Botnets for scalable management

Jérôme François, Radu State, Olivier Festor

Botnets for scalable management

Jérôme François, Radu State, and Olivier Festor

MADYNES - INRIA Lorraine, CNRS, Nancy-Université, France
{jerome.francois,radu.state,olivier.festor}@loria.fr

Abstract. With an increasing number of devices that must be managed, the scalability of network and service management is a real challenge. A similar challenge seems to be solved by botnets which are the major security threats in today's Internet where a botmaster can control several thousands of computers around the world. This is done although many hindernesses like firewalls, intrusion detection systems and other deployed security appliances to protect current networks. From a technical point of view, such an efficiency can be a benefit for network and service management. This paper describes a new management middleware based on botnets, evaluates its performances and shows its potential impact based on a parametric analytical model.

1 Introduction

Network and service management is an important component to assure the well functioning of a network. It is divided into five domains: fault management, configuration, accounting tasks, performance and security monitoring. However network management planes face several problems to be scalable. Authors of malware (bots, worms) already faced these challenges and some of their achievements are very surprising. There are cases, where one botmaster can control up to 400 000 bots [1]. It is thus natural to investigate if it is possible to use a botnet to perform management operations on a large scale infrastructure. This approach is somehow a time travel, since long time ago, among the first IRC (Internet Relay Chat) [2] bots, Eggdrop [3] was created not for hackers but for helping administrator of IRC networks. The main contribution of this paper is to propose a management plane based on a botnet model, evaluate its performance and show its feasibility. Our paper is structured as follows. In section 2, we introduce the malware communication system and its possible adaption for managing networks. In section 3, the mathematical model and the associated metrics are explained in details. The next section 4 highlights our first experimental results. Related works are presented in section 5. Finally, we conclude the paper and outline future works.

2 Malware-based management architecture

2.1 Classical management architecture and challenges

Network management solutions show their limits today due to several reasons. First of all, there are more and more hosts to be managed and the management

domains have no well delimited boundaries. The management domain is split on several sites and a lot of tasks are usually delegated to other companies which need to access to the network. Moreover, a management operation could be performed on different locations in different countries and has to pass through a lot of active equipments like firewalls or network address translators (not only under the responsibility of the company). For a comprehensive overview, please refer to [4]. The main challenges that we address are related to scalability.

2.2 Internet worm and malware communication paradigms

A worm primary goal is to infect multiple machines without being detected or countered. To reach this goal, the worm can exploit security holes and there are various ways to improve the infection rate. In [5], some existing mechanisms are listed. Malware contain generally malicious payload. The most dangerous malware are stealthy and are able to retrieve private information (password, credit card number...) or to get the control of a system in order to use it as a proxy for future malicious activities (spamming, distributed denial of service attacks, beginning a worm infection, password cracking...)

This kind of malware is based on a control mechanism as in figure 1. Once the bot software is installed on a computer, the bot connects itself to the botnet. This technique is able to bypass most of firewalls and network address translators related problems, since outgoing connections are used. If a firewall blocks outgoing traffic too, it should allow some traffic like web traffic. Thus the IRC server can use different ports to bypass this kind of firewalls.

2.3 Malware based management framework

We consider that malware communication scheme can be a reliable middleware solution for network and service management [4]. Firstly, the exchange of commands is simple and multiple operations are possible. Moreover, the decentralized communication topology of these networks allows to manage many bots. In [1] some statistics about botnets show that controlling 400 000 bots is possible contrary to the current management framework. In [6], the authors model and evaluate distributed management approaches and the main result is that a botnet management architecture is scalable.

IRC is one communication channel used to control a botnet as in the figure 1. A user wanting to chat connects to a server and chooses a chat channel. Many users can be connected simultaneously to the same channel due to the architecture of an IRC network. In fact, several servers are interconnected and share the different channels conceptually equivalent to a multicast group. Thus all the participants are not connected to the same server and this decentralized architecture avoids server overloading. The quantity of messages is well adapted because they are often sent to the channel. The servers form a spanning tree. In a botnet, the master is connected to one server and sends the orders on a channel, the bots are connected to any servers in the network and get the orders

through the chat channel. The responses can be sent to the master in the same way also.

These previous facts are the motivation to build a management system based on an IRC botnet where an administrator requests management operations through an IRC network. However an administrator need a proof of the real efficiency and benefits of this system before deploying it. In this study, our goal is to model IRC botnet and evaluate this approach from a network management point of view by asking several questions like:

- what is the probability to reach 80% of the hosts ?
- how many servers I need to deploy ?
- how should the servers be connected ?
- how much time is needed to reach 75% of hosts ?
- what is the server load ?

Since this new management framework is based on botnet, deploying some IRC servers is needed which is not necessary with a typical centralized management solution. In all cases, the devices to be managed have to execute a specific software: an agent for a typical solution or a modfied IRC client in our case.

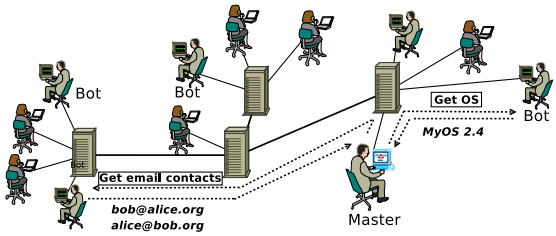


Fig. 1. An IRC botnet

3 An IRC Botnet mathematical model

Although IRC based botnets proved their efficiency in practice, little is known related to their analytical performance. The tree of servers is the main component of an IRC architecture. Thus our model is based on interconnected nodes (the servers) within a tree. We assume two kinds of failure. The first is due to the overloading of a server. The second introduces the risk to be attacked. In this case, a node or a server can be discovered by an attacker and we consider that once one node is discovered, all the system is unreliable because the attacker is able to use this server to compromise and command all the servers and bots.

The bots connected on the servers are not yet considered. The branching factor parameter m is the maximum number of adjacent links for every nodes in

the network. The number of adjacent links has to be between 1 and m and the probability function is equiprobable.

The overloading factor $\alpha(m)$ models the fact that the more a server can have connections with others, the more possible the server can be crashed due to needed operations to maintain the connectivity and synchronize the messages with the other servers. As mentionned, the worst case of the maximal possible branching factor is used which can be different from the real branching factor. Thus $\alpha(m)$ decreases when m increases and the probability for a node to have k neighbors is $p_k = \alpha(m) \times \frac{1}{m} = \dfrac{\alpha(m)}{m}$ when $1 \leq k \leq m$ i.e. k is a possible node degree except the node failure case. Obviously the degree can not be greater than the branching factor so $p_k = 0$ when $k > m$. The sum of probabilities has to be equal to one and so $p_0 = 1 - \sum_{i=1}^{m} p_i = 1 - \alpha(m)$.

The generating function of the probability function is defined as:

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k = \sum_{k=0}^{m} p_k x^k = p_0 + \sum_{k=1}^{m} p_k x^k = p_0 + \frac{\alpha(m)}{m} \times \sum_{k=1}^{m} x^k$$

This is a simple tool to compute the mean value by differentiating it;

$$\mathbb{E}(k) = G_0'(1)$$

We compute the generating function for the probability function to have k neighbors at the jth hop in a similar way as in [7] and we obtain a recursive formula;

$$G^j(x) = \begin{cases} G_0(x) \text{ when } j = 1 \\ G^{j-1}(G_1(x)) \text{ when } j \geq 2 \end{cases} \tag{1}$$

where $G_1(x) = \dfrac{G_0'(x)}{G_0'(1)}$ which is the distribution function of outgoing connections from a k degree reached node.

The average number of nodes at the jth hop is $z_j = (G^j)'(1)$. As the same manner as in [7], we have:

$$z_1 = (G^1)'(1) = G_0'(1) \qquad z_2 = (G^1(G_1))'(1) = \frac{G_0''(1)}{G_0'(1)} \times G_0'(1) = G_0''(1)$$

$$z_j = \left[\frac{z_2}{z_1}\right]^{j-1} z_1$$

Because p_0 is constant, we have:

$$G_0'(x) = \frac{\alpha(m)}{m} \sum_{k=1}^{m} k x^{k-1} \qquad G_0''(x) = \frac{\alpha(m)}{m} \sum_{k=1}^{m} k(k-1) x^{k-2}$$

Moreover the formula to compute the number of neighbors is recursive and thus the number of neighbors at the hop $j+1$ depends on the number of neighbors at hop j.

3.1 Reachability

One important entity is reachability. Assuming N nodes (servers), the reachability is the average number of reached nodes at a maximal distance k divided by the total number of nodes in the tree $\frac{\sum_{j=1}^{k} z_j}{N}$. From a network management point of view, we are able to know the potential impact of requesting a management operation and thus evaluate the possible needed additional operations for non reached hosts if necessary. We introduce the probability to have a node failure: the bigger the tree is, the more detectable it becomes and thus the probability of being attacked increases. Thus we consider that detecting the IRC network depends only on the single node/server detection probability β. So the IRC network remains undetected if all servers are not detected, that is the probability $(1-\beta)^N$. The reachability is expressed as:

$$reachability(k) = \frac{(1-\beta)^N \times min(\sum_{j=1}^{k} z_j, N)}{N} \tag{2}$$

Our model is focused on the IRC server only. Considering randomly and uniformly connected B bots, the average number of reached bots in k hops is:

$$bots(k) = reachability(k) \times B$$

The reachability metric for bots is the proportion of reached bots:

$$reachability\_bots(k) = \frac{bots(k)}{B} = reachability(k)$$

Due to the random uniform connection of bots, the different reachability metrics are the same for the bots and the servers.

3.2 Average reachability

Another useful metric is the average reachability over all possible distance in the tree. The minimal distance is one and the maximal distance is N in the case the tree is a chain. This metric gives a global overview of performance corresponding to a given N.

$$avg\_reachability(k) = \frac{\sum_{k=1}^{N} reachability(k)}{N} \tag{3}$$

3.3 The load of the system

In order to compare the botnet based management plane with typical solutions, we consider $load_s$; the overload of a server to maintain its connections with other servers and $load_c$; the overload to deal with a computer or a bot connected to the server. Thus any server needs to have sufficient resources, the load for managing C computers with a typical management solution is; $load\_server_{typical} = C \times$

$load_c$ and the worst case in our framework (when the server has the maximal branching factor m) is; $load\_server_{botnet} = C_{server} \times load_c + m \times load_s$ where C_{server} is the average number of bots per server. This formula takes in account the resources to deal with each connected bots and each of the m connected servers.

3.4 Time evaluation

Be the time t_h, the network transmission time to send a management instruction to a final host and t_s the time to forward a message between two IRC servers. In a standard management plane, the manager sends each request after sending the previous request and so the total time is; $time_{typical} = C \times t_h$

Since a server can be forced to forward message to other servers firstly, the total time is composed of the time to reach the last server (at k hops) and the time for this server to send the message to all the connected bots; $time_{botnet} = k \times t_s + C_{server} \times t_h$

This metric allows to evaluate when a request should be sent but an administrator could compute the total time of a management operations list and choose to execute urgent operations if he has not enough time.

4 Experimental results

4.1 Experimental settings

The function $\alpha(m)$ has to decrease when m increases. It has values between 0 and 1 for $2 \leq m \leq \infty$ to exclude the specific case of $m = 1$ which is limited to two nodes only. Two functions can be used for $\alpha(m)$:

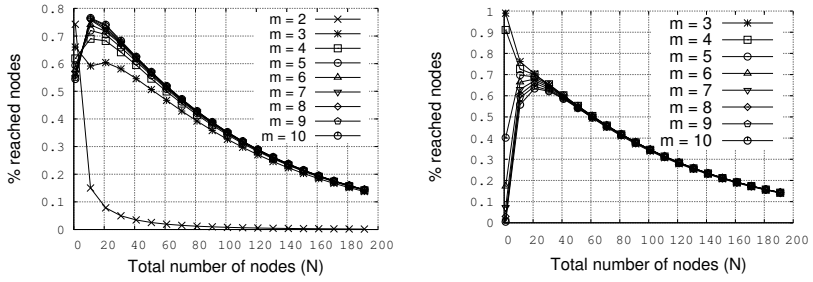
- $\alpha(m) = \alpha_1(m) = 1/m$
- $\alpha(m) = \alpha_2(m) = e^{(m-i)}$

The second function decreases more slowly than the first and the parameter i is chosen to have not too low values for the little m values. For example if we want to test m from three to five we will fix $i = 3$.

A node can be discovered with a probability $\beta = 0.01$. Other values can be used based on domain specific knowledge.

4.2 The average reachability

The first experiment is about the average reachability for different values of the branching factor m and number of nodes N. On the figure 2(a) with $\alpha(m) = \alpha_1(m)$, a curve represents a value of m between one and ten. The curves are plotted against the total number of nodes. At the beginning, more nodes can be reached and that is the reason why the curves increase. However, from a certain value of N close to 10 the curves decrease due to the parameter β: the tree is easily detectable and the reachability is affected. Thus choosing a graph with



(a) $\alpha(m) = \alpha_1(m)$ (b) $\alpha(m) = \alpha_2(m)$

Fig. 2. The average reachability with different branching factors

more than 10 nodes has limited performances in term of reachability. Since the branching factor concerns the children and the parent link, the values of the curve for $m = 2$ are very low (chain tree) as we can see on the figure 2(a), this is also one of the reason to exclude this trivial case.

The results with $\alpha(m) = \alpha_2(m)$ are on the figure 2(b). The difference is at the beginning because the curves with little branching factors are less affected when using $\alpha_2(m)$ i.e $\alpha_2(m) > \alpha_1(m)$. Therefore the maximal branching factor m has only an impact on the average reachability for small value of N (less than 20). When there are more than twenty nodes, the curves depend more on β.

The absolute number of reached nodes is plotted on the figure 3(a). All the curves are similar and there is a maximum number of reached nodes of about 35 nodes for 100 nodes in the graph. This means that there is no need to have more than 100 nodes in the tree because the more nodes there are, the fewer they could be reached. Thus, a management architecture can be based on the absolute value of reached nodes or on the reachability depending on the constraints and the choices of the target environment and the business requirements. For example, with only 100 bots on a server, we can manage 3500 computers. Thanks to formula in subsection 3.3, the total load of the managing station is;

$$load\_server_{typical} = 3500 \times load_c$$

Considering 100 hosts per server in our framework, the total load of a server is;

$$load\_server_{botnet} = 100 \times load_c + m \times load_s$$

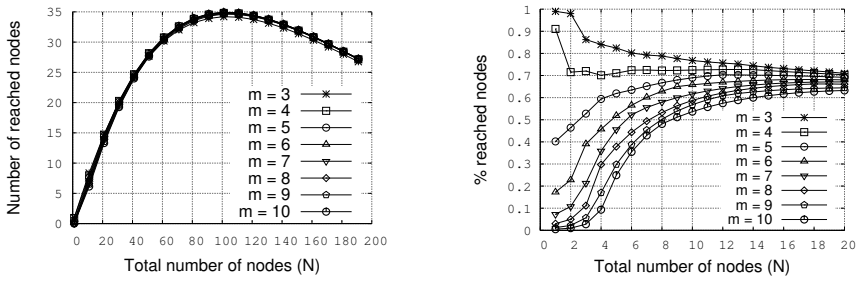
Our goal is to diminish the load of a server which implies to have:

$$load_s < \frac{3400}{m} load_c$$

We assume that $load_s > load_c$. Indeed a server has to maintain channels, to synchronize the messages and to maintain the connections and the topology. However even though a server is connected to many others as 10 for example,

$load_s$ can be equal to $340 \times load_c$ without loss of performance. This can be improved by decreasing the branching factor.

Except for $m = 2$, the curves are different with low values for N. With $\alpha_1(m)$, the higher the branching factor is, the higher the reachability is. This is in contradiction with the second case with $\alpha_2(m)$ presented on the figure 3(b). The reachability is the results of two antagonist "forces": the branching factor and $\alpha(m)$ i.e. the probability to have a node failure.



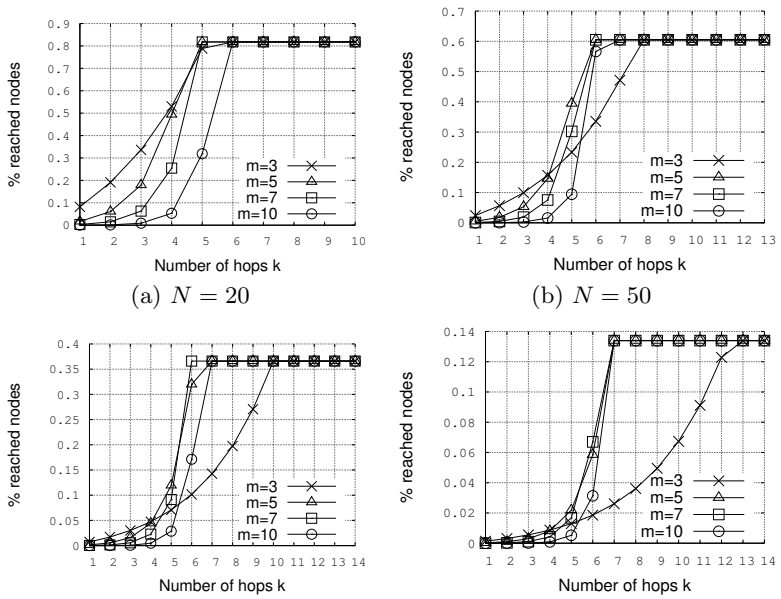
(a) The absolute value of reached nodes (b) The average reachability with $1 \leq N \leq 20$

Fig. 3. Reached nodes with $\alpha(m) = \alpha_2(m)$

4.3 The number of hops

The impact of the number of hops is studied with different values of N, respectively 20, 50, 100 and 200 on respectively the figure 4(a), 4(b), 4(c) and 4(d). All figures highlight a maximum value. This value depends only on β as can be observed in formula (2). For example for $N = 20$, the maximum of nodes that can be reached is limited by $(1 - \beta)^{20} = (1 - 0.01)^{20} = 0.81$ which is the probability to not be discovered. Using this method, we can know the maximum number of nodes/servers by fixing the maximal reachability.

The number of needed hops is very important because it is equivalent to the time needed to perform management operations. In this test only the function $\alpha_2(m)$ is used. On the figure 4(a), N is equal to 20 and the probability to be discovered is not important, which contrasts with the probability to have a node failure. Therefore a small branching factor is needed to have the best performances. We assume that the more there are nodes, the more effective a high branching factor is. This is highlighted by other curves in figure 4. For instance a branching factor of three has worse performances when N increases. However, as we can observe, an high branching factor like ten is never better than seven. To be brief, depending on how many bots have to be managed and so how many servers can be used, a different value for the branching factor has to be selected. A good compromise for high and low values of N is $m = 5$. Moreover the maximum value is reached within 5, 6 or 7 hops in main cases. So for a botnet based



(a) $N = 20$ (b) $N = 50$

(c) $N = 100$ (d) $N = 200$

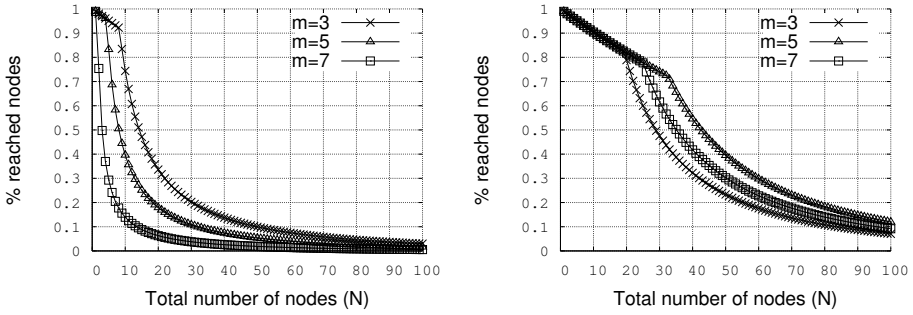
Fig. 4. The relationship between the reachability and the branching factor

management plane, the request will not pass through many servers which saves a lot of resources.

On the figure 4(b) with $m = 5$, $0.6 \times 50 = 30$ servers are reached in 6 hops. If there are 100 bots per server, it corresponds 3000 hosts. The total needed time to send a management request is given in 3.4 and is $6 \times t_s + 100 \times t_h$ which has to be lower or equal to $3000 \times t_h$ i.e $t_s < 483 \times t_h$ to obtain equivalent performance. To conclude, even though intermediate nodes are added, the delay is greatly reduced. In the previous section, we saw that a typical management solution implies more server overloading which can increase the delay.

4.4 The impact of the number of nodes

There are two factors which affect the reachability: the probability to be discovered β and the probability to have a node failure $\alpha(m)$. The figure 5 shows the reachability for a fixed number of hops k and with $\alpha(m) = \alpha_2(m)$. There is always a first stage where the curves decrease slowly. This first stage corresponds to reach all nodes limited by the probability to be discovered. After this moment, the curves decrease drastically when a specific N value is reached. In fact this value depends on the branching factor which limits the value of reached nodes (antagonist effects of m and $\alpha(m)$). Once again the branching factor $m = 5$ is the best compromise for different number of hops because on the figure 5(a), it's



(a) j = 3 (b) j = 5

Fig. 5. The reachability according the number of nodes in the graph and the number of hops j

the best curve and on the figure 5(b) it is the second best curve. We tested also $j = 7$ and saw that the branching factor $m = 5$ was the best case again.

5 Related works

Botnets are a powerful tool for attackers needing to manage large armies of compromised machines. Since a botnet can be build by a worm infecting several machine, the authors in [8] give full details about worms: definition, traffic patterns, history, ideas about future worms, detection and defense. Many papers about worm propagation were published and [9] describes the propagation of the CodeRed worm where the classical Kermack-McKendrick model is extended. The authors in [10] aim to determine the characteristics of some well known botnet families and show that all of them are based on IRC. In [11], the different ways to construct a botnet are discussed and the authors highlight that using IRC networks limits the latency and preserves the anonimity of the controller. With respect to these works we are the first to build an analytical model and evaluate experimentally the performances. A classical network and service management framework is a simple direct exchange of commands between the manager and the agents to manage. Several solutions were proposed to deal with a large scale management like the management by delegation [12]. Another solution is to deploy cascaded managers managers [13]. The mid-level managers can translate a management request into subrequests for each device they manage or get the results and analyze them. A similar approach is based on active network [14] [15] [16] where the network equipments execute and forward a light management program but the main disadvantage is that dedicated network equipments are needed. Our approach needs only to deploy some servers. In [17] a decentralized management is proposed where a query is send to a starting node which is responsible to distribute subqueries and get the results thanks to the echo pattern. Thanks to another subquery, each node is able to create an

aggregate result which is sent to the upstream node in the echo pattern. The starting node can finally provide the final result by using another subquery. In [18], the authors propose a solution for the specific case of the distribution of Windows update patches unlike our work which can be used for various applications. First of all, they observe that efficient clustering of patches is possible and so the generation of delta files (the final file which depends on a specific configuration) should be delegated to the servers in order to reduce the traffic. A dedicated solution to ad-hoc networks in [19] determines groups of machines which have a high connectivity within a group and elect a manager inside it. In [20], a worm is designed to patrol on different hosts to get information before coming back to the manager. The authors propose to use a worm which moves on the different hosts by looking for a way to reach them. However the experiments are very limited and the worm acts only as a monitor. Finally, some real examples of salutary worms exist like Code-Green [21] which counters the Code-Red. However these worms are not controlled and not limited to certain hosts. As mentionned, a lot of these approaches need to have active components like active mid-level managers which are in charge of analyzing and translating the queries. Our system is composed of passive servers which forward only the requests. Furtermore bypassing most of active equipments is an advantage of a botnet based management plan since only outgoing connections are used.

6 Conclusion and future works

The current malware seems to have a highly efficient communication channel. The effectiveness and the scalability of botnets is a proof that a large scale management is possible by adapting the botnet model. In this paper, we described a model for IRC based communication in a network management environment. A mathematical model of an IRC network and different performance metrics are proposed. The reachability metric is the percentage of reached hosts and can be calculated according the time delays. Secondly, as requests are transmitted through huge networks like the Internet, the system exposed to different problems: network failures or delay, denial of service attacks against the mid-level managers. We introduce these elements to evaluate the ability of our approach to continue to work when facing these issues. Now, we are able to determine the number of reached hosts for a certain network configuration and optimize it to have specific results. Due to the possibility to be attacked, the number of IRC servers has to be large enough but not too large in order to have a high reachability. Since our model is only analytical, the next step is to test an implementation. A case study is introduced in [4] where we defined a malware based large scale management plane to create and manage a honeynet to detect criminal predators in a P2P environment. Future work will consist in modeling epidemic propagation and experimenting our solution on a large grid network.

Acknowledgment This paper was supported in part by the EC ISTE-MANICS Network of Excellence (#26854).

References

1. McLaughlin, L.: Bot software spreads, causes new worries. IEEE Distributed Systems Online **5**(6) (2004)
2. Oikarinen, J., Reed, D.: rfc 1459: Internet relay chat protocol (1993)
3. Canavan, J.: The evolution of malicious irc bots. In: Proceedings of the Virus Bulletin Conference (VB2005), Dublin, Ireland, October 2005. (2005)
4. State, R., Festor, O.: Malware: a future framework for device, network and service management. Journal in Computer Virology **3**(1) (2007) 51–60
5. How to Own the Internet in Your Spare Time, USENIX Association (2002)
6. Chen, T.M., Liu, S.S.: A model and evaluation of distributed network management approaches. Selected Areas in Communications, IEEE Journal on **20**(4) (2002) 850–857
7. Ramachandran, K., Sikdar, B.: Modeling malware propagation in gnutella type peer-to-peer networks. International Parallel and Distributed Processing Symposium, 2006 (2006)
8. Nazario, J.: Defense and Detection Strategies against Internet Worms. Artech House, Inc., Norwood, MA, USA (2003)
9. Zou, C., Gong, W., Towsley, D.: Code red worm propagation modeling and analysis (2002)
10. Barford, P., Yegneswaran, V.: 1. In: An inside look at Botnets. Springer (2006)
11. Cooke, E., Jahanian, F., Mcpherson, D.: The zombie roundup: Understanding, detecting, and disrupting botnets. (June 2005) 39–44
12. Goldszmidt, G., Yemini, Y.: Distributed management by delegation. In: 15th International Conference on Distributed Computing Systems, IEEE Computer Society (1995)
13. SNMP, Research: The mid-level manager. http://www.snmp.com/products/mlm.html (accessed on 07/30/07)
14. Schwartz, B., Jackson, A.W., Strayer, W.T., Zhou, W., Rockwell, R.D., Partbridge, C.: Smart packets: applying active networks to network management. ACM Transactions on Computer Systems **18**(1) (2000) 67–88
15. Brunner, M., Stadler, R.: The impact of active networking technology on service management in a telecom environment. In: IFIP/IEEE International Symposium on Integrated Network Management, Boston (1999)
16. Brunner, M., Stadler, R.: Management in telecom environments that are based on active networks. Journal of High Speed Networks (2001)
17. Lim, K.S., Stadler, R.: Real-time views of network traffic using decentralized management. In: Integrated Network Management, 2005. 9th IFIP/IEEE International Symposium on. (2005)
18. Gkantsidis, C., Karagiannis, T., VojnoviC, M.: Planet scale software updates. SIGCOMM Comput. Commun. Rev. **36**(4) (2006) 423–434
19. Badonnel, R., State, R., Festor, O.: Probabilistic management of ad-hoc networks. In: 10th IEEE/IFIP Network Operations and Management Symposium - NOMS 2006, IEEE Computer Society (2006)
20. Ohno, H., Shimizu, A.: Improved network management using nmw (network management worm) system. In: Proceedings of INET'95: Honolulu, Hawai'i, June 27-30, 1995. (1995)
21. Szor, P.: The Art of Computer Virus Research and Defense. Addison-Wesley Professional (2005)