v, it is unaware of the maliciousness of a_i. The penetration probability of every new malicious application is 1. Our goal is to verify that at the end of the month, the penetration probability of all malicious applications released during this month are lower than a *penetration threshold* p_{MAX}, resulting in a "vaccination" of the network with regards to these applications. Formally, for some small ε we require that :

$$\forall \textit{Malicious application } a_i \quad Prob\,(p_{a_i} > p_{MAX}) < \varepsilon$$

The reason behind the use of the threshold p_{MAX} is increasing the efficiency of the collaborative system, defending against dominant threats. Given additional available resources, the parameter p_{MAX} can be decreased, resulting in a tighter defense grid (traded for increased convergence time and messages overhead).

We assume that any device v can send a message of some short content to any other device u. In addition, we assume that at the initialization phase of the algorithm each device is given a list containing the addresses of some X random network members. This can be implemented either by the network operator, or by distributively constructing and maintaining a random network overlay.

We assume that each device can locally monitor applications that are installed on it (as discussed for example in [7, 46]). However, this process is assumed to be expensive (in terms of the device's battery), and should therefore be executed as few times as possible. The result of an application monitoring process is a non-deterministic boolean value : $\{true, false\}$.

False-positive and false-negative error rates are denoted as :

$$\begin{aligned} P(\textit{Monitoring}(a_i) = \textit{ true } \mid A_i \textit{ is not malicious}) &= E_+ \\ P(\textit{Monitoring}(a_i) = \textit{false} \mid \quad A_i \textit{ is malicious}) &= E_- \end{aligned}$$

We assume that the monitoring algorithm is calibrated in such way that $E_+ \approx 0$.

As we rely on the propagation of information concerning the maliciousness of applications, our system might be abused by injection of inaccurate data. This may be the result of a deliberate attack, aimed for "framing" a benign application (either as a direct attack against a competitive application, or as a more general attempt for undermining the system's reliability), or simply as a consequence of a false-positive

[2] This will later come into effect when messages will be sent between the network's members, at which case the selection of "an arbitrary network member" can be assumed to be purely random.

result of the monitoring function. Therefore, in order for a device v to classify an application a_i as malicious, one of the following must hold :

- Device v had monitored a_i and found it to be malicious.
- Device v had received at least ρ alerts concerning a_i from different sources (for some *decision threshold* ρ).

In addition, note that the information passed between the devices concerns only applications discovered to be malicious. Namely, when an application is found to be benign, no message concerning this is generated. This is important not only for preserving a low message overhead of the algorithm but also to prevent malicious applications from displaying a normative behavior for a given period of time, after which they initiate their malicious properties. In such cases, soon after an application exits its "dormant" stage, it will be detected and subsequently reported, generating a "vaccination reaction" throughout the network.

4 Algorithm, Correctness & Analysis

We shall now present the *TPP* algorithm. The algorithm is executed by each device separately and asynchronously, where no supervised or hierarchical allocation of tasks, as well as no shared memory are required. Table 4 presents the main notations used in the presentation and analysis of the proposed algorithm.

n	The number of devices participating in the *TPP* algorithm
X	The number of alert messages generated and sent upon the discovery of a malicious application
p_N	The ratio $\frac{X}{n}$
p_{MAX}	Penetration threshold — the maximal percentage of non-vaccinated devices allowed by the network operator
E_-	False negative error rate of the local monitoring mechanism
T	Delay time between each two consecutive local monitoring processes
N	Number of new applications introduced to the network each month
p_{a_i}	The penetration probability of application a_i
$1-\varepsilon$	Confidence level of the correctness of the convergence time estimation
α	The polynomial confidence level $\ln_n \varepsilon^{-1}$
ζ_{T,N,P_M,E_-}	The *vaccination factor*, defined as $\frac{T \cdot N}{p_{MAX}(1-E_-)}$
timeout	The Time-To-Live counter assigned to alert messages
ρ	Decision threshold — the number of alerts a device must receive in order to classify an application as malicious
C_S	Cost of sending a single message
C_M	Cost of locally monitoring a single application

Table 3 Main notations used throughout this work

Overview of the Results. The *TPP* algorithm is analytically shown to guarantee a successful completion of the monitoring mission (Theorem 2). The performance

of the algorithm is then analyzed, and is shown to be superior compared to existing results in this domain (in terms of completion time and messages overhead). Upper bounds for the algorithm's completion time for the overall number of messages required are presented in Observation 1. Approximation of these bounds are given in Theorems 3 and 4. More explicit approximations of the bounds for *sparsely connected networks* (see Definition 1) are presented in Corollaries 1 and 2.

4.1 TTL Probabilistic Propagation (TPP) — *a Collaborative Monitoring Algorithm*

The *TPP* algorithm conceptually relies on the fact that in order to "vaccinate" a network with regards to malicious applications, it is enough that only a small number of devices will monitor this application. This way, although the application monitoring process is relatively expensive (in terms of battery and CPU resources), the amortized cost of monitoring each malicious application is kept to a minimum. A detailed implementation of the *TPP* algorithm appears in Algorithm 1.

At its initialization (lines 1 through 6), all the applications installed on the device are added to a list of *suspected applications*. In addition, an empty list of *known malicious applications* is created. Once an application is determined as *malicious*, it is added to the known malicious application list. In case this application was also in the suspected application list (namely, it is installed on the device, but has not been monitored yet), it is deleted from that list. Once a new application is encountered it is compared to the known malicious application list, and if found, an alert is sent to the user (alternatively, the application can be chosen to be uninstalled automatically). This feature resembles the long-term memory of the immune system in living organisms. If the new application is not yet known to be malicious, the application is added to the suspected application list.

Once executed, a periodic selection of an arbitrary application from the list of suspected applications is done, once every T steps (lines 23 through 31). The selected application is then monitored for a given period of time, in order to discover whether it is of malicious properties (see details about such monitoring in Section 3). If the application is found to be malicious, it is removed from the list of suspected applications and added to the known malicious application list (lines 28 and 27). In addition, an appropriate alert is produced and later sent to X random devices. The alert message is also assigned a specific TTL value (lines 29 and 30). Once a network device receives such an alert message it automatically forwards it to one arbitrary device, while decreasing the value of TTL by 1. Once TTL reaches zero, the forwarding process of this message stops (lines 21 and 22). In case a monitored application displays no malicious properties, it is still kept in the list of suspicious applications, for future arbitrary inspections.

A device may also classify an application as malicious as a result of receiving an alert message concerning this application (lines 14 through 20). In order to protect benign applications from being "framed" (reported as being malicious by adver-

saries abusing the vaccination system), a device classifies an application as malicious only after it receives at least ρ messages concerning it, from different sources (for a pre-defined *decision threshold* ρ). Note that when a device v receives an alert message concerning application a_i, it still forwards this message (assuming that $TTL > 0$), even when v has not yet classified a_i as malicious (for example, when the number of alert messages received is still smaller than ρ). When the ρ-th alerting message concerning an application is received, the device adds the application to its known malicious application list and removes it from the suspected application list if needed. The selection of the optimal value of ρ is discussed in Section 6. The values of T, ρ and TTL, as well as the number of generated alert messages can be determined by the network operators, or be changed from time to time according to the (known or estimated) values of n and N, and the penetration threshold p_{MAX}. Selecting an optimal value for TTL is discussed in Section 4.

Algorithm 1 *TTL Probabilistic Propagation*

Initialization :
  Let $A(v)$ be the list of installed applications
  Let $\dot{A}(v)$ be the list of suspected applications
  Let $\widetilde{A}(v)$ be the list containing known malicious applications
  Initialize $\dot{A}(v) \leftarrow A(v)$
  Initialize $\widetilde{A}(v) \leftarrow \emptyset$

Interrupt upon encountering a new application a_i :
  $\dot{A}(v) \leftarrow \dot{A}(v) \cup \{a_i\}$
  If $a_i \in \widetilde{A}(v)$ then
    $\dot{A}(v) \leftarrow \dot{A}(v) \setminus \{a_i\}$
    Issue an alert to the user concerning a_i
  End if

Interrupt receives malicious application a_i notice, for the j-th time :
  If $j \geq \rho$ then
    $\dot{A}(v) \leftarrow \dot{A}(v) \setminus \{a_i\}$
    $\widetilde{A}(v) \leftarrow \widetilde{A}(v) \cup \{a_i\}$
    If $a_i \in A(v)$ then
      Issue an alert to the user concerning a_i
    End if
  End if
  Decrease TTL of report by 1
  Forward report to a random network member

Execute every T **time-steps** :
  Select a random application a_i from $\dot{A}(v)$ for monitoring
  If a_i is found to be malicious then
    Issue an alert to the user concerning a_i
    $\widetilde{A}(v) \leftarrow \widetilde{A}(v) \cup \{a_i\}$
    $\dot{A}(v) \leftarrow \dot{A}(v) \setminus \{a_i\}$
    Report a_i to X random network members
    Set $TTL = timeout$
  End if

4.2 Optimal Parameters for Guaranteeing Successful Monitoring

Outline of Analysis. In order to analyze the algorithm's behavior, we shall model the movements of the notification messages between the network's devices as random walking agents, traveling in a random graph $G(n,p)$. Taking into account the fact that the messages have limited lifespan (namely, TTL), a relation between the size of the graph and the lifespan of the agents is produced. Having the value of TTL that guarantees a coverage of the graph, the algorithm's completion time, as well as the overall number of messages sent, can then be calculated.

While analyzing the performance of the *TPP* algorithm we imagine a directed *Erdös-Renyi* random graph $G(V,E) \sim G(n,p_N)$, where $p_N = \frac{X}{n}$. The graph's vertices V denote the network's devices, and the graph's edges E represent message forwarding connections. Notice that as G is a random graph, it can be used for the analysis of the performance of the *TPP* algorithm, although the message forwarding connections of the *TPP* algorithm are dynamic. In addition, although the identity of the "neighbors" of a vertex v in the real network overlay may change from time to time (as the overlay graph can be dynamic), it can still be modeled by static selection of X random neighbors of v.

Observing some malicious application a_i, every once in a while some device which a_i is installed on randomly selects it for monitoring. With probability $(1-E_-)$ the device discovers that a_i is malicious and issues alerts to X network's members. We look at these reports as the system's "agents", and are interested in finding :

- The time it takes the graph to be explored by the agents. Namely, the time after which every device was visited by at least ρ agents (and is now immune to a_i).
- The total number of messages sent during this process.
- The minimal TTL which guarantees a successful vaccination of the network.

Note also that agents have a limited lifespan, equals to $timeout$. As the graph is a random graph, the location of the devices in which a_i is installed is also random. Therefore, as they are the sources of the agents, we can assume that the initial locations of the agents are uniformly and randomly distributed along the vertices of V. In compliance with the instruction of the *TPP* algorithm, the movement of the agents is done according to the random walk algorithm.

Application a_i is installed on $n \cdot p_{a_i}$ devices, each of which monitors a new application every T time steps. Upon selecting a new application to monitor, the probability that such a device will select a_i is $\frac{1}{N}$. The probability that upon monitoring a_i the device will find it to be malicious is $(1-E_-)$, in which case it will generate $n \cdot p_N$ alerting messages. The expected number of new agents created at time t, denoted as $\hat{k}(t)$, therefore equals :

$$\hat{k}(t) = \frac{n^2 \cdot p_{a_i} \cdot p_N}{T \cdot N}(1-E_-)$$

and the accumulated number of agents which have been generated in a period of t time-steps is therefore $k_t = \sum_{i \le t} \hat{k}(i)$.

The value of $timeout$ (the assigned TTL) is selected in such a way that the complete coverage of the graph, and therefore its vaccination against a_i, is guaranteed (in probability greater than $1-\varepsilon$). We now artificially divide the mission to two phases, the first containing the generation of agents and the second discussing the coverage of the graph. Note that this division ignores the activity of the agents created in the second phase. Note also that the fact that the agents are working in different times (and in fact, some agents may already vanish while others have not even been generated yet) is immaterial. The purpose of this technique is to ease the analysis of the flow of the vaccination process.

Denoting the completion time by T_{Vac} we therefore have :

$$T_{Vac} \leq T_{Generation} + T_{Propagation}$$

It is easy to see that $T_{Propagation} \triangleq timeout$. We now artificially set :

$$\begin{cases} timeout &= \lambda \cdot (T_{Generation} + timeout) \\ T_{Generation} &= (1-\lambda) \cdot (T_{Generation} + timeout) \end{cases}$$

From this we can see that :

$$T_{Generation} = \frac{(1-\lambda)}{\lambda} \cdot timeout$$

We later demonstrate an upper bound for $timeout$, based on the number of agents created in $t \leq T_{Generation}$ (ignoring the activity of the agents created between $t = T_{Generation}$ and $t = T_{Generation} + timeout$).

We now examine the case of $\lambda = 0.5$ (which we show to be the optimal selection for λ, in the paper's Appendix). In this case, we can now write: $T_{Vac} \leq 2 \cdot timeout$.

Let us denote the number of agents created in the first $T_{Generation}$ time-steps by $k = k_{T_{Generation}}$. We now find the time it takes those k agents to completely cover the graph G, and from this, derive the value of $timeout$.

Recalling that upon the detection of a malicious application, devices are required to send notification messages to exactly X random network neighbors. We can still use a random graph $G(n,p)$ for analyzing the algorithm's behavior. Since our bounds are probabilistic, we can state that the following "bad event" occurs with very low probability (e.g. $2^{-\omega(n)}$). Event $E_{low\ degree}$, defined as the existence of some vertex $v \in V$ with $deg(v) < \frac{n \cdot p_N}{2}$. Using the *Chernoff* bound on G we get : $Prob[deg(v) < \frac{n \cdot p_N}{2}] < e^{-\frac{n \cdot p_N}{8}}$. Applying union bound on all vertices we get :

$$Prob[E_{low\ degree}] < n \cdot e^{-\frac{n \cdot p_N}{8}} < 2^{-\omega(n)}$$

Similarly :

$$Prob[E_{high\ degree}] < 2^{-\omega(n)}$$

From now on we assume that $E_{low\ degree}$ and $E_{high\ degree}$ do not occur, and condition all probabilities over this assumption. In the private case of $\forall v \in V$, $deg(v) = p_N \cdot n$, every analysis that is based on the expected number of neighbors shall hold.

In order to continue analyzing the execution process of the *TPP* algorithm we note that as the initial placement of the agents is random, their movement is random and the graph G is random, we can see that the placement of the agents after every-step is purely random over the nodes. Using these observation, the number of agents residing in adjacent vertices from some vertex v can be produce :

Lemma 1. *Let $v \in V$ be an arbitrary vertex of G. Let $N_1(v,t)$ be the number of agents which reside on one of $Neighbor(v)$ (adjacent vertices to v) after step t :*

$$\forall t \geq 0 : E[N_1(v,t)] \geq \frac{p_N \cdot k}{2}$$

In other words, the expected number of agents who reside in distance 1 *from v after every step is at least $\frac{p_N \cdot k}{2}$.*

Proof. Upon our assumption, in $G(n, p_N)$ the number of incoming neighbors for some vertex v is at least $\frac{1}{2} p_N \cdot n$. In addition, for every $u \in V(G)$, Prob[some agent resides on u] $= \frac{k}{n}$. In addition, for every $u \in V$ such that $(u,v) \in E$ we also know that $deg(u) \leq \frac{3}{2} p_N \cdot n$. Combining the above together, we get $\forall t \geq 0 : E[N_1(v,t)] \geq \frac{p_N \cdot n \cdot k}{2n} \geq \frac{1}{2} p_N \cdot k$.

Lemma 2. *For any vertex $v \in V$, the probability of v being notified at the next time-step that a_i is malicious is at least $1 - e^{-\frac{k}{2n}}$.*

Proof. The probability that an agent located on a vertex u such that $(u,v) \in E$ will move to v at the next time-step is $\frac{1}{p_N \cdot n}$. The number of agents that are located in adjacent vertices to v is $\frac{k}{2} p_N$. Therefore, the probability that v will not be reported about a_i at the next time-step is $(1 - \frac{1}{p_N \cdot n})^{\frac{1}{2} p_N \cdot k}$. Using the well known inequality $(1-x) < e^{-x}$ for $x < 1$, we can bound this probability from above by :

$$(e^{-\frac{1}{p_N \cdot n}})^{\frac{1}{2} p_N \cdot k} \leq e^{-\frac{k}{2n}}$$

Therefore, the probability that v will be notified on the next time-step is at least $1 - e^{-\frac{k}{2n}}$.

Interestingly, this fact holds for any positive p_N (the density parameter of G).

Let us denote by ρ-*coverage* of a graph the process the result of which is that every vertex in the graph was visited by some agent at least ρ times.

Theorem 1. *The time it takes k random walkers to complete a ρ-coverage of G in probability greater than $1 - \varepsilon$ (denoted as $T(n)$) can be bounded as follows :*

$$T(n) \leq \frac{2\left(\rho - \ln \frac{\varepsilon}{n}\right)}{1 - e^{-\frac{k}{2n}}}$$

Proof. Lemma 2 states the probability that some vertex $v \in V$ will be reported of a_i at the next time-step. This is in fact a Bernoulli trial with :

$$p_{success} = 1 - e^{-\frac{k}{2n}}$$

Now we bound the probability of failing this trial (not notifying vertex v enough times) after m steps. Let $X_v(m)$ denote the number of times that a notification message had arrived to v after m steps, and let $F_v(m)$ denote the event that v was not notified enough times after m steps (i.e. $X_v(m) < \rho$). We additionally denote by $F(m)$ the event that one of the vertices of G is not notified enough times after m steps (i.e. $\bigcup_{v \in V(G)} F_v(m)$). We use the Chernoff *bound :*

$$P[X_v(m) < (1-\delta)p_{success}m] < e^{-\delta^2 \frac{mp_{success}}{2}}$$

in which we set $\delta = 1 - \frac{\rho}{mp_{success}}$. We can then see that :

$$P[X_v(m) < \rho] < e^{-(1-\frac{\rho}{mp_{success}})^2 \frac{mp_{success}}{2}}$$

namely : $P[F_v(m)] < e^{\rho - \frac{mp_{success}}{2}}$. Applying the union bound we get :

$$P[e_1 \cup e_2 \cup \ldots \cup e_n] \leq P[e_1] + P[e_2] + \ldots + P[e_n]$$

on all n vertices of G. Therefore we can bound the probability of failure on any vertex v (using Lemma 2) as follows :

$$Pr[F(m)] \leq ne^{\rho - \frac{mp_{success}}{2}} \leq ne^{\rho - \frac{m(1-e^{-\frac{k}{2n}})}{2}} \leq \varepsilon$$

and the rest is implied.

We now show how to select a value of *timeout* that guarantees a successful vaccination process :

Theorem 2. *For every values of timeout that satisfies the following expression, the* TPP *algorithm is guaranteed to achieve successful vaccination for any penetration threshold p_{MAX}, in probability greater than $1-\varepsilon$:*

$$\frac{2\left(\rho - \ln\frac{\varepsilon}{n}\right)}{timeout\left(1 - e^{-timeout \cdot \frac{n \cdot p_{MAX} \cdot p_N}{2T \cdot N}(1-E_-)}\right)} = 1$$

Proof. Recalling the expected number of agents generated at each time step, the expected number of agents k that appears in Theorem 1 equals :

$$E[k] = \sum_{i \leq T_{Generation}} \frac{n^2 \cdot p_{a_i} \cdot p_N}{T \cdot N}(1 - E_-)$$

A successful termination of TPP *means that the penetration probability of (any) malicious application is decreased below the threshold p_{MAX}. Until this is achieved, we can therefore assume that this probability never decreases below p_{MAX} :*

$$\forall t < T_{Generation} : p_{a_i} \geq p_{MAX}$$

Therefore, we can lower bound the number of agents as follows :

$$k \geq timeout \cdot \frac{n^2 \cdot p_{MAX} \cdot p_N}{T \cdot N}(1 - E_-)$$

Assigning timeout = m into Theorem 1, successful vaccination is guaranteed for :

$$timeout = \frac{2(\rho - \ln \frac{\varepsilon}{n})}{1 - e^{-\frac{k}{2n}}} \leq \frac{2\left(\rho - \ln \frac{\varepsilon}{n}\right)}{1 - e^{-\frac{n \cdot p_{MAX} \cdot p_N}{2T \cdot N \cdot timeout^{-1}}(1 - E_-)}}$$

and the rest is implied

4.3 Number of Messages and Time Required for Vaccination

From the value of *timeout* stated in Theorem 2, the vaccination time T_{Vac} as well as the overall cost of the *TPP* algorithm can now be extracted. The cost of the algorithm is measured as a combination of the overall number of messages sent during its execution and the total number of monitoring activities performed. Let us denote the cost of sending a single message by C_S and the cost of executing a single local monitoring process by C_M.

Observation 1 *For any timeout* $= \tau$ *which satisfies Theorem 2, the time and cost of the* TPP *algorithm can be expressed as :*

$$T_{Vac} = O(\tau) \quad , \quad M = O(k \cdot \tau \cdot C_S + \frac{k}{X} C_M) =$$

$$O\left(\frac{p_{MAX} \cdot p_N}{n^{-2} T \cdot N} \cdot (1 - E_-) \cdot \left(\tau^2 \cdot C_S + \frac{\tau}{n \cdot p_N} \cdot C_M\right)\right)$$

Let us assume that ε is polynomial in $\frac{1}{n}$, namely : $\varepsilon = n^{-\alpha} \quad s.t. \quad \alpha \in \mathbb{Z}^+$.
Using the bound $(1 - x) < e^{-x}$ for $x < 1$ we can see that when assuming [3] that :

$$timeout \cdot \frac{n \cdot p_{MAX} \cdot p_N}{2T \cdot N}(1 - E_-) < 1$$

[3] The intuition behind this assumption is as follows : we aspire that the number of messages each device is asked to send upon discovering a new malicious application is kept to a minimum. As the value of P_N is required to be greater than $\frac{\ln n}{n}$ in order to guarantee connectivity [23], it is safe to assume that $P_N = O\left(\frac{\ln n}{n}\right)$. Notice that under some assumptions, a connected pseudo-random graph can still be generated, such that $p_N = O(\frac{1}{n})$ (see for example [21]). However, as we are interested in demonstrating the result for any random graph $G(n, p)$, this lower bound of p_N is still mentioned. In addition, we later show that $timeout \approx O(\log n)$. It is also safe to assume that $N \approx \Omega(\ln n)$ and that $P_{MAX} \approx O\left(\frac{1}{\ln n}\right)$. This assumption is later discussed in great details.

Theorem 2 can be written as :

$$\rho + (\alpha+1)\ln n \geq timeout^2 \cdot \frac{n \cdot p_{MAX} \cdot p_N \cdot (1-E_-)}{4T \cdot N}$$

and therefore :

$$timeout \leq \sqrt{\frac{4T \cdot N(\rho + (\alpha+1)\ln n)}{n \cdot p_{MAX} \cdot p_N \cdot (1-E_-)}}$$

Assigning this approximation of *timeout* into the above assumption yields :

$$\sqrt{\frac{2T \cdot N}{n \cdot p_{MAX} \cdot p_N(1-E_-)} \cdot 2(\rho + (\alpha+1)\ln n)} < \frac{2T \cdot N}{n \cdot p_{MAX} \cdot p_N(1-E_-)}$$

obtaining the following *sparse connectivity* assumption :

Definition 1. Let a network be called *sparsely connected* when :

$$p_N < \frac{T \cdot N}{n \cdot p_{MAX} \cdot (\rho + (\alpha+1)\ln n)(1-E_-)}$$

We can now obtain the algorithm's completion time and cost :

Theorem 3. *Under the sparse connectivity assumption, the completion time of the* TPP *algorithm is :*

$$T_{Vac} \leq 4\sqrt{\frac{T \cdot N(\rho + (\alpha+1)\ln n)}{n \cdot p_{MAX} \cdot p_N \cdot (1-E_-)}}$$

Theorem 4. *Under the sparse connectivity assumption, the overall cost of the* TPP *algorithm (messages sending and monitoring costs) is :*

$$M \leq k \cdot timeout \cdot C_S + \frac{k}{X} \cdot C_M \leq$$

$$\leq 4n(\rho + (\alpha+1)\ln n)C_S + 2C_M\sqrt{\frac{n(\rho + (\alpha+1)\ln n) \cdot p_{MAX} \cdot (1-E_-)}{p_N \cdot T \cdot N}}$$

Proof. When the vaccination process is completed, no new messages concerning the malicious application are sent. The above is received by assigning the approximated value of timeout into Observation 1.

Definition 2. Let the *vaccination factor* ζ_{T,N,P_M,E_-} be defined as :

$$\zeta_{T,N,P_M,E_-} \triangleq \frac{T \cdot N}{p_{MAX}(1-E_-)}$$

Using the sparse connectivity assumption as an upper bound for p_N and $\frac{\ln n}{n}$ as a lower bound for p_N which guarantees connectivity [16], the following corollaries can now be produced :

Corollary 1. *The completion time of the* TPP *algorithm is :*

$$T_{Vac} = O\left(\rho + \ln n + \frac{T \cdot N}{p_{MAX} \cdot (1 - E_{-}) \cdot \ln n}\right)$$

Proof. Assigning the upper bound for p_N into Theorem 3 immediately yields $O(\rho + \ln n)$. When assigning the lower bound of $p_N = \frac{\ln(n)}{n}$ the following expression is received :

$$T_{Vac} \leq 4\sqrt{\frac{T \cdot N \cdot (\rho + (\alpha+1)\ln n)}{\ln(n) \cdot p_{MAX}(1 - E_{-})}}$$

However, using the sparse connectivity we see that :

$$\frac{\ln n}{n} < \frac{T \cdot N}{n \cdot p_{MAX} \cdot (\rho + (\alpha+1)\ln n)(1 - E_{-})}$$

which in turn implies that :

$$\rho + (\alpha+1)\ln n < \frac{T \cdot N}{\ln(n) \cdot p_{MAX}(1 - E_{-})}$$

Combining the two yields :

$$T_{Vac} = O\left(\frac{T \cdot N}{\ln n \cdot p_{MAX} \cdot (1 - E_{-})}\right)$$

Note that although $O\left(\frac{T \cdot N}{p_{MAX}(1-E_{-})}\right)$ is allegedly independent of n, by assigning the connectivity lower bound $P_N > \frac{\ln n}{n}$ into the sparse connectivity assumption, we can see nevertheless that :

$$\zeta_{T,N,P_M,E_{-}} = \frac{T \cdot N}{p_{MAX}(1 - E_{-})} = \Omega(\rho \ln n + \ln^2 n)$$

For similar arguments, the vaccination's cost can be approximated as :

Corollary 2. *The overall cost of the* TPP *algorithm (messages sending and monitoring costs) is :*

$$M = O\left(k \cdot timeout \cdot C_S + \frac{k}{X} \cdot C_M\right) =$$

$$O\left((n\rho + n\ln n)C_S + \left(\frac{n}{\ln n} + n(\rho + \ln n)\zeta^{-1}_{T,N,P_M,E_{-}}\right)C_M\right)$$

In networks of $E_{-} < 1 - o(1)$, provided that[4] $\rho = O(\ln n)$, and remembering that in this case $\zeta_{T,N,P_M,E_{-}} = \Omega(\ln^2 n)$ we can see that the dominant components of Corollary 2 become :

$$M = O\left(n \ln n C_S + \frac{n}{\ln n} C_M\right)$$

[4] See Section 6 for more details.

4.4 Messages Propagation Among Network Members

In order for a vaccination algorithm which relies on the propagation of valuable information concerning possible threats in the networks to be implemented, the network's devices must have a way of passing relevant information between them. Specifically, the proposed algorithm requires that any network member can perform the following two actions :

- Send a notification message concerning a malicious application to X random network members.
- Forward a received notification message to a single random network member.

Those two commands can be generalized to the requirement that any network member must be able to send upon request a message of some content to a list of (up to X) random network members. The trivial implementation of this feature of the algorithm would use a central server, which will be in charge of storing the network's devices who have registered into the vaccination service. Upon request, the server would be able to provide a list (of any required length) of random network's members. This could either be done one time during the registration of a new device to the service, or be refreshed on a periodic basic.

However, as we would like to decrease the dependency of the system on a centralized component to the minimum, we would prefer this service to be implemented in a decentralized fashion. The implementation of this aspect of the algorithm depends on the specific model of the network. For example, suppose the network is a mobile network, in which each member has a unique address which is its 7-digits phone number, and in which all of the network's node take part in the vaccination efforts. In this example, upon the request to forward a message to a random network member, a device can simply select a random 7-digits number as the message's destination. In case this random number is inactive, the sender will receive an appropriate notification from the network, and will select another number, until successfully guessing an active number.

When a previous knowledge of the addresses of the members of the vaccination service is unavailable, a virtual network overlay in the form of a random graph $G(n,p)$ (of neighbors probability $p = \frac{X}{n}$) can gradually be generated, to be used for messages forwarding (as new devices are joining the service, they are given a list of X random network's members). One example for such an implementation is the use of "random scouts", to be sent by any device joining the service, and returning with a list of X random network members. In addition, each device will periodically execute an additional service, designed for estimating the current number of members in the service. When the value of this number is significantly higher than its number in the previous scouting process, the device will refresh its neighbors list by re-running the scouting algorithm. This can be implemented for example using an approach similar to the one described in [21], which enables the generation of an expander of requested properties from an unknown network, using only $O(n \log n)$ short ($\log^4 n$ length) random walks. Another implementation can be based on the ap-

proach of [20], that uses a *probabilistic group communication service*. Yet another possible implementation can be found at [15].

Note that one may suggest that entire vaccination mechanism can be implemented by a centralized server, receiving all the alerts that are generated by the network's units, and upon the detection of a malware, forward this information to all the members of the network. It is important to note that we would like to refrain from implementing the entire system in one central server for three main reasons. First, such an implementation would make the server a single point of failure, that when compromised, great amount of damage can be caused. Second, implementing the system in the form of a single server that collects all the alerts that are produced by the units significantly increases the chance of a "framing attack" (causing a benign application to deliberately be reported as a malicious one — see Section 6 for more details), creating total havoc in the system, and subsequently resulting in the abandonment of the system as a reliable solution for defending against network attacks. This problem is prevented in the proposed system architecture as the ability of a group of collaborating adversaries to cause damage is highly limited. Third, the purpose of this chapter is to propose a vaccination algorithm that can be implemented apart from in mobile networks also in other networks, including social based services, where the implementation of a centralized server may be hard, or even impossible.

5 Experimental Results

We have implemented the *TPP* algorithm in a simulated environment and tested its performance in various scenarios. Due to space considerations, we hereby describe one example, concerning a network of $n = 1000$ devices, having access to $N = 100$ applications, one of which was malicious[5]. We assume that each device downloads 30 random applications, monitors 1 application every week, and can send notification messages to 10 random network members (namely, $p_N = 0.01$). We require that upon completion, at least 990 network members must be aware of the malicious application (namely, $p_{MAX} = 0.01$), with $\varepsilon = 0.001$. In addition, we assumed that among the network members there are 100 adversaries, trying to deceive at least 50 network devices to believe that some benign application is malicious.

Figure 1 shows the time (in days) and messages required in order to complete this mission, as a function of the decision threshold ρ. We can see that whereas the adversaries succeed in probability 1 for $\rho < 3$, they fail in probability 1 for any $\rho \geq 3$. Note the extremely efficient performance of the algorithm, with completion time of ~ 260 days using only 5 messages and at most 30 monitored applications per user. The same scenario would have resulted in 100 messages per user using the conventional *flooding* algorithm, or alternatively, in 700 days and 100 monitored applica-

[5] Note that the number of malicious applications does not influence the completion time of algorithm, as monitoring and notification is done in parallel. The number of message, however, grows linearly with the number of malicious applications.

tions per user using a non-collaborative scheme. Figure 2 demonstrates the decrease in completion time and message requirement as a result of increasing the penetration threshold p_{MAX}. Figure 3 demonstrates the evolution in the malicious application's penetration probability throughout the vaccination process. An interesting phenomenon is demonstrated in Figure 4, where the number of adversarial devices is gradually increased, and their success in deceiving 5% of the network's members is studied. It can be seen that as the deception rate increases linearly, the success to generate a successful attack displays a phase transition — growing rapidly from "a very low attack success probability" to "very high attack success probability" with the increase of only 20% in the number of adversaries.

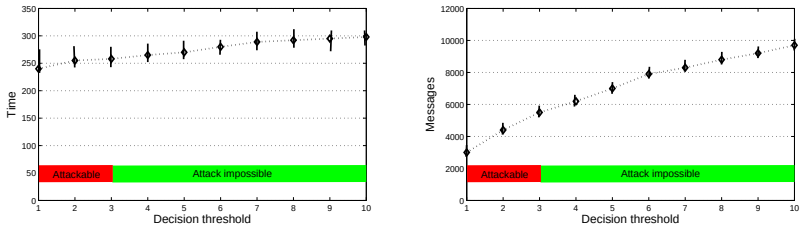

Fig. 1 An experimental result of a network of $n = 1000$ members, with $N = 100$ applications, penetration threshold of $p_{MAX} = 0.01$, $p_N = 0.01$ and 100 adversaries that try to mislead at least 5% of the network into believing that some benign application is malicious. Notice how changes in the decision threshold ρ dramatically effect the adversaries' success probability, with almost no effect on the completion time.

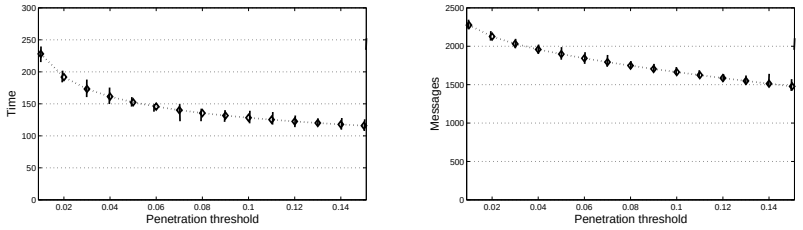

Fig. 2 The effect of decreasing the penetration threshold p_{MAX} on the algorithm's completion time and number of messages ($\rho = 1$).

Many viruses, trojans or other malicious applications contain an integral part in charge of proliferating them throughout the network. We would therefore like to integrate this property into the analysis of our *TPP* vaccination algorithm. For this, we use a variation of the *SIR* (Susceptible—Infectious—Recovered) epidemic model. According to the SIR model, once becoming infected, a network device can infect other devices until it is "cured" (namely, receives the relevant vaccinated informa-

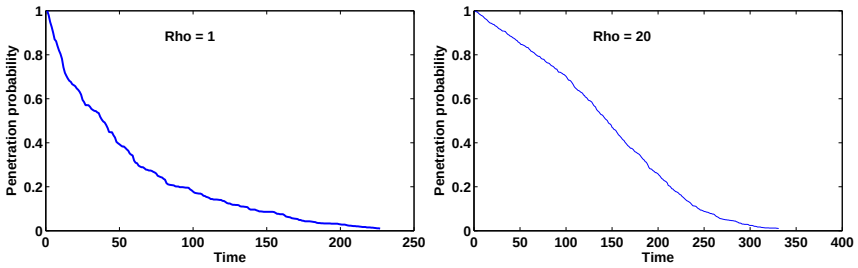

Fig. 3 The *penetration probability* of the malicious application, as a function of the time, with $\rho = 1$ (on the left) and $\rho = 20$ (on the right).

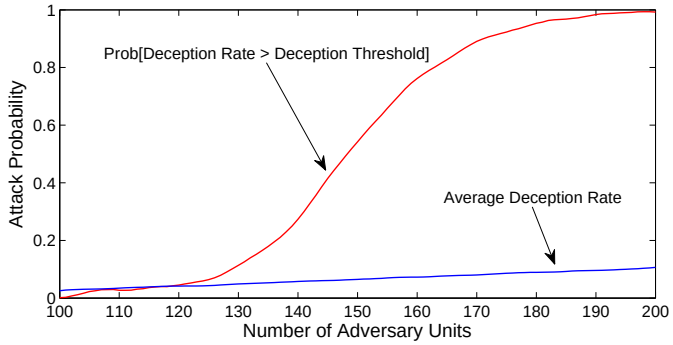

Fig. 4 An illustration of the phase transition that is displayed when observing the influence of the number of adversaries over their probability to successfully generate an attack. Note how an increase of 20% in the number of adversaries increases the probability to deceive a large enough portion of the network from less than 0.2 to approximately 0.8.

tion). Once vaccinated, the device is immune to reinfections of this particular threat. This epidemic model can be represented as follows :

- $Infectious(t) \triangleq I(t) = I(t-dt) + (DI - DR) \cdot dt$
- $Recovered(t) \triangleq R(t) = R(t-dt) + DR \cdot dt$
- $Susceptible(t) \triangleq S(t) = S(t-dt) - DI \cdot dt$
- $New\_recoveries \triangleq DR = \frac{1}{n} I \cdot p_R$
- $New\_infections \triangleq DI = \frac{1}{n} I \cdot \frac{1}{n} S \cdot r_I$

In addition, at any given time, an infected device has a certain possibility of suffering from other symptoms of the infection (such as malicious OS crash, identity theft through key-loggers, etc.). As we would later be interested in the overall number of possible compromised devices, we would like to add this to our model, as follows :

- $Infected\_undmg(t) \triangleq IuD(t) = IuD(t-dt) + (DI - DuR - DD) \cdot dt$

- $Damaged(t) \triangleq D(t) = D(t-dt) + DD \cdot dt$
- $New\_undmg\_recoveries \triangleq DuR = \frac{1}{n} IuD \cdot p_R$
- $New\_damages \triangleq DD = \frac{1}{n} IuD \cdot p_D$

This model is initialized by the following values :

- $p_D \triangleq$ Damage probability
- $p_R \triangleq$ Recovery probability
- $r_I \triangleq$ Infection rate
- $S(0) = n - I_0$
- $D(0) = 0$
- $R(0) = 0$
- $IuD(0) = I(0) = I_0$

Let us denote the maximal level of infectious units throughout the evolution of the epidemic by $\widehat{I}$. A graphic illustration of this model is presented in Figure 5.

Similar to many biological systems, devices infected with malicious applications can self-cure [20,46]. In addition, devices can be cured by receiving assistance from other network's devices (by receiving maliciousness notification messages from enough members of the network). Therefore, p_R is derived from the *TPP* algorithm's performance, and will be discussed in the following sections.

The infection rate r_I is a configurable parameter of the system, denoting the spreading speed of the threat. The value of r_I is provided by the operators or the monitors of the network, and can be changed in response to a detection of new threats. A larger value of r_I would cause the vaccination system to require a larger number of messages for its operation, while a lower value of r_I would allow the system to guarantee a proper vaccination using less messages. As to the initial number of infectious members, I_0, we can assume that it is equal to the initial penetration probability multiplied by the number of network's devices.

The effect of the *TPP* vaccination algorithm on an epidemically spreading network threat is illustrated in Figure 5.

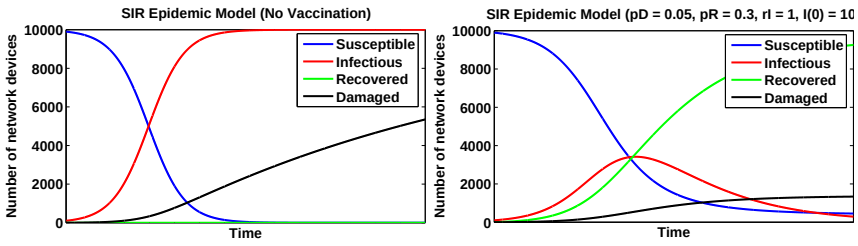

Fig. 5 An illustration of the effect of the *TPP* vaccination algorithm on an epidemically spreading network threat. The epidemic model is initialized with the following values : $n = 10000$, $p_{Damage} = 0.05$, $p_{Recovery} = 0.1$ (due to the vaccination algorithm), $r_{Infection} = 1$, $I_0 = p_{MAX} = \frac{n}{100}$.

6 Avoiding Benign Applications "Frame" Through Adversarial Use of the *TPP* Vaccination Algorithm

As mentioned in previous sections, the *TPP* algorithm is fault tolerant to the presence of adversarial devices which try to compromise the system's integrity by causing a large enough portion of the network to consider (one or more) benign applications as malicious. This aspect of the algorithm is crucial, as it is a collaborative algorithm which relies on the participation of as many network devices as possible — devices who should be guaranteed that the efficient collaborative monitoring will not be accompanied with erroneous results. In the *TPP* algorithm, this fault tolerance is being provided by the introduction of the ρ "decision threshold" into the decision system. Namely, the fact that an application is being marked as malicious only when at least ρ relevant messages (from different origins) are being received. Relying on the common agreement of several devices as a tool for overcoming system noise (which can be either coincidental or intentional) is often used in swarm based systems. For example, a similar mechanism called "threshold cryptography" is used for enabling the collaborative use of cryptographic tasks (see for example [26, 48].

Definition 3. Let us denote by $P_{Attack}(TTL, \rho, \frac{k}{n}, \varepsilon)$ the probability that a "framing attack" done by a group of k organized adversaries will successfully convince at least $\varepsilon \cdot n$ of the network's devices that some benign application a_i is malicious.

A trivial example is the use of very large values for TTL, which allow a group of k adversaries to convince the entire network that any given application is malicious, provided that $k > \rho$, namely :

$$\forall k \geq 1 \quad , \quad \forall \rho \leq k \quad , \quad \forall \varepsilon < 1 \quad , \quad \lim_{TTL \to \infty} P_{Attack}(TTL, \rho, \frac{k}{n}, \varepsilon) = 1$$

Theorem 5. *The probability that k attackers will be able to make at least an ε portion of the network's devices treat some benign application a_i as malicious, using the* TPP *algorithm with a decision threshold ρ is :*

$$P_{Attack}\left(TTL, \rho, \frac{k}{n}, \varepsilon\right) \leq 1 - \Phi\left(\sqrt{n} \cdot \frac{\varepsilon - \tilde{P}}{\sqrt{\tilde{P}(1-\tilde{P})}}\right)$$

where :

$$\tilde{P} = e^{(\rho - TTL \cdot (1 - e^{-\frac{k \cdot p_N}{2}}))} \cdot \left(\frac{TTL \cdot (1 - e^{-\frac{k \cdot p_N}{2}})}{\rho}\right)^{\rho}$$

and where $\Phi(x)$ is the cumulative normal distribution function, defined as :

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt$$

and also provided that :

$$\rho > TTL\left(1-e^{-\frac{k\cdot p_N}{2}}\right)$$

Proof. We use Lemma 2 to calculate the probability that a device $v \in V$ will be reported of some malicious application a_i by a message sent by one of the k adversaries at the next time-step. This is yet again a Bernoulli trial with :

$$p_s = 1-e^{-\frac{(k\cdot n\cdot p_N)}{2n}} = 1-e^{-\frac{k\cdot p_N}{2}}$$

Denoting as $X_v(t)$ the number of times a notification message had arrived to v after t steps, using Chernoff *bound :*

$$P[X_v(t) > (1+\delta)t\cdot p_s] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{t\cdot p_s}$$

in which we set $\delta = \frac{\rho}{t\cdot p_s} - 1$. We can therefore see that :

$$\tilde{P} \triangleq P_{Attack}(TTL,\rho,\frac{k}{n},n^{-1}) = P[X_v(TTL) > \rho] < e^{(\rho - TTL\cdot p_s)}\cdot\left(\frac{TTL\cdot p_s}{\rho}\right)^{\rho}$$

It is important to note that the Chernoff *bounds requires that $\delta > 0$. This is immediately translated to the following requirement, necessary for the validity of this Theorem :*

$$\rho > TTL(1-e^{-\frac{k\cdot p_N}{2}})$$

As we want to bound the probability that at least εn of the devices are deceived, we shall use the above as a success probability of a second Bernoulli trial. As n is large, the number of deceived devices can be approximated using normal distribution, as follows :

$$P_{Attack}\left(TTL,\rho,\frac{k}{n},\varepsilon\right) \leq 1-\Phi\left(\frac{\varepsilon\cdot n - n\cdot\tilde{P}}{\sqrt{n\cdot\tilde{P}(1-\tilde{P})}}\right)$$

and the rest is implied.

Theorem 5 is illustrated in Figures 6 and 8. Figure 6 presents the execution of the vaccination algorithm in a network of 1,000,000 devices, where each device is connected to 50 neighbors. In this example, the network operators require that the number of devices that may be deceived as a result of an adversarial attack would be at most 100. With the decision threshold ρ properly calibrated according to Theorem 5, it is shown that as long as the number of adversaries is below 480 the adversaries cannot launch a successful attack. However, as the number of adversaries increases, such an attack quickly becomes a feasible option. Specifically, by increasing the number of adversaries by 3% (from 480 to 494) the probability of a successful attack rises from 0 to 1. In this case, in order to compensate this change in adversaries numbers, all the devices operating the *TPP* algorithm have to do is simply increase the value of ρ by 1. Figure 8 shows how each such small increase in the value of

the decision threshold ρ requires the adversaries to invest a lot of effort and increase their numbers by approximately 7%. Although severely preventing the adversaries attempts to launch successful attacks, the effect of such changes in the value of ρ on the "normative" devices of the network is very small, as can be observed in Figure 1.

Note that in the proof of Theorem 5 we assumed that the adversarial devices may decide to send a false message concerning an application's maliciousness, but they must do so using the definitions of the *TPP* algorithm. Namely, each device may send at most $p_N \cdot n$ messages, and send these messages to random destinations. If adversarial devices had been allowed flexibility in those constraints as well, a small group of adversarial devices could have sent an infinitely large number of false messages, that would have been propagated throughout the network, resulting in a successful attack (similarly to the case where $TTL \to \infty$). Alternatively, had adversarial devices been allowed to chose the destination of the $p_N \cdot n$ messages they send, they could have all send them to the same destination v, thus guaranteeing that v would be deceived. More generically, a group of $k = 1 \cdot \rho$ of adversarial devices could have send their messages to $i \cdot p_N \cdot n$ different devices, guaranteeing their deception.

Definition 4. Let us denote by $P_{Attack-Destination}\left(TTL, \rho, \frac{k}{n}, \varepsilon\right)$ the attack success probability when adversarial devices are allowed to control the destination of the messages they produce.

The following Corollary can be drawn :

Corollary 3. *The probability that k attackers that can control the destination of the $p_N \cdot n$ messages they produce will be able to make at least an ε portion of the network's devices treat some benign application a_i as malicious, using the* TPP *algorithm with a decision threshold ρ is :*

$$P_{Attack-Destination}\left(TTL, \rho, \frac{k}{n}, \varepsilon\right) \leq P_{Attack}\left(TTL-1, \rho, \frac{k}{n}, \varepsilon - \frac{k}{\rho} \cdot p_N\right)$$

7 Conclusions and Future Work

In this work we have presented the *TPP* TTL-based propagation algorithm, capable of guaranteeing the collaborative vaccination of mobile network users against malicious applications. The performance of the algorithm was analyzed and shown to be superior compared to state of the art in this field, guaranteeing fast collaborative detection of attack attempts, and do so using a lower network overhead.

The algorithm was also shown to be capable of overcoming the presence of adversarial devices who try to inject messages of false information into the network. The challenge of filtering out false information which is injected into a collaborative information propagation networks resembles the known "faulty processors"

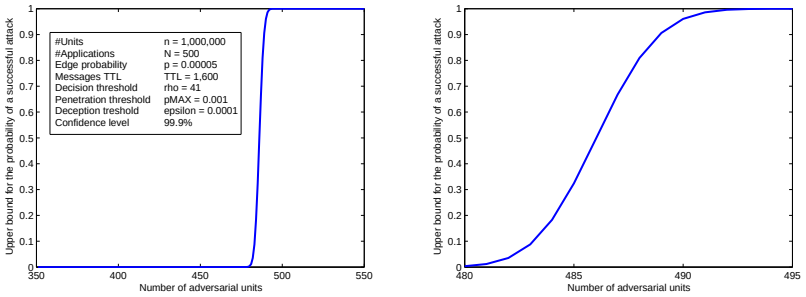

Fig. 6 An illustration of Theorem 5 — an upper bound for the success probability of a collaborative "framing attack" as a function of the number of adversarial devices. In this example, a changing number of adversaries are required to deceive at least 100 devices to think that some benign application is in fact malicious. Notice the phase transition point around 485 adversarial device.

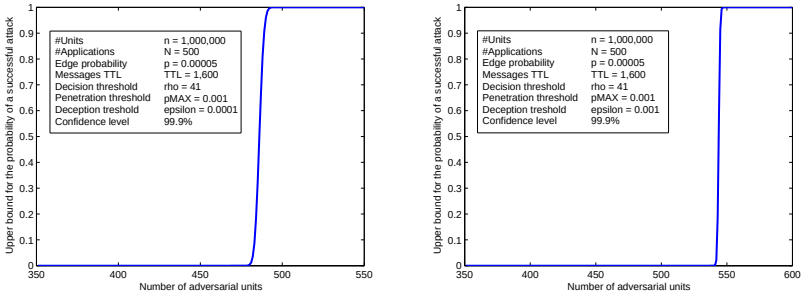

Fig. 7 An illustration of Theorem 5 — the influence of the deception threshold ε (namely, the size of the network portion which is needed to be deceived in order for an attack to be considered successful) on the number of adversarial devices required for a successful attack.

problem. For example, the following work discusses the challenge of synchronizing the clock of a communication network of size n when $\frac{n}{3}$ faulty processors are present [10]. Another interesting work in this scope is the work of [14] which discusses a collaborative fault tolerant "acknowledgement propagation" algorithm, for reporting on the receipt (or lack of) of sent messages.

It should be noted that during the analysis of the fault tolerance of the algorithm, we assumed that although an attacker can send reports of benign applications, or alternatively — refuse to forward messages passed through it, the basic parameters of the algorithm are still preserved. Namely, adversarial devices are not allowed to send more than X messages or generate messages with TTL values higher than the value allowed by the network operator. The exact implementation details of a cryptographic protocol of these properties, however, is out of the scope of this work,

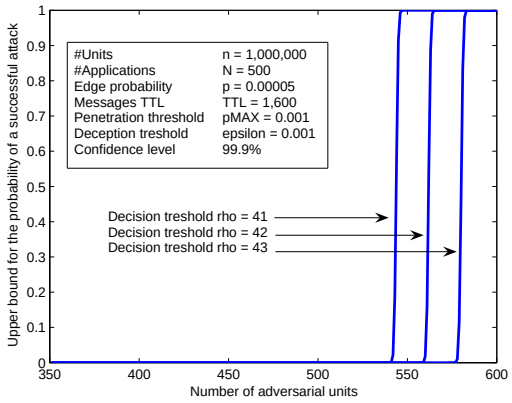

Fig. 8 An illustration of Theorem 5 — the influence of the value of decision threshold ρ on the number of adversarial devices required for a successful attack. Note how as the adversaries increase their numbers, all that has to be done in order to prevent them from launching successful attacks is simply to slightly increase the value of the decision threshold ρ. This feature of the *TPP* algorithm makes it economically favorable for networks operators. From experimental observations it can be seen that the resources required in order to defend a network against attacks of growing scales (both convergence time and number of messages) converge asymptotically, whereas the resources required in order to launch stronger attacks grow approximately linearly.

Future versions of the work should investigate the generalization of the propagation mechanism, allowing the number of alert messages generated to be dynamically calibrated in order to further decrease the algorithm's overhead (for example, as a function of the number of alerts already received concerning the application). Alternatively, upon forwarding a message, devices might be requested to send more than a single copy of the message they received.

Another interesting topic to investigate is the use of the mechanism proposed in this work as an infrastructure for other security related problems. One example can be the problem of collaboratively coping with malicious beacons in hostile wireless environment, as discussed in [43]. Most of the existing localization protocols for sensor networks are vulnerable in hostile environments, requiring for the enhancement of the security of location discovery. The work of [43] presents voting-based methods to tolerate malicious attacks against range-based location discovery in sensor networks. This problem seems to benefit from the use a mechanism which is fault tolerant to the injection of false information, such as the algorithm we propose in this work.

References

1. Mcafee mobile security report 2008. Tech. rep. (2008)

2. Mcafee mobile security report 2009. Tech. rep. (2009)
3. Adamic, L.A., Lukose, R.M., Puniyani, A.R., Huberman, B.A.: Search in power-law networks. Phys. Rev. E **64**(4), 046,135 (2001). DOI 10.1103/PhysRevE.64.046135
4. Altshuler, Y., Wagner, I., Bruckstein, A.: Collaborative exploration in grid domains. In: Sixth International Conference on Informatics in Control, Automation and Robotics (ICINCO) (2009)
5. Altshuler, Y., Yanovsky, V., Bruckstein, A., Wagner, I.: Efficient cooperative search of smart targets using uav swarms. ROBOTICA **26**, 551–557 (2008)
6. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Dist. Comp. **21**, 87–102 (2008)
7. Apap, F., Honig, A., Hershkop, S., Eskin, E., Stolfo, S.: Recent Advances in Intrusion Detection, chap. Detecting Malicious Software by Monitoring Anomalous Windows Registry Accesses, pp. 36–53. Springer Berlin Heidelberg (2002)
8. Aspnes, J., Ruppert, E.: Middleware for Network Eccentric and Mobile Applications, chap. An Introduction to Population Protocols, pp. 97–120. Springer Berlin Heidelberg (2009)
9. Bailey, N.: The Mathematical Theory of Infectious Diseases and its Applications (second edition). Hafner Press (1975)
10. Barak, B., Halevi, S., Herzberg, A., Naor, D.: Clock synchronization with faults and recoveries (extended abstract). In: PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, pp. 133–142. ACM, New York, NY, USA (2000). DOI http://doi.acm.org/10.1145/343477.343534
11. Batalin, M., Sukhatme, G.: Spreading out: A local approach to multi-robot coverage. In: 6th International Symposium on Distributed Autonomous Robotics Systems (2002)
12. Cagalj, M., Hubaux, J., Enz, C.: Minimum-energy broadcast in all-wireless networks: Np-completness and distribution issues. In: MOBICOM (2002)
13. Canalys: Canalys estimates, Canalys (2009)
14. Castelluccia, C., Jarecki, S., Kim, J., Tsudik, G.: Secure acknowledgement aggregation and multisignatures with limited robustness. Computer Networks **50**(10), 1639–1652 (2006)
15. Choy, M., Singh, A.K.: Efficient fault-tolerant algorithms for distributed resource allocation. ACM Trans. Program. Lang. Syst. **17**(3), 535–559 (1995). DOI http://doi.acm.org/10.1145/203095.203101
16. Chung, F., Lu, L.: The diameter of sparse random graphs. Advances in Applied Mathematics **26**, 257–279 (2001)
17. Crisostomo, S., Barros, J., Bettstetter, C.: Flooding the network: Multipoint relays versus network coding. In: 4th IEEE Intl. Conference on Circuits and Systems for Communications (ICCSC), pp. 119–124 (2008)
18. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: In Proc. of the Sixth ACM Symp. on Principles of Distributed Computing, pp. 1–12 (1987)
19. D.F. Zucker M. Uematsu, T.K.: Markup-based smartphone user interface using the web browser engine. In: Proceedings XTech, pp. 25–27 (2005)
20. Dolev, S., Schiller, E., Welch, J.: Random walk for self-stabilizing group communication in ad hoc networks. IEEE Transactions on Mobile Computing **5**, 893–905 (2006)
21. Dolev, S., Tzachar, N.: Spanders: Distributed spanning expanders. In: Proc. of the 25th ACM Symposium on Applied Computing (SAC-SCS) (2010)
22. Erdos, P., Renyi, A.: On random graphs. Publ. Math. Debrecen **6**, 290–291 (1959)
23. Erdos, P., Renyi, A.: On the evolution of random graphs. Publications of the Mathematical Institute of the Hungarian Academy of Sciences **5**, 17–61 (1960)
24. Fragouli, C., Widmer, J., Boudec, J.L.: A network coding approach to energy efficient broadcasting: from theory to practice. In: The 25th IEEE International Conference on Computer Communications (INFOCOM2006), pp. 1–11 (2006)
25. Ganesa, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S.: An empirical study of epidemic algorithms in large scale multihop wireless networks — technical report ucla/csd-tr 02-0013. Technical report, UCLA Computer Science (2002)

26. Gemmel, P.: An introduction to threshold cryptography. CryptoBytes pp. 7–12 (1997)
27. GetJar: Getjar statistics (2010)
28. Golding, R., Long, D., Wilkes, J.: The refdbms distributed bibliographic database system. In: In Proc. of Usenix94, pp. 47–62 (1994)
29. Haas, Z., Halpern, J., Li, L.: Gossip-based ad-hoc routing. IEEE/ACM Transactions of networks **14**(3), 479–491 (2006)
30. Hyponnen, M.: Malware goes mobile. Sci. American **295**, 70 (2006)
31. Hypponen, M.: State of cell phone malware in 2007. Tech. rep. (2007)
32. J. Cheng S. Wong, H.Y.S.L.: Smartsiren: Virus detection and alert for smartphones. In: In Proceedings of the Fifth ACM International Conference on Mobile Systems Applications and Services (MOBISYS), pp. 258–271 (2007)
33. Jacoby, G., Davis, N.: Battery-based intrusion detection. In: Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM04, vol. 4, pp. 2250–2255 (2004)
34. Jacquet, P., Laouiti, A., Minet, P., Viennot, L.: Performance analysis of olsr multipoint relay flooding in two ad-hoc wireless network models. technical report 4260. Technical report, INRIA (2001)
35. Jonasson, J., Schramm, O.: On the cover time of planar graphs. Electron. Comm. Probab. **5**, 85–90 (electronic) (2000)
36. Kim, H., Smith, J., Shin, K.G.: Detecting energy-greedy anomalies and mobile malware variants. In: MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services, pp. 239–252. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1378600.1378627
37. Kleinberg, J.: The wireless epidemic. Nature **449**, 287–288 (2007)
38. Koenig, S., Liu, Y.: Terrain coverage with ant robots: A simulation study. In: AGENTS'01 (2001)
39. Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. Annals of Mathematics and Artificial Intelligence **31**, 41–76 (2001)
40. Kong, C., Peng, N., Rekleitis, I.: Distributed coverage with multi-robot system. In: IEEE International Conference on Robotics and Automation (2006)
41. Korf, R.: Real-time heuristic search. Artificial Intelligence **42**, 189–211 (1990)
42. Lim, H., Kim, C.: Multicast tree construction and flooding in wireless ad hoc networks. In: In Proceedings of the ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM) (2000)
43. Liu, D., Ning, P., Liu, A., Wang, C., Du, W.K.: Attack-resistant location estimation in wireless sensor networks. ACM Trans. Inf. Syst. Secur. **11**(4), 1–39 (2008). DOI http://doi.acm.org/10.1145/1380564.1380570
44. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS '02: Proceedings of the 16th international conference on Supercomputing, pp. 84–95. ACM, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/514191.514206
45. Moskovitch, R., Gus, I., Pluderman, S., Stopel, D., Glezer, C., Shahar, Y., Elovici, Y.: Detection of unknown computer worms activity based on computer behavior using data mining. In: CISDA 2007. IEEE Symposium on Computational Intelligence in Security and Defense Applications, pp. 169–177 (2007)
46. Moskovitch, R., Pluderman, S., Gus, I., Stopel, D., Feher, C., Parmet, Y., Shahar, Y., Elovici, Y.: Host based intrusion detection using machine learning. In: 2007 IEEE Intelligence and Security Informatics, pp. 107–114 (2007)
47. Mutz, D., Valeur, F., Vigna, G., Kruegel, C.: Anomalous system call detection. ACM Trans. Inf. Syst. Secur. **9**(1), 61–93 (2006). DOI http://doi.acm.org/10.1145/1127345.1127348
48. Narasimha, M., Tsudik, G., Yi, J.H.: On the utility of distributed cryptography in p2p and manets: the case of membership control. In: Proceedings of the 11th IEEE International Conference on Network Protocols, pp. 336–345 (2003)
49. Nash, D.C., Martin, T.L., Ha, D.S., Hsiao, M.S.: Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices. Pervasive Computing and Communications Workshops, IEEE International Conference on **0**, 141–145 (2005). DOI http://doi.ieeecomputersociety.org/10.1109/PERCOMW.2005.86

50. Ni, S., Tseng, Y., Chen, Y., Sheu, J.: The broadcast storm problem in a mobile ad hoc network. In: In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pp. 151–162 (1999)
51. Peng, W., Lu, X.C.: On the reduction of broadcast redundancy in mobile ad hoc networks. In: MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing, pp. 129–130. IEEE Press, Piscataway, NJ, USA (2000)
52. Polycarpou, M., Yang, Y., Passino, K.: A cooperative search framework for distributed agents. In: IEEE International Symposium on Intelligent Control, pp. 1–6 (2001)
53. Qayyum, L., Laouiti, A.: Multipoint relaying for flooding broadcast messages in mobile wireless networks. In: Proceedings of HICSS (2002)
54. Rekleitis, I., Lee-Shuey, V., Newz, A.P., Choset, H.: Limited communication, multi-robot team based coverage. In: IEEE International Conference on Robotics and Automation (2004)
55. van Renesse, R.: Power-aware epidemics. Reliable Distributed Systems, IEEE Symposium on **0**, 358 (2002). DOI http://doi.ieeecomputersociety.org/10.1109/RELDIS.2002.1180210
56. van Renesse, R., Birman, K.: Scalable management and data mining using astrolabe. In: In Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS02) (2002)
57. Sasson, Y., Cavin, D., Schiper, A.: Probabilistic broadcas for flooding in wireless mobile ad-hoc networks. In: Proceedings of IEEE Wireless communication and networks (WCNC) (2003)
58. Shevchenko, A.: An overview of mobile device security (available at *www.viruslist.com/en/analysis?pubid=170773606*) (2005)
59. Stojmenovic, I., Seddigh, M., Zunic, J.: Ahbp: An efficient broadcast protocol for mobile ad hoc networks. Journal of Computer Science and Technology **16**(2), 114–125 (2001)
60. Stojmenovic, I., Seddigh, M., Zunic, J.: Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. IEEE Transactions on Parallel and Distributed Systems **13**(1), 14–25 (2002). DOI http://doi.ieeecomputersociety.org/10.1109/71.980024
61. Stone, L.: Theory of Optimal Search. Academic Press, New York (1975)
62. Svennebring, J., Koenig, S.: Building terrain-covering ant robots: A feasibility study. Autonomous Robots **16**(3), 313–332 (2004)
63. Vogels, W., van Renesse, R., Birman, K.: The power of epidemics: robust communication for large-scale distributed systems. SIGCOMM Comput. Commun. Rev. **33**(1), 131–135 (2003). DOI http://doi.acm.org/10.1145/774763.774784
64. Wagner, I., Altshuler, Y., Yanovski, V., Bruckstein, A.: Cooperative cleaners: A study in ant robotics. The International Journal of Robotics Research (IJRR) **27**(1), 127–151 (2008)
65. Wang, P., Gonzalez, M., Hidalgo, C., Barabasi, A.: Understanding the spreading patterns of mobile phone viruses. Science **324**, 1071–1075 (2009)
66. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: MOBIHOC, pp. 9–11 (2002)
67. Zlot, R., Stentz, A., Dias, M., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proceedings of the IEEE International Conference on Robotics and Automation (2002)