Delay-Tolerant Botnets

Zesheng Chen\textsuperscript{1}, Chao Chen\textsuperscript{2}, and Qian Wang\textsuperscript{1}

\textsuperscript{1}Department of Electrical & Computer Engineering
Florida International University
Miami, FL 33174
E-mails: \{zchen, qian.wang\}@fiu.edu

\textsuperscript{2}Department of Engineering
Indiana University - Purdue University Fort Wayne
Fort Wayne, IN 46805
E-mail: chen@engr.ipfw.edu

Abstract—Botnets have become one of top threats to the Internet. Many detection methods have been developed to distinguish botnet behaviors from normal human behaviors. Future botnets, however, may incorporate the characteristics of human beings and weaken the existing detection techniques. In this work, we study an intelligent botnet, called the delay-tolerant botnet, that intentionally adds random delays to the command propagation and endeavors to avoid the detection. We then apply mathematical analysis to derive the average delay required to distribute a command to all bots in three types of command and control architectures: centralized, distributed, and hybrid delay-tolerant botnets. We find that in all cases, the delay increases approximately logarithmically with the number of bots, indicating that the delay-tolerant botnets are scalable. Finally, we verify the analytical results by simulations.

I. INTRODUCTION

A botnet is a network of compromised computers that are organized by a malicious attacker called the botmaster. Botnets have been exploited to set off denial-of-service attacks, send spam emails, and search for confidential information. For example, the Storm botnet affected tens of millions of hosts and was used to send spam emails and distributed denial-of-service (DoS) attacks in 2007 [11]. Therefore, botnets have become one of top threats to the current Internet.

A botmaster uses command and control (C&C) channels to deliver commands to all bots. Many methods have been developed to detect such C&C channels [8], [9], [10], [13], [21]. For example, Gu \textit{et al.} explored the spatial-temporal correlation and similarity among bots to detect botnet traffic [10]. Zhang and Paxson applied a timing-based algorithm to detect stepping stones [21]. Livadas \textit{et al.} used machine learning techniques to identify IRC (Internet Relay Chat) based botnets [13]. These methods attempt to distinguish botnet behaviors from normal human behaviors. Future botnets, however, may incorporate the characteristics of human beings, invalidating or weakening the current detection systems.

In this work, we study an intelligent botnet, which is called the delay-tolerant botnet and is named after the delay-tolerant network [6], [7]. Delay-tolerant networks are proposed to support applications that do not demand the immediate transmission of packets from the source to the destination. Similarly, many commands in botnets are indeed not required to be simultaneously delivered to all bots, such as searching for confidential information and sending spam emails. Even for DoS attacks, if the botmaster can accurately predict when a command can be received by all bots, the timing of attacks can be set, and the command is still delay tolerant. A simple way to introduce delays into botnets is that each bot retrieves a command from the server with a random inter-query delay or holds a command for random delays before forwarding it to other bots. In this way, different bots can obtain the command at different time instants, which is more similar to human behaviors and is more difficult to detect.

Delivering a command to all bots is analogous to multicasting a packet to all receivers [17] or transmitting a file to all peers in peer-to-peer networks [12]. For multicast or peer-to-peer networks, an important metric is how long it takes to deliver a packet or a file to all receivers or peers. Similarly, in this work we study the average delay for a command to be distributed to all bots in the delay-tolerant botnets, which is called the botnet delay. Then, it is crucial to answer the following question: Are delay-tolerant botnets scalable? That is, can a delay-tolerant botnet support a large number of bots? Generally, if the botnet delay increases linearly or exponentially with the number of bots, the botnet is not scalable. Otherwise, if the botnet delay increases logarithmically with the number of bots, the system is scalable.

The goal of this work is to better understand the potential threats of future botnets. Our research work makes several contributions as follows:

- We study the botnet delay in three C&C architectures: centralized, distributed, and hybrid delay-tolerant botnets. Through mathematical analysis, we find that in all three architectures, the botnet delay increases approximately logarithmically with the number of bots. We also use simulations to verify these results. Therefore, the designed delay-tolerant botnets are scalable.
- We discover that in centralized delay-tolerant botnets, if a bot retrieves a command from the server with an exponential inter-query delay with mean \(1/\lambda\), the botnet delay is \(H_N/\lambda\), where \(N\) is the number of bots and \(H_N\) is the \(N\)-th harmonic number \([4]\). Moreover, when considering an arbitrary distribution of the bot inter-query delay, we can apply an exponential distribution as a performance lower bound in delay-tolerant botnets.
- We find that in distributed delay-tolerant botnets or in the core overlay networks of hybrid delay-tolerant botnets, if the topology is an Erdos-Renyi random graph with an average nodal degree of \((k)\) and a bot delivers a command to its neighbors with an exponential delay with mean \(1/\lambda\), the botnet delay is \(\ln(N)/(2\lambda \ln((k)))\), where \(N\) is the number of bots.
number of bots in a distributed delay-tolerant botnet or
servent bots in the core overlay network of a hybrid delay-
tolerant botnet.

The remainder of this paper is structured as follows. Sec-
tion II describes three architectures of delay-tolerant botnets.
Section III derives the botnet delay. Section IV further uses
simulations to verify analytical results. Finally, Section V
concludes this paper.

II. DELAY-TOLERANT BOTNET ARCHITECTURES

In this section, we provide the background on the existing
botnet C&C architectures and introduce random delays to
these systems to design delay-tolerant botnets (DTBs). In a
botnet, the botmaster has to propagate a command to all bots.
As shown in Figure 1, C&C architectures can be categorized
into three types:

- **Centralized:** The botmaster issues a command to a C&C
  server, and all bots are connected to the server to obtain the
  command, as illustrated in Figure 1(a). There are two styles of
  centralized botnets: “push” and “pull” [10]. Push-style botnets
  (e.g., IRC-based C&C) deliver commands from the server to
  bots in the channel, whereas pull-style botnets (e.g., HTTP-
  based C&C) require bots to connect back to the server for
  commands. In our designed DTBs, we choose the pull style,
  since the push style introduces synchronization among bots,
  which is easy to detect. Specifically, we consider that a bot
  connects to the server with an inter-query delay. In the existing
  pull-style botnets, the inter-query delay is a fixed value and
  follows a repeating and regular pattern. Instead, we design the
  inter-query delay to follow a random distribution such as the
  exponential distribution. In this way, a bot does not follow a
  repeating and regular pattern to connect to a server. Moreover,
  different bots connect to the server at different time instants,
  which is more similar to human behaviors and is more difficult
to detect.

- **Distributed:** The botnet is constructed as an overlay
  network and behaves similarly to a peer-to-peer system, as
  illustrated in Figure 1(b). Once a bot receives a command,
  it will forward the command to its neighbors in the overlay
  network [18]. The topology of distributed botnets can be an
  Erdos-Renyi random graph or a power-law topology [5]. In
  our designed DTBs, if a bot receives a command, it will
  hold the command for random delays before forwarding it
  to its neighbors. Moreover, the bot delivers the command to
  different neighbors at different time instants. Here, the delays
  are random variables and follow probability distributions such
  as the exponential distribution.

- **Hybrid:** The botnet incorporates the advantages of both
  centralized and distributed botnets, and is proposed in [20].
  As illustrated in Figure 1(c), some bots in the core of the
  botnet, called servent bots, are constructed like the distributed
  botnet. Other bots, called client bots, connect to one of servent
  bots and behave similarly to the centralized botnet. Therefore,
  the hybrid botnet can be regarded as a combination of the
  centralized botnet and the distributed botnet, and our designed
  random delays can similarly be applied to the hybrid botnet.

III. MATHEMATICAL ANALYSIS

In this section, we study the scalability of DTBs through
mathematical analysis. Specifically, we derive the average time
for a botmaster to distribute a command to all bots in a
DTB, which is called the botnet delay and denoted by E(T). We
analyze the botnet delays in the three different DTB
architectures.

A. Centralized DTBs

For the centralized DTB as illustrated by Figure 1(a), if a
botmaster releases a command to the server at time 0, bot i
will retrieve the command at time $D_i$. Here, $D_1, D_2, \ldots, D_N$ are independent and identically distributed (i.i.d.) with
distribution function $F_D(\cdot)$, where $N$ is the number of bots.
Hence, the time interval for all bots to retrieve the command is

$$T = \max_i D_i.$$ (1)

The distribution function of $T$ follows

$$F_T(t) = \Pr(T \leq t) = \Pr(\max_i D_i \leq t)$$ (2)

$$= \Pr(D_1 \leq t, D_2 \leq t, \ldots, D_N \leq t)$$ (3)

$$= \prod_{i=1}^{N} \Pr(D_i \leq t)$$ (4)

$$= (F_D(t))^N.$$ (5)
If a continuous-time system is considered, the botnet delay is

\[
E(T) = \int_0^{+\infty} tf_T(t)dt = \int_0^{+\infty} \Pr(T > t)dt \tag{6}
\]

\[
= \int_0^{+\infty} \left[ 1 - (F_D(t))^N \right] dt. \tag{7}
\]

1) Exponential Distribution of the Inter-Query Delay: Random variable \( D \) may follow different probability distributions. If the distribution of \( D \) is assumed to be exponential with mean \( 1/\lambda \), i.e.,

\[
f_D(t) = \left\{ \begin{array}{ll} \lambda e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0, \end{array} \right. \tag{8}
\]

then

\[
F_D(t) = 1 - e^{-\lambda t}, \quad t \geq 0 \tag{9}
\]

\[
F_T(t) = (1 - e^{-\lambda t})^N, \quad t \geq 0. \tag{10}
\]

Note that \( D_i \) is the actual time interval between the command arriving at the server and \( i \)th retrieving the command from the server, which is different from the inter-query delay of bot \( i \). Because of the memoryless property of the exponential distribution, if bot \( i \) uses an exponential inter-query delay for the command at the server, \( D_i \) is also exponential with the same mean. Thus, we can derive an explicit expression for the botnet delay

\[
E(T) = \int_0^{+\infty} \left[ 1 - (1 - e^{-\lambda t})^N \right] dt \tag{11}
\]

\[
= \int_0^{+\infty} \left[ 1 - \sum_{i=0}^{N} \binom{N}{i} (-1)^i e^{-i\lambda t} \right] dt \tag{12}
\]

\[
= \sum_{i=1}^{N} \binom{N}{i} (-1)^{i+1} \int_0^{+\infty} e^{-i\lambda t} dt \tag{13}
\]

\[
= \sum_{i=1}^{N} \binom{N}{i} \frac{(-1)^{i+1}}{i\lambda}, \tag{14}
\]

where in Equation (12) we apply the binomial theorem [2]. We set \( H_n = \sum_{i=1}^{n} \binom{n}{i} \frac{(-1)^{i+1}}{i} \). Then, \( H_1 = 1 \), and when \( n \geq 1 \),

\[
H_{n+1} - H_n = \sum_{i=1}^{n+1} \binom{n+1}{i} \frac{(-1)^{i+1}}{i} - \sum_{i=1}^{n} \binom{n}{i} \frac{(-1)^{i+1}}{i} \tag{15}
\]

\[
= \frac{(-1)^{n+2}}{n+1} + \sum_{i=1}^{n} \left[ \binom{n+1}{i} - \binom{n}{i} \right] \frac{(-1)^{i+1}}{i} \tag{16}
\]

\[
= \frac{(-1)^{n+2}}{n+1} + \sum_{i=1}^{n} \binom{n}{i-1} \frac{(-1)^{i+1}}{i} \tag{17}
\]

\[
= \frac{(-1)^{n+2}}{n+1} + \frac{1}{n+1} \sum_{i=1}^{n} \binom{n+1}{i} \frac{(-1)^{i+1}}{i} \tag{18}
\]

\[
= \frac{1}{n+1} \sum_{i=0}^{n+1} \binom{n+1}{i} \frac{(-1)^{i+1} + 1}{i} \tag{19}
\]

\[
= \frac{1}{n+1}, \tag{20}
\]

where we apply the binomial theorem in the last equation. The above derivation leads to \( H_n = \sum_{i=1}^{n} \frac{1}{i} \). That is, \( H_n \) is the \( n \)-th harmonic number [4]. Hence, we have the following theorem.

**Theorem 1**: If a bot retrieves a command from the server with an exponential inter-query delay with mean \( 1/\lambda \), the botnet delay is

\[
E(T) = \frac{H_n}{\lambda} = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{1}{i}. \tag{21}
\]

We can make two interesting observations from Theorem 1. First, the botnet delay is proportional to the mean of \( D \) (i.e., \( 1/\lambda \)). Second, since \( H_n \) is \( O(1 + \ln(N)) \) [4], \( E(T) \) is \( O(1 + \ln(N)/\lambda) \). Therefore, \( E(T) \) increases with the rate of \( \ln(N) \) when \( N \) increases, which indicates that such a simple randomized system is scalable.

2) General Distribution of the Inter-Query Delay: For a general distribution of the inter-query delay (denoted by \( Q \)), however, the command can be released to the server at any arbitrary time, and thus \( D \) usually does not have the same distribution as \( Q \). To calculate the distribution of \( D \), let \( F_Q(t) \) be the cumulative distribution function (CDF) of the bot inter-query delay. According to the renewal theorem [15], the CDF of \( D \) can be derived from the CDF of \( Q \):

\[
F_D(t) = \frac{\int_0^t (1 - F_Q(x))dx}{\int_0^{+\infty} (1 - F_Q(x))dx}. \tag{22}
\]

Similar to the argument in [19], we can define the hazard function \( \lambda(t) \) that denotes the intensity of queries from a bot made at time \( t \), given that no query from this bot is made during \( [0, t) \). Then,

\[
\lambda(t) = \frac{d}{dt} F_D(t). \tag{23}
\]

For an arbitrary hazard function \( \lambda(t) \), if \( \lambda(t) \geq \lambda \) for all \( t \geq 0 \) and \( \lambda > 0 \), then the DTB with the inter-query delay having CDF \( F_Q(\cdot) \) would perform at least as well as the DTB with the exponential inter-query delay with mean \( 1/\lambda \).

To demonstrate this, we show an example when \( Q \) follows a uniform distribution with mean \( 1/\lambda \), i.e.,

\[
f_Q(t) = \begin{cases} \frac{1}{2}, & 0 \leq t < \frac{1}{\lambda} \\ 0, & \text{otherwise.} \end{cases} \tag{24}
\]

The CDFs of \( Q \) and \( D \) are

\[
F_Q(t) = \frac{\lambda}{2} t, \quad 0 \leq t < \frac{2}{\lambda} \tag{25}
\]

\[
F_D(t) = \left( 1 - \frac{\lambda}{4} t \right) \lambda t, \quad 0 \leq t < \frac{2}{\lambda}. \tag{26}
\]

Thus, the hazard function \( \lambda(t) \) can be obtained

\[
\lambda(t) = \frac{\lambda}{1 - \frac{\lambda}{2} t}, \quad 0 \leq t < \frac{2}{\lambda}. \tag{27}
\]

and \( \lambda(t) \) is undefined for \( t \geq 2/\lambda \). It is noted that \( \lambda(0) = \lambda \) and \( \lambda(t) > \lambda \) for all \( t \in (0, 2/\lambda) \). Therefore, compared with
the exponential distribution with mean $1/\lambda$ for the bot inter-query delay, the command can be delivered to all bots faster. Indeed, from Equation (7),

$$E(T) = \int_0^{2/\lambda} \left\{ 1 - \left[ \left( 1 - \frac{\lambda}{4} t \right) \lambda \right]^N \right\} dt < \frac{2}{\lambda}. \quad (28)$$

The upper bound of $E(T)$ is $O(1 + 1/\lambda)$ and independent of the number of bots.

Therefore, we can use Equations (22) and (7) to calculate the botnet delay from an arbitrary distribution of the bot inter-query delay. Moreover, when considering an arbitrary distribution of the bot inter-query delay, we can apply an exponential distribution as a performance lower bound in DTBs, which simplifies the dynamics significantly.

### B. Distributed DTBs

As illustrated by Figure 1(b), a command is released to one bot and is then forwarded to all other bots in the distributed DTB. Assume that bot $B_1$ is the first bot that obtains the command and bot $B_{P+1}$ is the last bot that receives the command. Specifically, the command is delivered from bot $B_1$ through bots $B_2, B_3, \cdots, B_{P}$ to reach bot $B_{P+1}$. Here, $P$ is the path length between bots $B_1$ and $B_{P+1}$. We also assume that bot $B_i$ ($i = 1, 2, \cdots, P$) holds a command for a random delay of $D_i$ before forwarding it to bot $B_{i+1}$. Thus, the time interval for all bots to receive the command is

$$T = \sum_{i=1}^{P} D_i. \quad (29)$$

Similar to centralized DTBs, we assume that $D_i$'s are i.i.d. with the exponential distribution with mean $1/\lambda$. Then, given the value of random variable $P, T$ follows a gamma distribution with parameters $P$ and $\lambda$, i.e.,

$$f_T(t|P) = \lambda e^{-\lambda t} \left( \frac{\lambda t}{P-1} \right)^{P-1}, \quad t \geq 0, \quad (30)$$

which leads to the botnet delay

$$E(T) = \frac{1}{\lambda} E(P). \quad (31)$$

It has been studied that the average path length (i.e., $E(P)$) increases approximately logarithmically with $N$ in Erdos-Renyi random graphs and power-law topologies [1], [14]. That is, for most botnet topologies in which we are interested [5], $E(P)$ is $O(1 + \ln(N))$. Therefore, $E(T)$ is $O(1 + \ln(N)/\lambda)$, indicating that distributed DTBs are scalable.

Specifically, we further study DTBs with Erdos-Renyi random graphs. According to [1],

$$E(P) = \frac{1}{2} \ln\left( \frac{N}{\ln(\langle k \rangle)} \right), \quad (32)$$

where $\langle k \rangle$ is the average nodal degree in a graph. Hence, we have the following theorem.

**Theorem 2:** If a bot delivers a command to its neighbors with an exponential delay with mean $1/\lambda$ in a distributed DTB with an Erdos-Renyi random graph with the average nodal degree of $\langle k \rangle$, the botnet delay is

$$E(T) = \frac{\ln(N)}{2\lambda \ln(\langle k \rangle)}. \quad (33)$$

Note that the above analysis assumes the spatial independence of message delivery among different paths in a network. It has been shown that if arrival rate $\lambda$ is very small, the spatial dependence can affect information dissemination significantly [3]. However, if arrival rate $\lambda$ is not too small, which is the case for DTBs, we can ignore the spatial dependence to reduce computational complexity, without losing much accuracy.

### C. Hybrid DTBs

In the hybrid DTB as illustrated by Figure 1(c), a command is first released to one of servent bots and is then forwarded to other servent bots in the overlay network. A client bot connects back to a servent bot for the command. Assume that there are $N_S$ servent bots and $N_C$ client bots, where $N_S + N_C = N$. The command is released to servent bot $B_1$ and is delivered along the path $B_2, B_3, \cdots, B_P$ to reach $B_{P+1}$ that is the last servent bot receiving the command. Servent bot $B_{i}$ ($i = 1, 2, \cdots, P$) holds a command for a random delay of $D_i^S$ before forwarding it to servent bot $B_{i+1}$. We also assume that when servent bot $B_{P+1}$ receives the command, there are still $L (0 \leq L \leq N_C)$ client bots that have not received the command from servent bots. The set of these bots is denoted by $J$. Let $D_j^C (j \in J)$ denote the time interval between the command arriving the client bot $j$'s servent bot and client bot $j$ retrieving the command. Then, the time interval for all bots to receive the command is

$$T = \sum_{i=1}^{P} D_i^S + \max_{j \in J} D_j^C, \quad (34)$$

which consists of the components from both centralized and distributed DTBs. If $D_i^S$'s and $D_j^C$'s are i.i.d. with the exponential distribution with mean $1/\lambda$, the botnet delay is

$$E(T) = \frac{1}{\lambda} (E(P) + H_L). \quad (35)$$

Since $E(P)$ is $O(1 + \ln(N_S))$ and $H_L$ is $O(1 + \ln(L))$, $E(T)$ is $O(1 + (\ln(N_S) + \ln(L))/\lambda)$. If $N_C \gg N_S$, i.e., the hybrid DTBs has a relatively small core overlay network, all servent bots will receive the command before the majority of client bots, i.e., $L \approx N_C$. In this case, $E(T)$ is about $O(1 + \ln(N_S \cdot N_C)/\lambda)$. Therefore, hybrid DTBs are still scalable in terms of the botnet delay.

Moreover, if the core overlay network is an Erdos-Renyi random graph, we have the following theorem.

**Theorem 3:** If a servent bot delivers a command to its neighbors or a client bot retrieves a command from its servent bot with an exponential delay with mean $1/\lambda$ in a hybrid DTB with an Erdos-Renyi random graph as the core overlay network, the botnet delay is

$$E(T) = \frac{1}{\lambda} \left( \frac{\ln(N_S)}{2\ln(\langle k \rangle)} + \sum_{i=1}^{L} \frac{1}{i} \right), \quad (36)$$
Figure 3 shows the botnet delays in centralized DTBs when inter-query delays, i.e., can obtain the botnet delay with the geometric distribution of $\lambda < 1$ assuming $\Pr$ distribution with mean $1/\lambda$. Equation (8)), a uniform distribution is uniform, the botnet delay is always less than 2/$\lambda$. Moreover, compared with the exponential distribution, the geometric distribution spreads the command slightly faster. These results verify our analysis in Section III-A.

A. Centralized DTBs

To evaluate the scalability of centralized DTBs, we vary the number of bots (i.e., $N$) from 1,000 to 50,000. Figure 2 compares the simulation results with the analytical results in Theorem 1 for $E(T)$ in centralized DTBs. Here, we assume that the inter-query delays follow the exponential distribution with a mean of 20 mins or 10 mins (i.e., $\lambda = 0.05 / \text{min}$ or $\lambda = 0.1 / \text{min}$). Simulation results are averaged over 1,000 runs. It can be seen that the simulation results are (almost) identical to the analytical results, especially for large $N$. Therefore, this verifies that Theorem 1 holds and demonstrates that centralized DTBs are scalable. For example, even in a centralized DTB with 50,000 bots, the botnet delay is less than 4 hours in the case of $\lambda = 0.05 / \text{min}$.

Next, we study the effect of inter-query delay distributions on centralized DTBs. Specifically, we consider three probability distributions of inter-query delays: an exponential distribution with mean $1/\lambda$ (i.e., Equation (8)), a uniform distribution over $[0, 2/\lambda]$ (i.e., Equation (24)), and a geometric distribution with mean $1/\lambda$. For the geometric distribution, $\Pr(Q = i) = \Pr(D = i) = \lambda(1 - \lambda)^{i-1}$, $i = 1, 2, \ldots$, assuming $\lambda < 1$. Following the steps in Section III-A, we can obtain the botnet delay with the geometric distribution of inter-query delays, i.e., $E(T) = \sum_{i=0}^{+\infty} \{1 - [1 - (1 - \lambda)^i]N\}$. Figure 3 shows the botnet delays in centralized DTBs when the inter-query delays follow the exponential, uniform, or geometric distribution. Since the simulation results are identical to the analytical results, Figure 3 only gives the simulation results over 1,000 runs. It can be seen that when the delay distribution is uniform, the botnet delay is always less than 2/$\lambda$. Moreover, compared with the exponential distribution, the geometric distribution spreads the command slightly faster. These results verify our analysis in Section III-A.

B. Distributed DTBs

We then study the scalability of distributed DTBs through simulations. Specifically, we simulate the command delivery over Erdos-Renyi random graphs with the average nodal degree of 20. A bot holds a command for a random delay that follows the geometric distribution with mean $1/\lambda$. Figure 4 shows the simulation results when $N$ varies from 1,000 to 38,000 and $\lambda$ equals 0.05 or 0.1. The curve is the average over 1,000 runs, whereas the error bar represents the standard deviation. Figure 4 also shows the analytical results. Since the geometric distribution can be regarded as the discrete-time counterpart of the exponential distribution, the analytical result
bots construct an Erdos-Renyi random graph. The x-axis uses a log scale of Theorem 2 is applied. It can be seen that when \( \lambda = 0.1 \), the analytical results almost overlap with the simulation results. When \( \lambda = 0.05 \), the analytical results slightly over-estimate the botnet delays because the spatial dependence over different paths is ignored. The analytical results, however, characterize the tendency of the botnet delay accurately when \( N \) varies from small to large. That is, the simulation results verify that distributed DTBs are scalable.

### C. Hybrid DTBs

Finally, we consider the scalability of hybrid DTBs through simulations. Specifically, we simulate \( N_S \) servant bots that construct an Erdos-Renyi random graph with the average nodal degree of 15. A servant bot holds a command for a random delay that follows the geometric distribution with a mean of 10 mins. A client bot has an exponential inter-query delay with a mean of 10 mins. We consider two cases for assigning client bots to servant bots: the uniform assignment where each servant bot has \( N_C/N_S \) client bots and the extreme assignment where all client bots are assigned to servant bot \( B_{P+1} \). Figure 5 shows the simulation results when \( N \) varies from 1,000 to 40,000 and \( N_S/N \) equals 0.2 or 0.05. Note that the x-axis uses a log scale. As expected, the botnet delay is larger for the case of the extreme assignment. However, both \( N_S/N \) and the assignment scheme do not affect the botnet delay significantly. Figure 5 also shows the analytical results by applying Theorem 3 and assuming \( L = N_C \). It can be seen that our mathematical analysis predicts the tendency of the botnet delay accurately when \( N \) varies from small to large. Therefore, the simulation results verify that hybrid DTBs are scalable.

### V. Conclusions

In this work, we attempt to better understand the potential threats of future botnets. Specifically, we have studied DTBs that introduce random delays to command propagation, making botnet behaviors more similar to human behaviors and bots more difficult to detect. We have designed three types of DTBs: centralized, distributed, and hybrid. Through mathematical analysis, we have found that in all cases, the botnet delay increases approximately logarithmically with the number of bots. We have also used simulations to verify that the designed DTBs are indeed scalable.

As part of our on-going work, we plan to develop effective detection and defense mechanisms against DTBs.

### References


