Bluetooth Worms: Models, Dynamics, and Defense Implications

Guanhua Yan and Stephan Eidenbenz*
Discrete Simulation Sciences (CCS-5)
Los Alamos National Laboratory
{ghyan, eidenben}@lanl.gov

Abstract

Recent occurrences of mobile worms like Cabir, Mabir and CommWarrior have created growing concerns over the security of data stored on mobile devices such as cell phones and PDAs. These worms have in common that they all use Bluetooth communication as their infection channel. In order to prepare effective defense strategies against such worms, we study the nature, characteristics, and spreading dynamics of Bluetooth worms in the safe environment of simulation. Our key findings are: (i) Mobility may not boost the Bluetooth worm propagation; instead, link instability owing to it has negative impact on the worm spreading speed; (ii) The inherent capacity constraints imposed by the wireless channel (e.g. interference) and the specifics of the Bluetooth protocol can significantly slow down the Bluetooth worm propagation; (iii) Intelligently designed worms can improve their propagation speed to a noticeable degree by strategically selecting worm model parameters or exploiting out-of-band propagation capabilities.

1 Introduction

Bluetooth, originally designed as a cable replacement solution, is a short-range radio technology that connects wireless devices. It differentiates itself from other competing radio technologies such as IEEE 802.11 by operating at low power consumption and at low cost. Bluetooth has a wide range of applications, including wireless headsets, peer-to-peer file exchanges, and data synchronization. The market for Bluetooth devices has been growing tremendously in recent years: world-wide, 272 million Bluetooth devices have been shipped in 2005, twice as many as in 2004 [18].

The wide-spread deployment of Bluetooth devices has made the technology attractive for worm propagation. The first cell phone worm called Cabir [9], which hit mobile cell phones in 2004, used Bluetooth channels on cell phones running the Symbian Operating System to spread onto other phones. The Cabir descendant Mabir [6] and the CommWarrior worm [13] are both capable of propagating themselves through the Bluetooth interfaces of cell phones. While these worms created considerable nuisance by draining the batteries of infected devices due to intensive scanning operations and probably also by congesting the wireless channel, they have not caused any serious security breaches as none of them actually carried a malicious payload. However, security concerns over Bluetooth worms that spread on cell phone networks are hard to exaggerate: once a worm has compromised a cell phone, it can easily place bogus calls, propagate spam emails, and steal confidential or private information that is stored on the cell phone [10][14]. More advanced worms might gain control over a large number of cell phones in which they implant zombies. These resulting wireless botnets could be used to launch Distributed Denial of Service (DDoS) attacks against base stations, cellular switches, specific IP addresses or phone numbers such as emergency numbers [10][7].

The potential security impact of Bluetooth worms on cell phone networks calls for an in-depth understanding of their nature, characteristics, and dynamic behavior. We are particularly interested in the propagation speed of Bluetooth worms. Examples of extremely fast Internet worms like Slammer [16] tell us that manual human intervention is hardly effective in stopping such worms. We now face a similar question: how fast can a worm propagate in a Bluetooth network? To answer it, we conduct a simulation study of Bluetooth worms. Mathematical analysis of MAC/PHY layer interactions between Bluetooth devices in a network at the scale in our study is difficult, if not impossible. On the other hand, it is costly to deploy a real Bluetooth testbed and it is also difficult to control the speed of real Bluetooth devices in the way as what we want. Hence, simulation, due to its repeatability and controllability in a laboratory setup, stands out as the only practical tool to conduct this study.

In this paper, we propose a baseline worm model that mirrors the designs of existing Bluetooth worms such as Cabir and CommWarrior. We use simulations to understand the propagation property of this baseline worm in a variety of scenarios. We observe that the spreading speed of

*Los Alamos National Laboratory Publication No. LA-UR-06-1476

Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06)
0-7695-2716-7/06 $20.00 © 2006 IEEE

the Bluetooth worm depends on factors such as device density, device mobility, and existence of insusceptible devices. We also observe that the Bluetooth worm propagation time grows almost linearly with the network size.

An important finding from our simulation results is that mobility does not necessarily boost the Bluetooth worm propagation. This disagrees with the observation made in some earlier work. Further examination on the dynamics of the Bluetooth worm reveals that link instability due to device mobility forces the Bluetooth worm to rely on timers to detect connection failures, thus slowing down its spreading speed. Another observation is that as a Bluetooth network of high device density becomes populated with infected devices, heavy co-channel interference may result and in return slows down the worm propagation process itself. After a model fitting exercise, we conclude that the logistic model, which has successfully been used to characterize the Internet worm propagation, fails to fully capture the dynamics of Bluetooth worm propagation. We thus provide some guidelines on what properties of Bluetooth worms should be considered when building an analytical model.

We further explore whether an advanced Bluetooth worm can strategically select its model parameters to maximize its propagation speed. We find that a Bluetooth worm that aggressively carries out its infection activities may not improve its propagation speed and on the contrary, if devices in the network are relatively static, its propagation speed can be negatively affected. As a reflection of recent worms such as CommWarrior that exploit communication capabilities besides Bluetooth, we examine how availability of MMS (Multimedia Message Service) accelerates Bluetooth worm propagation. The simulation results reveal that with only a couple of MMS contact "buddies" per device a Bluetooth worm can speed up its propagation by 50%.

We structure this paper as follows. Section 2 provides a brief overview of Bluetooth technology as background knowledge. We present the design of a baseline worm model in Section 3 and the experimental results with this model in varied scenarios in Section 4. We analyze the spreading dynamics of the Bluetooth worm in Section 5 and provide some considerations from the modeling perspective in Section 6. We explore the effect of model parameters in Section 7 and investigate how long-range infection channels can boost Bluetooth worm propagation in Section 8. Section 9 briefly presents some implications on the defense against Bluetooth worms. Section 10 discusses related work and Section 11 summarizes this paper.

2 Bluetooth Primer

Bluetooth is a short-range radio technology that is aimed at connecting different wireless devices at low power consumption and at low cost. It operates in the 2.4GHz frequency band and its channels are shared among devices through a time-division duplexing (TDD) scheme. It also uses frequency hopping scheme to reduce interference.

When a Bluetooth device wants to find other devices in its vicinity, it broadcasts inquiry packets by hopping 3,200 times per second along a 32-channel inquiry hopping sequence. A nearby device in the *discoverable* mode listens on the same frequency sequence but moves forward its listening carrier every 1.28 seconds. When a device hears an inquiry packet, it backs off for a random period of time and then reenters the scanning state. When it receives another inquiry packet, it responds with a *Frequency Hop Synchronization* (FHS) packet. On the arrival of this packet, the inquirer device knows that the responder is in its radio range.

Once a device has discovered its neighboring devices, it may want to establish a connection with one or more of them. In order to set up a Bluetooth link with a neighbor device, it goes through the paging process. This process is similar to the inquiry process, except that the paging device explicitly specifies the receiver's address to indicate with which device it wants to set up a connection. After a connection is established, the pager device and the paged device are called the *master* and *slave* of the new link respectively. In the connected state, the master and the slave can exchange normal data packets by hopping 1,600 times per second along a 79-channel frequency sequence decided by the master's local clock and its device address.

A Bluetooth link has a maximum capacity of 1Mbps. The Bluetooth specification defines two types of links: synchronous connection-oriented (SCO) links for voice communication and asynchronous connectionless (ACL) links for data communication. ACL links use the Automatic Repeat Request (ARQ) scheme to recover lost packets. ACL links support 6 types of packets, among which DH5 packets have the highest data rates. The payload of each DH5 packet has 339 bytes.

3 A Bluetooth Worm Model

We present the design and implementation of our Bluetooth worm in a Bluetooth protocol simulator. In our Bluetooth worm model, an infected device only attempts to infect devices within its range. It does not need the networking layer to route packets to devices multiple hops away. Such design captures the behavior of existing Bluetooth worms like Cabir and CommWarrior. This single-hop propagation behavior is the key difference from Internet worms.

The infection cycle of a Bluetooth worm is illustrated in Figure 1. When a Bluetooth worm is activated, it starts looking for Bluetooth-enabled devices in its vicinity. When it starts an inquiry, it specifies both the inquiry timeout, T_{inq}, and the expected number of responses, N_{inq}. If the infected device sees the N_{inq}-th response before T_{inq} time units after it starts the inquiry, the inquiry process finishes; otherwise, if the infected device receives less than N_{inq} re-

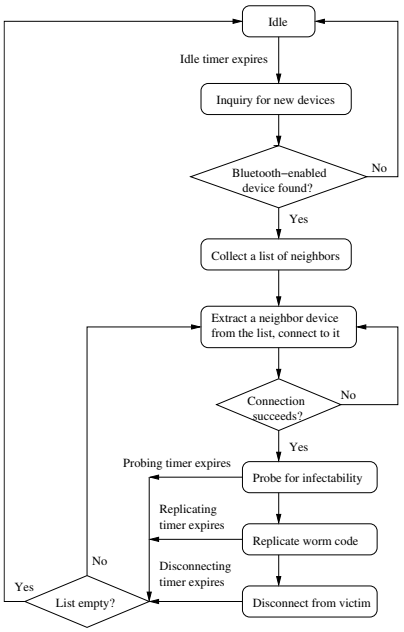


Figure 1. Infection Cycle of A Bluetooth Worm

sponses when T_{inq} time units elapse after it initiated the inquiry, the inquiry process also terminates. The default value for T_{inq} in the Bluetooth specification is 10.24 seconds, which ensures that all devices can be discovered in a loss-free environment if there is no limit on N_{inq}.

After the Bluetooth worm has compiled a list of Bluetooth-enabled devices in its radio range, it iterates through the list attempting the following with each neighbor device: establish a connection, probe infection possibility, replicate itself, and disconnect from the victim device.

Establishing a connection to a victim device involves the paging process. It is, however, possible that paging fails. One obvious reason is that the victim device has moved out of radio range at time of paging request. Paging failures can also result from interference. For example, if two devices page each other simultaneously, they both will fail; if a device pages a node that is performing an inquiry, it will also fail [19]. In order to detect paging failures, a Bluetooth device schedules a paging (or connection establishing) timer that expires after T_{conn} time units. The default value of T_{conn} in the Bluetooth specification is 5.12 seconds.

If the infected device fails to establish a connection to a victim device, it removes it from the list and attempts to infect the next one; otherwise, it probes the victim device for the infection possibility. When the victim device receives the probing packet, its response depends on its internal state: (1) if at state *insusceptible*, it sends back a REJECTED response packet; (2) if at state *susceptible but infected*, it sends back an INFECTED response packet; (3) if at state *susceptible and uninfected*, it sends back an UNINFECTED response packet.

On the arrival of a REJECTED or INFECTED response packet, the probing device removes the victim device from the list and attempts to infect the next one. If the probing device finds that the victim device is vulnerable and has not yet been infected, it starts to replicate the worm code onto the victim device. Similar to the paging process, the probing device may not receive any response from the victim device. Hence, it starts a timer as it sends out the probing packet. The probing timer expires after T_{probe} time units if no response has been received. When the timer fires, the worm attempts to infect the next device on the list.

The probing process in our worm model closely mirrors the behavior of most real Bluetooth worms. For instance, a CommWarrior worm probes each victim device for the availability of the 'Obex Push' service; on a positive reply, the worm replicates itself onto that device [8]. Moreover, if a victim device is susceptible to the worm attack, our model distinguishes two cases: it has not been infected, and it has already been infected. This is possible because a worm, after gaining control over a victim device, can inform probing devices of the fact that the probed device has already been infected. This prevents the worm from replicating itself to the same device unnecessarily for more than one time.

The time that a worm takes to upload itself onto another device depends on its size S_{worm} and the packet type. We assume all worm packets are DH5 packets. The size of a Bluetooth worm seen so far ranges from thousands to tens of thousands of bytes. For example, the Cabir.H worm (a variant of Cabir worm) consists of only about 7,000 bytes; the CommWarrior.a!sys worm (a variant of CommWarrior worm) has 30,582 bytes [2].

Worm replication may fail because devices move out of each other's radio range. Thus, a replication timer is started when an infected device starts uploading. If it expires after T_{rep} time units, the infected device gives up the attempt and tries to infect the next one on its neighbor list. After a worm successfully copies itself onto another device, it disconnects from it. In case that the communication channel is lost during this period, we also schedule a timer to ensure that the channel between them is destroyed successfully. The disconnection timeout is T_{disc} time units in our worm model.

After attempts to replicate itself onto all the devices found in the vicinity, a worm keeps inactive for T_{idle} time units before another infection cycle starts.

4 Baseline Worm Experiments

We implemented the Bluetooth worm model described in Section 3 in the ns-2 network simulator [3] using the Bluetooth module UCBT [4]. The UCBT module has a full implementation of the Bluetooth protocol stack. However, it does not provide a detailed radio propagation model. Hence, we have implemented the radio propagation model used in [11] to calculate signal attenuation. In this model, the path loss, which refers to signal attenuation on the prop-

Notation	Value	Notation	Value
T_{inq}	10.24 (sec)	N_{inq}	7
T_{conn}	5.12 (sec)	T_{probe}	0.1 (sec)
T_{rep}	10 (sec)	S_{worm}	20,000 bytes
T_{disc}	0.1 (sec)	T_{idle}	20 (sec)

Table 1. Notations and values in the baseline worm model

agation path, is defined as follows:

$$P_{loss} = \begin{cases} 20 \times log_{10}(4\pi d/0.1224) & d \leq 8m \\ 58.3 + 33 \times log_{10}(d/8) & d > 8m \end{cases} \quad (1)$$

This model assumes free space propagation for the first 8 meters (i.e., it assumes that there always exists an unobstructed line-of-sight path between the source and the destination); beyond this range, the signal attenuates at a rate that is a function of $d^{3.3}$, where d is the distance to the source.

Interference is a major limiting factor in the performance of Bluetooth networks [11]. In our simulation study, we take co-channel interference into consideration. Co-channel interference occurs when devices on different channels use the same frequency to transmit packets at the same time. The Bluetooth specification requires that in order for the receiver to successfully decode a signal, the carrier to interference ratio (C/I) must be at least 11 dB. In this study, we assume that a lower C/I ratio leads to packet loss.

We configure the parameters in our baseline Bluetooth worm model as illustrated in Table 1. In all the experiments in this paper, we simulate Bluetooth networks operating at power class 2; thus, each Bluetooth device has a communication range of 10 meters. In the following sections, we investigate how the worm propagation speed is affected by varied scenario parameters, including device density, device mobility, network size and fraction of insusceptible devices.

4.1 Effects of Density and Speed

In this set of experiments, we simulate a network with 200 Bluetooth devices. We vary device densities by putting these devices in square areas with side length 50, 75, 100, 125, and 150 meters. An important part of our simulation study is how to model device mobility. Ideally, we would like to use traces from real Bluetooth networks. Unfortunately, the few publicly available mobility datasets are not suitable as they do not provide position data of each device at fine-grained time scales. Under such circumstance, we apply the widely used random waypoint mobility model to generate device movement traces despite its well-known flaws. In this model, a node stays at a location for a uniformly distributed period between τ_{min} and τ_{max}, then chooses another location randomly and moves towards it at a uniformly distributed speed between v_{min} and v_{max}. In all the simulation studies in this paper, we have: $\tau_{min} = 0(sec)$, $\tau_{max} = 10(sec)$, $v_{min} = 0(m/s)$. In this set of experiments, we vary the mean speed (i.e., $(v_{min} + v_{max})/2$) among 1, 2, 3, 4, and 5 m/s. For each scenario we simulate 10 sample runs.

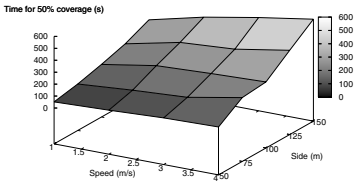
Figure 2 depicts the average propagation time needed for 50% and 95% infection coverage in 3-dimension graphs. We observe that given the same mean device speed, both times increase roughly linearly with the side length. The simulation results agree with our intuition that Bluetooth worms should propagate more slowly in a sparse network than in a dense network. On the other hand, given the same device density, both propagation times seem to grow roughly linearly with the mean speed. The observed linearity, however, may hold only in the parameter space examined. It is worth mentioning that our simulation results disagree with the observation made in [15] that mobility boosts viral propagation in mobile environments. The key reason for this different conclusion is that our work considers detailed MAC/PHY layer characteristics. We will present detailed explanation later in Section 5.

4.2 Effects of Insusceptible Devices

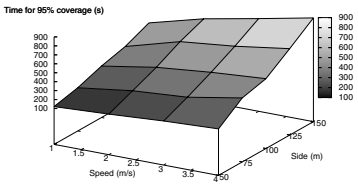
Real Bluetooth networks usually mix susceptible and insusceptible devices. For instance, Cabir worm only spreads on smart cell phones running Symbian OS 6.1 or higher; it can not infect those that run other operating systems. An interesting question, then, is to understand how coexistence of susceptible and insusceptible devices affects worm propagation in Bluetooth networks. We consider a network with the same number of Bluetooth devices. There are two conflicting factors that may affect worm propagation. On one hand, existence of insusceptible devices eliminates the necessity of replicating worm code onto these devices, but on the other hand, it also lowers the probability that two susceptible devices are within each other's radio range.

To understand which factor is important in Bluetooth worm propagation, we perform another set of experiments where susceptible and insusceptible devices are mixed to varied degrees. The proportion of susceptible devices is chosen among 0.25, 0.5, 0.75, and 1.0. We still use the random waypoint mobility model to generate device movement traces; we vary its mean speed between 1 and 3 meters per second. We also vary device densities by simulating 200 devices in square areas with side 75 and 150 meters.

Figure 3 depicts how varied fractions of susceptible devices impact the worm propagation time for 50% and 95% coverage. It is clear that in all scenarios both times decrease monotonically as the fraction of susceptible devices increases. This reveals that the second factor plays a dominant role in affecting worm propagation. We also find that when the device density is relatively low, increasing the fraction of insusceptible devices leads to super-linear growth of worm propagation time.

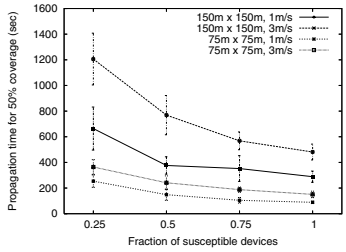


(1) 50% infection coverage

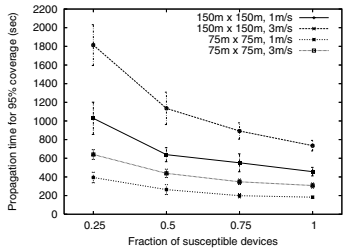


(2) 95% infection coverage

Figure 2. Propagation times under different device densities and mean device speeds

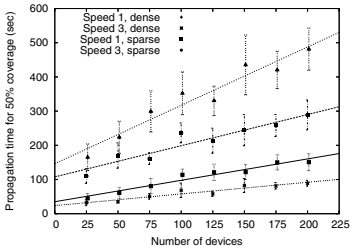


(1) 50% infection coverage

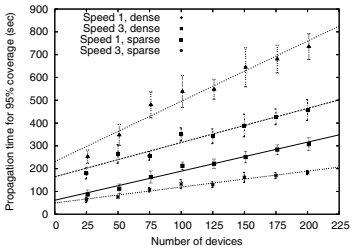


(2) 95% infection coverage

Figure 3. Propagation times under different fractions of susceptible devices (95% confidence interval)



(1) 50% infection coverage



(2) 95% infection coverage

Figure 4. Propagation times under different network sizes (95% confidence interval)

4.3 Effects of Network Size

The total number of devices in the network considered is a key parameter. We simulated our worm on several network sizes with the aim to see how the propagation speed changes with growing network size, while keeping the device density constant. We vary device densities in the network between 0.0356 device/m^2 (dense) and 0.0089 device/m^2 (sparse). We also vary the mean speed in the random waypoint model between 1 and 3 meters per second. The results, including 95% confidence intervals, are presented in Figure 4. The figure shows average propagation times for 50% and 95% infection coverage as a function of network size under four scenarios. It seems that in all cases both times grow linearly with the network size, except that the ones corresponding to mean speed 3m/s and low density grow slightly sub-linearly. We use least squares linear regression technique to fit the simulation results and the derived linear curves are also depicted in Figure 4. The observed linearity allows us to extrapolate worm propagation times in Bluetooth networks at scales beyond simulation capabilities as of today. Consider a network that has one million Bluetooth devices in a 10^8 m^2 area. This network has almost the same device density as the aforementioned sparse scenarios. If we assume that all devices are vulnerable to worm infection and move according to the random waypoint model at mean speed $1m/s$, then using the slopes derived from Figure 4, we can infer that it would take approximately 10 days for the worm to infect half of the population and 17 days to infect 95% of all the devices.

5 Dynamics Analysis

In the previous section, we have presented how Bluetooth worms propagates under varied scenarios. In this section, we give a more detailed analysis on the dynamics unique to the Bluetooth worm propagation. We perform two new sets of experiments, both simulating a network with 200 Bluetooth devices in a $75m \times 75m$ area. In the first set, devices remain stationary throughout the simulation, and in the second one, devices move according to the random waypoint mobility model. In each experiment, we randomly choose a Bluetooth device and let it be infected at the beginning of the simulation.

5.1 Results for Static Devices

We simulate the static network for 300 seconds. Every device in the network is susceptible to the worm infection. We perform 10 sample runs, in each of which devices are randomly distributed in the area (we throw away those that do not yield a fully connected network).

Propagation curve. For ease of explanation, we define notation $T_{p\%}$ to be the time in seconds needed to infect p percent of the devices in the network. Figure 5 presents the cumulative numbers of infected devices as a function of simulation time in three sample runs. These sample runs are selected in the following way: We rank the 10 sample runs based on their $T_{50\%}$ and choose the one with the smallest value, the one with the largest value, and the one with the median value. We also plot the curve that averages all the sample runs. From the graph, we observe that there exists

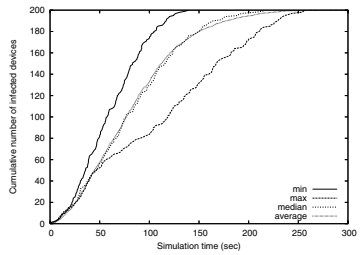


Figure 5. Cumulative number of infected devices as a function of time (Static, 75m×75m)

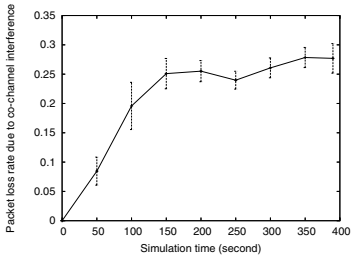


Figure 6. Packet loss rate due to co-channel interference (Static, 75m×75m, 95% confidence interval)

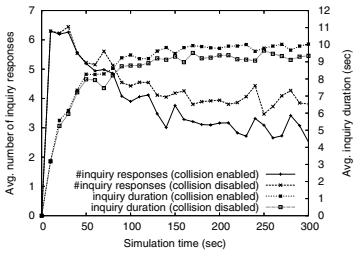


Figure 7. Average number of responses per inquiry and average inquiry duration (Static, 75m×75m)

large variance in worm propagation speed among the sample runs. The largest $T_{50\%}$ is twice as much as the smallest one. This is because we randomly place the devices in the area and the initial worm is also randomly chosen. On the other hand, the propagation curve of the sample run with the median $T_{50\%}$ is very close to the propagation curve that averages all the sample runs. Among all the sample runs, $T_{95\%}$ has mean 161.35 and standard deviation 37.41; $T_{50\%}$ has mean 80.15 and standard deviation 21.35.

Interference Analysis. Throughout each simulation run, we also collect the average packet loss rate due to co-channel interference every 10 seconds. We denote it by $L_{p,co}$. The result with 95% confidence interval is depicted in Figure 6. We observe the general tread of increasing packet loss rate due to co-channel interference; this is because as the number of infected devices grows, interference among devices also increases. When almost all the devices in the network are infected, $L_{p,co}$ reaches its peak at about 27%; after that point, the network enters a relatively stable state and $L_{p,co}$ thus becomes steady.

It is interesting to see how much co-channel interference slows down worm propagation. In another set of experiments, we do not drop packets when co-channel interference occurs. Simulation results tell us that without packet losses due to co-channel interference, $T_{95\%}$ has mean 119.10 and standard deviation 20.28; $T_{50\%}$ has mean 67.06 and standard deviation 14.85. Hence, for the 95% infection coverage, co-channel interference slows down worm propagation by about 36%, and for the 50% infection coverage, co-channel interference leads to about 20% slowdown in spreading speed.

Phase Analysis. To gain further understanding on Bluetooth worm propagation, we analyze the three major phases in a worm infection cycle: searching for nearby devices, establishing a connection to a particular device, and replicating worm code onto a victim device. Figure 7 presents the average number of responses received in an inquiry process. We observe that as more devices are infected, fewer devices respond to those infected devices searching victims in their vicinities. This observation still holds even when we disable packet losses due to co-channel interference. Further examination on the experimental results reveals that among all the devices that do not respond to inquiries, most of them have already been infected. This occurs when they also engage in searching nearby devices or infecting a new victim. As more devices in the network are infected, it is more likely that an infected device is surrounded by others that are also searching for new victims.

This seems helpful in accelerating worm propagation because those devices that do not respond have already been infected and failing to find them saves time from establishing connections to them and then probing them. These savings, however, may be offset by the increasing inquiry durations when more devices are infected. In Figure 7, we also depict how the average inquiry duration evolves in the simulation. It is clear that, when the number of infected devices increases, the time needed to finish an inquiry grows dramatically towards its upper bound T_{inq} (which is 10.24 seconds in the baseline worm model). At the beginning of the simulation, because most of the devices are not infected, a device scanning nearby devices can receive up to N_{inq} (which is 7 in the baseline worm model) responses much earlier before T_{inq} time units elapse. Hence, during this period, an infected device is able to finish the inquiry process quickly. As the number of infected devices increases, some of them do not respond to inquiries launched by nearby devices, as we explained above. Many inquiries are thus finished only when their associated timers expire after T_{inq} time units. This results that average inquiry duration approaches to T_{inq} with increasing saturation of infected devices in the network.

Figure 8 describes the results on connection establishing attempts per 10 seconds. Not surprisingly, both successful and failed attempts at establishing connections to infected devices increase with the growing number of infected devices in the network. Furthermore, establishing a connection to an uninfected device seldom fails. A notable observation from the graph is that among attempts at connecting

to infected devices, failed ones outnumber successful ones almost by 100% after about 25 seconds since the initial infection. Moreover, Figure 9 tells us that as the network is saturated with Bluetooth worm, a successful connection establishing process takes about 2.0 seconds.

In Figure 9, we also present the average amount of time spent on replicating the worm code. If there is no box during a time period, it indicates that no worm replication occurs. Further examination reveals that the duration of worm code replication is severely impacted by the co-channel interference. If we disable packet losses due to co-channel interference, the durations have very small variance.

5.2 Results for Mobile Devices

Propagation curve. We simulate the same network in Section 5.1 for 700 seconds except that nodes are not stationary. Figure 10 presents the cumulative numbers of infected devices as a function of simulation time in several sample runs and the propagation curve that is averaged over all the sample runs. The sample runs shown in this graph are selected in the same way as in Figure 5. The variance among the sample runs is still large, which is similar to the observation we made from the static case. The largest $T_{50\%}$ is almost twice as much as the smallest one. A close examination on the graph further tells us that the shapes of these curves are actually quite similar, but the differences in the propagation speeds at their early phases are noticeable. This suggests that the propagation speed of the Bluetooth worm in a mobile network is significantly impacted by how fast it spreads itself shortly after it is released.

From the simulation results, $T_{95\%}$ has mean 307.76 and standard deviation 38.61; $T_{50\%}$ has mean 150.67 and standard deviation 35.04. Compared with the worm propagation in the static network, $T_{95\%}$ has increased by 90.7% and $T_{95\%}$ has increased by 86.7%. Here, we do not intend to make a hasty conclusion that mobility must lead to slowdown in worm propagation. Imagine a situation in which a static network consists of several disjoint "islands" between which there is no communication path. In this network, infecting a device in one island can only lead to worm infections in that particular island but devices in the other islands will never be infected. But if mobility is introduced, devices from different islands can be mixed and it is possible that all devices are eventually infected.

Investigating how mobility models affect Bluetooth worm propagation is out of the scope of this paper. Here, provided the observation that mobility can slow down Bluetooth worm propagation, we are thus motivated to expose the characteristics inherent in the Bluetooth protocol that leads to it. The findings can gain us more insights into the dynamics of Bluetooth worm propagation. If an analytical model is necessary, they can also provide a guideline on deciding what details in the Bluetooth protocol should be captured. In this regard, our work differs from some earlier work [15] that attempts to build an analytical model directly without taking the MAC/PHY properties into consideration.

Interference Analysis. Figure 11 illustrates the packet loss rate due to co-channel interference. During the first 50 seconds, as the number of infected devices increases, packet loss rate due to co-channel interference ramps up to about 27%. Thereafter, the packet loss rate due to co-channel interference fluctuates around 27%. Comparing this graph with Figure 6, we notice that as most of the devices are infected, the average packet loss rates due to co-channel interference are similar to those in the static network. To examine how co-channel interference slows down worm propagation, we disable packet losses due to co-channel interference in the simulation. The results show that $T_{95\%}$ has mean 215.63 and standard deviation 34.80; $T_{50\%}$ has mean 130.09 and standard deviation 38.25. Hence, for the 95% infection coverage, co-channel interference slows down worm propagation by about 43%, and for the 50% infection coverage, co-channel interference leads to about 16% slowdown in spreading speed.

We, however, expect that the impact of co-channel interference on worm propagation speed is a function of device density in the network. It is obvious that denser devices lead to more congestion in communication channels and thus lower propagation speed. This is confirmed by some other simulation results. Use the same network in the 150m$\times$150m area as an example. Co-channel interference causes only 9% slowdown in 95% infection coverage and 3% slowdown in 50% infection coverage. Hence, the worm propagation slowdown due to co-channel interference is much smaller than that in the 75m$\times$75m case.

Phase Analysis. We plot the average number of responses per inquiry and the average inquiry duration in Figure 12. We can make similar observations as from Figure 7: as the network has more infected devices, the average number of responses per each inquiry decreases, and correspondingly, the average inquiry duration approaches to the inquiry timeout value; as the majority of the devices are infected, both measures become relatively steady.

Figure 13 shows the results on connection establishing attempts in the mobile Bluetooth network. Comparing it with Figure 8, we notice that in the mobile network connection establishing attempts fail more frequently. We mentioned that the proportion of failed attempts at establishing connections to already infected devices in the static network is about $2/3$. By contrast, in the mobile network this proportion reaches $7/8$. Moreover, in the mobile network the number of failed attempts at building connections to uninfected devices is also larger than that in the static network. All these observations suggest that device mobility increases the difficulty of setting up connections between two devices, thus making it harder for worm propagation.

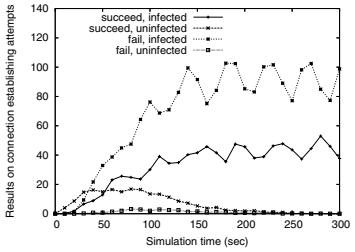


Figure 8. Results on successful and failed connection establishing attempts (Static, 75m×75m)

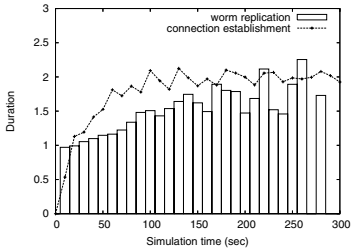


Figure 9. Average duration of connection establishment and worm replication (Static, 75m×75m)

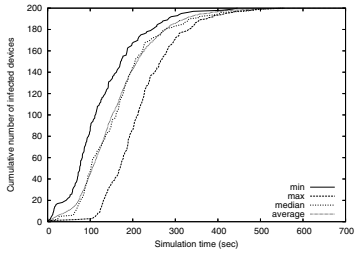


Figure 10. Cumulative number of infected devices as a function of time (Mobile, 75m×75m)

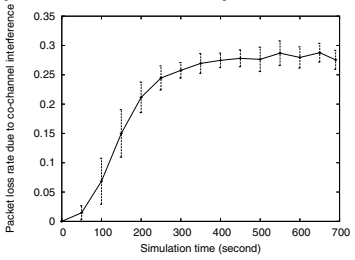


Figure 11. Packet loss rate due to co-channel interference (Mobile, 75m×75m, 95% confidence interval)

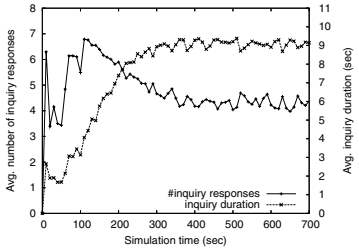


Figure 12. Average number of responses per inquiry and average inquiry duration (Mobile, 75m×75m)

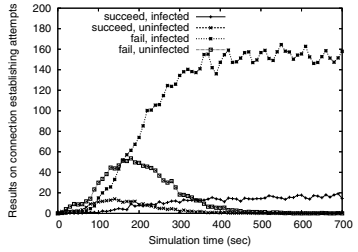


Figure 13. Results on successful and failed connection establishing attempts (Mobile, 75m×75m)

Figure 14 depicts the average time needed to successfully establish a connection. After most of the devices are infected, it is maintained between 1.5 and 2 seconds, which is similar to the static case. The same graph also gives the average amount of time needed to replicate the worm code onto a victim device. If we disable packet losses due to co-channel interference, very small variance is observed on the duration of worm code replication. This conforms to our earlier finding that co-channel interference is the major factor affecting the performance of worm code replication.

6 Modeling Considerations

An accurate worm propagation model can provide useful guidance for effective detection and defense. There have been substantial efforts on analytically characterizing the dynamics of Internet worms. A natural question is whether existing Internet worm propagation models can be directly borrowed to model Bluetooth worm propagation. To answer this question, our simulation results can serve for validation purpose and also provide guidances on what elements should be captured in a good analytical model.

We start from the logistic model proposed in [17]:

$$N(t) = N_{dev} \times \frac{e^{K(t-T)}}{1 + e^{K(t-T)}}, \qquad (2)$$

where $N(t)$ be the number of infected devices at time t, N_{dev} is the total number of devices in the network, and K and T are the two parameters in the model. In the logistic model, the growth curve is symmetric around the *inflection point*, where the growth rate reaches its peak.

A fundamental assumption underlying the logistic model is that individuals should mix homogeneously so that the probability an individual infects another one is exactly the same between any two individuals. This does not hold for Bluetooth worm propagation in a static network because a worm can only infect its neighbors. Hence, the model fitting exercise here is only done on the scenario in Section 5.2 where devices move in a bounded area and are thus mixed throughout the simulation.

We now examine how well the logistic model fits the simulation results. Here, we try a simple model fitting method that still gives us enough insight about how well the model characterizes the worm propagation. We fix the initial point $N(0)$ to be 1 and derive the inflection point from the simulation results. Using these two points, we are able to derive the parameters for the model. In Figure 15, we plot the average number of infected devices as simulation time advances and the logistic curve derived according to the above model fitting method. It is clear that the logistic curve underestimates the propagation speed at the early phase of the worm propagation but overestimates it at the late phase. It is noticed that the logistic model in Equation (2) does not consider any congestion, which can slow down the worm

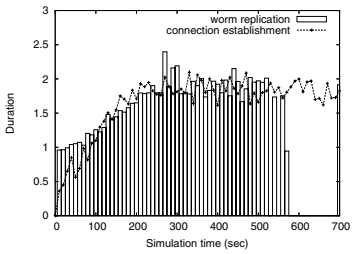


Figure 14. Average duration of connection establishment and worm replication (Mobile, 75m×75m)

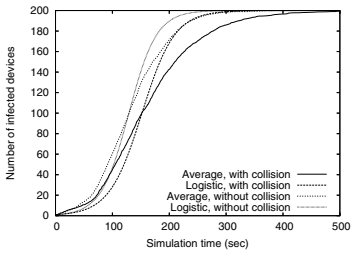


Figure 15. Cumulative number of infected devices over simulation time (Mobile, 75m×75m)

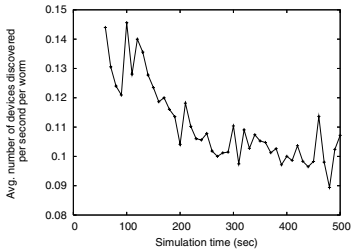


Figure 16. Average number of devices discovered per second per worm as simulation advances

propagation. One may wonder whether incorporating elements that capture packet losses due to co-channel interference in the model may lead to a much better model fitting. We plot the cumulative number of infected devices without co-channel interference and the fitted logistic curve also in Figure 15. The graph shows that although both curves move left relative to those with co-channel interference enabled, the logistic curve still underestimates the propagation speed at the early phase of the worm propagation but overestimates it at the late phase.

The observed mismatch suggests that something is still missing. We mentioned earlier that the logistic function in Equation (2) assumes that devices are sufficiently mixed. It is important, though, to distinguish the two cases: devices are *physically* mixed and devices are *logically* mixed. Two devices which are in each other's communication range can be neighbors physically, but if both of them perform worm activities simultaneously, they may not discover each other and thus they are not neighbors logically. It is obvious that if we do not consider co-channel interference, worm propagation is essentially affected by the degree to which devices are mixed logically instead of physically.

In Figure 16, we plot the average number of devices discovered per second per worm in a sample run (the results from other sample runs are similar and thus omitted here). The co-channel interference in this run is disabled to make explanation easier. It is also worth mentioning that this graph is different from that in Figure 12 because the latter gives the average number of devices discovered in *a single infection cycle*. The average duration of an infection cycle may change in the course of the worm propagation. This is because as more devices are infected in the network, the inquiry phase lasts longer but fewer neighbors are discovered and contacted. Figure 16 reveals that the overall logical contact rate between devices actually decreases as the network is populated with more infected devices. This can slow down the worm propagation at its late phase.

On the other hand, because an infected device performing worm activities may not be discovered by its neighbors, the visibility of uninfected devices to a worm increases when there exist infected devices in the neighborhood. In Figure 17, we compare the fraction of uninfected devices among the whole population and the fraction of uninfected devices among all the devices discovered as the simulation time advances in a sample run. It is clear that relatively more uninfected devices are discovered than their proportion in the whole population. Hence, a model assuming the same discovery ratio for uninfected and infected devices does not fully capture the dynamics of Bluetooth worms.

We also notice that the position of a device on the neighbor list collected by a worm affects the probability that the device is infected. If a device is closer to the front of the list, the worm will try fewer devices before starting to infect it and it is thus more likely that the device still remains in the worm's radio range when the worm tries to infect it. Therefore, devices on the front part of the neighbor lists have higher probability to get infected and vice versa. This is verified from our simulation results. Figure 18 plots the infection probability if a device is uninfected when discovered as opposed to its position on the neighbor list. The decreasing curve confirms our observation.

7 Effects of Model Parameters

The simulation results with the baseline Bluetooth worm model in the previous sections gain us some insights about Bluetooth worm propagation, but it is still unclear how varying model parameters impacts worm spreading. An advanced worm may select its parameters strategically to improve its propagation speed. In the following discussion, we focus on the type of worms whose objective is to minimize the time required to infect the majority (e.g., 95%) of all vulnerable devices.

7.1 Tuning Inquiry Parameters

From both Figures 7 and 12, we know that as more devices are infected, the baseline worm has to spend longer time on collecting its neighbor information and this period approaches the inquiry timeout value (i.e., T_{inq}) when the network is populated with infected devices. The Bluetooth

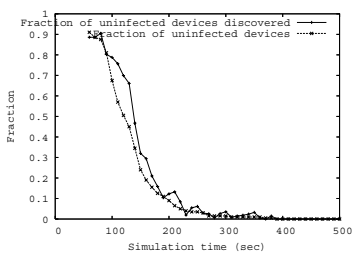


Figure 17. Differences between the fractions of uninfected devices and uninfected devices discovered

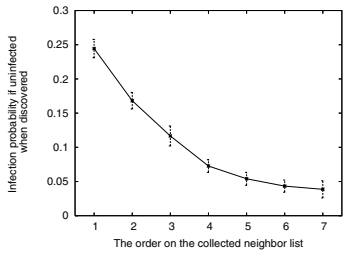


Figure 18. Infection probability vs. their orders on the neighbor lists (95% confidence interval)

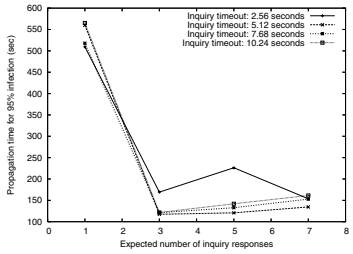


Figure 19. 95% propagation time with different inquiry parameters (Static,75m$\times$75m)

specification states that T_{inq} must be at least 10.24 seconds to discover all devices in an error-free environment. An "intelligent" worm, however, may not follow this suggestion if this does not help improve its propagation speed.

Changing inquiry parameters has mixed effects. If a worm uses a larger inquiry timeout value, then it can possibly discover more nearby devices within one infection cycle. However, if there are not enough devices responding to its inquiry, the inquiry process has to wait until the inquiry timer expires. In such circumstance a smaller inquiry timeout value can shorten the whole inquiry process, but the worm may not receive as many responses as with a larger value. A worm can also change N_{inq}, the expected number of responses. Similarly, a worm using large N_{inq} can potentially discover more devices in its vicinity, but has to wait for the expiration of the inquiry timer if not as many as N_{inq} responses are received within T_{inq} time units.

In a new set of experiments, we aim to find optimal selections on the inquiry parameters for the Bluetooth worm. We vary T_{inq} among 10.24, 7.68, 5.12, and 2.56 seconds and N_{inq} among 7, 5, 3, and 1. The simulation results for the static and mobile networks are illustrated in Figures 19 and 20 respectively. For the static network, varying inquiry timeout value does not impose significant performance difference except the smallest value (i.e., 2.56 seconds). On the other hand, expecting only one inquiry response for each inquiry is always a bad choice for shortening propagation time, irrespective of the inquiry timeout value. When T_{inq} is 5.12 seconds and N_{inq} is 3, the shortest propagation time is needed to infect 95% of all vulnerable devices; it is 117.36 seconds, which is 27.3% shorter than what is used by the baseline worm.

For the mobile network, we notice that if the expected number of inquiry responses is 1 or 3, the propagation time for 95% infection coverage is relatively insensitive to the inquiry timeout value used; otherwise, small inquiry timeout value can help shorten the propagation time. Another observation from Figure 20 is that regardless of inquiry timeout value, smaller N_{inq} always leads to less time required for 95% infection coverage. In other words, expecting only one response for each inquiry in the mobile network is the best choice for the worm. This phenomenon can be explained from two aspects. First, in the mobile network, it is relatively more difficult to obtain inquiry responses from nearby devices because of device mobilities; thus, it is more likely that a worm using large N_{inq} finishes its inquiry process until the inquiry timer expires. Second, although a worm using large N_{inq} can potentially find more devices in its vicinity, it may suffer more unsuccessful attempts at establishing connections to them because of node mobility. Figure 20 also reveals that when N_{inq} is 1 and T_{inq} is 7.68 seconds, the least time is needed for 95% infection coverage; under this configuration, 189.22 seconds is used, which is about 60% of the time used by the baseline worm.

7.2 Tuning Inactive Duration

As seen from Table 1, the period that the baseline worm remains inactive after an infection cycle (i.e., T_{idle}) lasts 20 seconds. As we discuss how a worm accelerates its propagation, a natural strategy for an "aggressive" worm may be to shorten its inactive period. To see how effective this strategy can be, we perform a new set of experiments in which we vary T_{idle} among 1, 10, 20, and 30 seconds. The simulation results are illustrated in Figure 21. We, surprisingly, do not observe that smaller T_{idle} leads to faster worm propagation. Instead, we find that when the inactive period lasts only 1 second, the worm propagation in the static network takes longer time with an enormously large confidence interval; this phenomenon, however, is not observed in the mobile networks. It can be explained as follows. A short inactive period causes severe co-channel interference, which significantly slows down worm propagation in the static network. In the mobile network, device mobility leads to uneven device densities; hence, a healthy device is more likely to move into a area with low interference and thus gets infected there.

Figure 21 also tells us that for the static network, as we increase the inactive duration from 1 to 30 seconds, the time for 95% infection coverage actually decreases, but we be-

lieve that this observation may not hold as we continue increasing T_{idle}. On the other hand, for the mobile network, varying T_{idle} has little impact on worm propagation speed. These results reveal that a Bluetooth worm that aggressively reduces its inactive duration may not improve its propagation speed at all, and on the contrary, this can significantly slow down its propagation if the network is relatively static.

8 Exploiting Out-of-Band Propagation

Bluetooth interfaces often coexist with other communication channels on the same devices. For example, a smart cell phone can have access to a cellular network such as GSM/CDMA and UMTS, and it also provides various communication interfaces such as infrared, Bluetooth, 802.11, and GPRS/CDMA1X [10]. A Bluetooth worm can exploit out-of-band channels to accelerate its propagation. Actually, such worms have already appeared on cell phones. Both Mabir and CommWarrior can spread not only through Bluetooth interface, but also by using MMS, which is based on GPRS (General Packet Radio Service) technology.

As a Bluetooth worm uses proximity-based infection, the victims that can be infected are limited to those in its radio range. If out-of-band propagation channels are available, a worm can replicate itself onto devices that are multiple hops away or even those devices that can never be reached via mere Bluetooth communication. An intelligent Bluetooth worm can thus exploit out-of-band communication channels to expand its infection range.

In this part, we investigate how a Bluetooth worm can exploit MMS communication to speed up its propagation. We assume that all devices are equipped with both Bluetooth and MMS capabilities. We also assume that MMS messages are coded by the GPRS 3+2 scheme [1]. The download and upload speeds are 60.0kbps and 40kbps respectively. We simulate the same networks in Sections 5.1 and 5.2, but each device has a list of "buddies", which are devices it often sends MMS messages to. For simplification, we do not model the social network that yields these buddy lists. Instead, we assume that each device randomly picks k other devices as its buddies, where k is a configurable parameter. The baseline worm behavior is slightly changed as follows. After a device gets infected, it first checks its buddy list, and uploads a copy of worm code to each one on the list. Once this is finished, it starts its Bluetooth infection cycle shown in Figure 1. We only consider uploading and downloading time in the worm code transfer.

We vary the number of buddies that each device has in the experiments. The simulation results are presented in Figure 22. We find that one buddy per device is sufficient in reducing worm propagation time by half in the static network, and increasing that to two buddies per device only marginally improves the propagation speed. By contrast, in the mobile network, there is more noticeable difference between one buddy per device and two buddies per device: under the former setting, out-of-band propagation helps reduce the time for 95% infection coverage by 35.5%, but under the latter setting, this time is decreased by 53%.

9 Defense Implications

As Bluetooth worms employ more sophisticated strategies that accelerate their propagation, it makes it increasingly difficult to defend against them through manual human intervention. Active defense schemes such as patching counter-worms, which have been proposed to fight Internet worms, may not be effective in fighting aggressive and malicious Bluetooth worms. The reasons are manifold. First, using active worms to stop malicious worms requires devices to carry on frequent Bluetooth activities, which can drain their power rapidly. Second, intense worm behavior can lead to severe congestion in the network. High packet loss rates will severely impact normal communications.

Increasing software diversity can be considered as a possible approach to slowing down Bluetooth worm propagation. As observed in Figure 3, the total propagation time increases almost super-linearly with increasing fraction of insusceptible devices, especially when device density is relatively low. This suggests that if we reduce the number of vulnerable devices by implementing diverse but functionally equivalent software on Bluetooth devices, we can slow down worm propagation.

Another defense scheme that can be used against Bluetooth worms is the quarantine defense. For example, if Bluetooth worms are found on cell phones located at a specific area, quarantine tools can be deployed there to prevent them from spreading to other places. Bluetooth by principle is a proximity-based protocol, suggesting that quarantine defense can be an effective counter-measure against worms using it as a single infection means. However, as discussed in Section 8, Bluetooth often coexists with other communication channels. Such out-of-band communication capabilities overcome the spatial obstacles to Bluetooth worm propagation. It is, therefore, much harder to quarantine devices infected by a worm like CommWarrior that uses multiple communication mechanisms to spread itself.

10 Related Work

Although substantial efforts have been made in analyzing, modeling and simulating Internet worms, worm propagation in mobile networks has only been investigated in a few papers. Mickens *et al.* [15] also observed that traditional epidemic models fail to characterize worm propagation in mobile networks. In addition, they claim that a worm is easier to spread in a more dynamic network. Our results with a detailed MAC/PHY layer protocol reveal that this may not be true. Although in a highly dynamic network

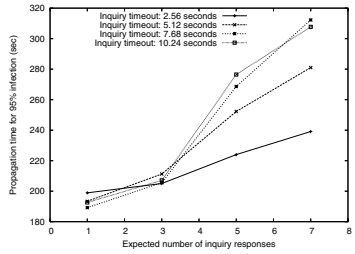


Figure 20. 95% propagation time with different inquiry parameters (Mobile, 75m×75m)

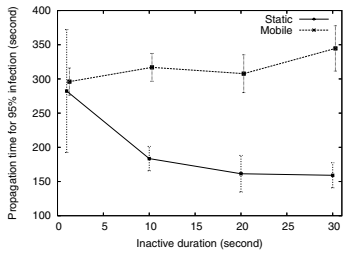


Figure 21. 95% propagation time with different inactive periods (95% confidence interval)

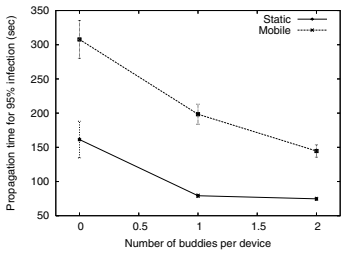


Figure 22. 95% propgation time under different buddies per device (95% confidence interval)

infected devices mix better with healthy ones, higher mobility also lead to more unsuccessful interactions between devices. Khayam *et al.* [12] developed a topologically-aware worm propagation model for stationary wireless sensor networks. They incorporate MAC layer interference into this model by specifying a constant infection rate when a worm spreads itself onto its neighbors. This differs from our work because we use detailed MAC/PHY protocol interactions to decide whether an infection attempt succeeds. Anderson *et al.* [5] simulated mobile contagion using mobility traces collected from a campus wireless network. Their work considers worm propagation in a different type of mobile networks from the one we investigated.

11 Summary

Recent occurrences of Bluetooth worms have created growing security concerns over the data stored on mobile devices like cell phones and PDAs. This paper investigates the nature and dynamics of Bluetooth worm propagation. We find that Bluetooth worm propagation speed can be negatively impacted by the device mobility and the channel congestion. We also observe that an advanced worm can improve its propagation speed by strategically selecting its parameters or exploiting extra long-distance communication capabilities.

References

[1] General packet radio service. http://en.wikipedia.org/wiki/GPRS.

[2] The McAfee AVERT Virus Information Library. http://vil.nai.com/vil/.

[3] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/index.html.

[4] UCBT - Bluetooth Extention for NS2 at the University of Cincinnati. http://www.ececs.uc.edu/ cdmc/ucbt/ucbt.html.

[5] E. Anderson, K. Eustice, S. Markstrum, M. Hansen, and P. Reiher. Mobile contagion: Simulation of infection and defense. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, June 2005.

[6] E. Chien. Security response: Symbos.mabir. Symantec Corporation, 2005.

[7] W. Enck, P. Traynor, P. McDaniel, and T. F. La Porta. Exploiting open functionality in sms-capable cellular networks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, November 2005.

[8] P. Ferrie and F. Perriot. Paradise lost. Virus Bulletin, http://www.virusbtn.com/, April 2005.

[9] P. Ferrie, P. Szor, R. Stanev, and R. Mouritzen. Security responses: Symbos.cabir. Symantec Corporation, 2004.

[10] C. Guo, H. J. Wang, and W. Zhu. Smart-phone attacks and defenses. In *Proceedings of HotNets III*, November 2004.

[11] J. P. Lynch Jr. Co-channel interference in bluetooth piconets. Master's thesis, Virginia Polytechnic Institute and State University, 2002.

[12] S. A. Khayam and H. Radha. A topologically-aware worm propagation model for wireless sensor networks. In *Proceedings of The 2nd International Workshop on Security in Distributed Computing Systems (SDCS-2005)*, 2005.

[13] M. Lactaotao. Security information: Virus encyclopedia: Symbos_comwar.a: Technical details. Trend Micro Incorporated, 2005.

[14] N. Leavitt. Mobile phones: The next frontier for hackers. *Computer*, April 2005.

[15] J. W. Mickens and B. D. Noble. Modeling epidemic spreading in mobile environments. In *Proceedings of the 4th ACM workshop on Wireless security*, September 2005.

[16] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security & Privacy*, 1(4), 2003.

[17] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium (Security '02)*, August 2002.

[18] C. Taylor and N. Mawston. Bluetooth market doubles: CSR still gaining momentum. http://www.strategyanalytics.net/, December 2005.

[19] J. Tourrilhes. On-demand bluetooth: Experience integrating bluetooth in connection diversity. Technical Report HPL-2003-178, Hewlett-Packard Labs, August 2003.