Report Number: wucse-2009-7

2009

Defending Against Distributed Denial-of-Service Attacks With Weight-Fair Router Throttling

Abusayeed Saifullah

A high profile internet server is always a target of denial-of-service attacks. In this paper, we propose a novel technique for protecting an internet server from distributed denial-of-service attacks. The defense mechanism is based on a distributed algorithm that performs weight-fair throttling at the upstream routers. The throttling is weight-fair because the traffics destined for the server are controlled increased or decreased) by the leaky-buckets at the routers based on the number of users connected, directly or through other routers, to each router. To the best of our knowledge, this is the first weight-fair technique for saving an... Read complete abstract on page 2.

Defending Against Distributed Denial-of-Service Attacks With Weight-Fair Router Throttling

Abusayeed Saifullah

Complete Abstract:

A high profile internet server is always a target of denial-of-service attacks. In this paper, we propose a novel technique for protecting an internet server from distributed denial-of-service attacks. The defense mechanism is based on a distributed algorithm that performs weight-fair throttling at the upstream routers. The throttling is weight-fair because the traffics destined for the server are controlled increased or decreased) by the leaky-buckets at the routers based on the number of users connected, directly or through other routers, to each router. To the best of our knowledge, this is the first weight-fair technique for saving an internet server from denial-of-service attacks. The system is guaranteed to work even if some of the routers are compromised. Furthermore, in the beginning of the algorithm, the server's capacity is underestimated by the routers so as to protect the server from any sudden initial attack.

2009-7

Defending Against Distributed Denial-of-Service Attacks With Weight-Fair Router Throttling

Authors: Abusayeed Saifullah

Corresponding Author: saifullaha@cse.wustl.edu

Web Page: http://www.cse.wustl.edu/~saifullaha

Abstract: A high profile internet server is always a target of denial-of-service attacks. In this paper, we propose a novel technique for protecting an internet server from distributed denial-of-service attacks. The defense mechanism
is based on a distributed algorithm that performs weight-fair throttling at the upstream routers. The throttling is weight-fair because the traffics destined for the server are controlled increased or decreased) by the leaky-buckets at the routers based on the number of users connected, directly or through other routers, to each router. To the best of our knowledge, this is the first weight-fair technique for saving an internet server from denial-of-service attacks. The system is guaranteed to work even if some of the routers are compromised. Furthermore, in the beginning of the algorithm, the server's capacity is underestimated by the routers so as to protect the server from any sudden initial attack.

Type of Report: Other

Defending Against Distributed Denial-of-Service Attacks With Weight-Fair Router Throttling

Abusayeed M Saifullah
Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130 USA
Email: saifullaha@cse.wustl.edu

ABSTRACT

A high profile internet server is always a target of denial-of-service attacks. In this paper, we propose a novel technique for protecting an internet server from distributed denial-of-service attacks. The defense mechanism is based on a distributed algorithm that performs weight-fair throttling at the upstream routers. The throttling is weight-fair because the traffics destined for the server are controlled (increased or decreased) by the leaky-buckets at the routers based on the number of users connected, directly or through other routers, to each router. To the best of our knowledge, this is the first weight-fair technique for saving an internet server from denial-of-service attacks. The system is guaranteed to work even if some of the routers are compromised. Furthermore, in the beginning of the algorithm, the server's capacity is underestimated by the routers so as to protect the server from any sudden initial attack.

1 Introduction

Distributed denial-of-service, abbreviated as DDoS, attack is considered as one of the most serious attacks over internet [4, 17]. It is an attempt by the malicious users to make a networked resource unavailable to its legitimate users. A denial-of-service attack can be perpetrated either by flooding a network or by disrupting a server by sending more requests than it can possibly handle, thereby preventing access to a service. There are many other forms of DDoS attacks. A better taxonomy is available in [15]. High-profile internet servers as well as premium sites with huge customer utilization are the basic targets of the attackers.

Due to the seriousness and ultimate effects of the denial-of-service attack, its detection and prevention call for considerable attention in the network security community all over the world. Different detection

mechanism are available in the literature [2, 5, 6, 10, 12]. The accelerated research for countering DDoS attacks resulted into a flurry of papers over the last decade [3, 7, 8, 9, 11, 13, 14, 16, 18, 19, 20]. However, none of the mechanisms proposed so far can completely protect the resource under DDoS attack. Most of the techniques are proactive and try to shield the resource after aggressive traffic overwhelms it. To protect an internet server, the method proposed by Yau et al. [21] installs a uniform leaky bucket in a set of routers that forward traffic towards the server. The technique sounds better than the recursive push-back mechanism proposed by Mahajan et al. [13]. However, Yau et al. [21] assumes that the server remains active and healthy after the first attack and launches its defense mechanism after the first attack. The mechanism is initiated by multicasting a rate increase or rate decrease signal from the server towards a set of routers selected and maintained by the server. That is, the server has to keep working all the time to counteract the attack. Furthermore, since a subset of users forward traffic to a common router responsible for throttling, if any throttling router is compromised, the server may get vulnerable to the attack. The most remarkable drawback lies in the fact that the mechanism is not completely fair. Although fairness is defined and claimed throughout their paper, we analyze the fairness from the perspective of router profile. For example, let a throttling router A is connected to only one legitimate user who sends traffic towards A at a rate of 10. Let B be another throttling router to which 3 legitimate users forward traffic, each at a rate of 8. Let Yau et al. [21] installs a uniform leaky bucket of size 20 at every throttling router. In this scenario, half of the bandwidth at the router A is not used at all even when the user connected to A sends all the traffics. On the other hand, neither of the legitimate users connected to B can send their full traffic to B due to lack of bandwidth at B. The most disappointing scenario is viewed when A is connected to an attacker only. The attacker uses the entire bandwidth of A and keeps sending traffic to the server at a rate of 20 while a legitimate user connected to B can send only a fraction of its traffic. Thus, Yau et al. [21] does not distinguish between the heavy routers and the light routers and hence is considered unfair.

We point out all these issues and propose a distributed algorithm based on weight-fair router throttling that counteracts denial-of-service attacks directed to an internet server. The protection mechanism is similar to that of Yau et al. [21] in the sense of throttling the upstream routers. However we install leaky buckets of different capacities and the buckets are installed at all routers instead of a uniform leaky bucket in a subset of routers. The fairness is achieved by giving more bandwidth to the routers connected to higher number of legitimate users and vice versa. To the best of our knowledge, this is the first weight-fair technique for saving an internet server from denial-of-service attacks. For a dynamic environment where the number of

users changes frequently, a breadth-first search tree is constructed and hence every router, and the server as well, can calculate the number of users within its subtree. Since we install leaky buckets in all routers, not all the routers need to be trust-worthy. That is, if any router is compromised, the upper-level routers (nearer to the server) can overcome it. All the computations are performed locally. In the beginning of the algorithm, the server's capacity is underestimated by the routers so as to protect the server from any sudden initial attack. The rate is updated (increased or decreased), based on the servers feedback sent to its child routers and eventually propagated downward to all routers, in the subsequent rounds of the algorithm with a view to converging the total server load to the server's tolerable capacity range .

2 Model of Computation

The model for our DDoS defense mechanism is similar to that of Yau et al. [21]. The server under consideration for defense is represented by s. The distributed system is modelled as a connected graph $G = (V, E)$ where V is the set of networked machines and each $e \in E$ represents a communication link. Communication is restricted between the neighbors. Our algorithm is applied on a breadth-first search tree T_{bfs} of G rooted at s. In T_{bfs}, any subtree rooted at $v \in V$ is denoted by $T(v)$. All leaf nodes are hosts or users and thus can be a traffic source. "*An internal node is a router; a router cannot generate traffic, but can forward traffic received from its connected hosts or peer router*" [21]. The set of internal routing nodes is denoted by R. A host can either be a legitimate user (i.e. good user) or be an attacker. The set of legitimate (i.e. good) users and the set of attackers are denoted by U_g and U_a, respectively. The terms *user* and *host* are used interchangeably throughout the paper.

All server load quantities and traffic rates are measured in units of kb/s. The traffic that a user u, either a good user or an attacker, wants to send is denoted by $f(u)$. For a good user g and an attacker a, we assume that $f(a) \gg f(g)$. "*This is because if every attacker sends at a rate comparable to a good user, then an attacker must recruit or compromise a large number of hosts to launch an attack with sufficient traffic volume*" [21]. The total load that the server s desires to receive at a time is denoted by L while the total traffic received at the server at a time (i.e. the rate of received traffic) is $f(s)$. However, the server has another parameter named *tolerance*, denoted by d, $d \ll L$, meaning that the server can tolerate even if it receives d loads beyond L. The server load, and hence the system, *converges* if the server's traffic receiving rate is within its capacity range i.e. $L - d \leq f(s) \leq L + d$.

3 Weight-Fair Router Throttling

The idea underlying our proposed protection mechanism is to set up non-uniform leaky-buckets at the routers and control the traffic rate at each router based on the number of users that forward traffic to that router. To count the number of users under a router, specially in a dynamic environment, we need to maintain a breadth-first search tree T_{bfs} rooted at the server s of our system $G = (V, E)$.

The traffics destined for the server are controlled by the leaky-buckets at the routers on different levels of the breadth-first search tree T_{bfs}. In T_{bfs}, the numbers of leaf nodes in each subtree are computed in a bottom up manner. We denote the number of hosts or users (i.e. leaf nodes) under $T(v)$, the subtree rooted at v, by $l(v)$. For any leaf node v, $l(v) = 1$. The routers that are directly connected only to the leaf nodes first calculate these numbers. Based on these values, the next level routers (i.e. the parents of those routers) then compute these numbers. Thus, eventually, the server s computes the total number of leaf nodes, $l(s)$, in T_{bfs}. That is, every non-leaf node v, either a router or the server, computes $l(v)$ by summing the values $l(x)$, where x is a child of v as shown in Equation 1:

$$l(v) = \sum l(x), \forall x \text{ such that } x \text{ is a child of } u; \tag{1}$$

The algorithm installs a leaky-bucket at each upstream router. The capacity of the bucket at router v is denoted by $b(v)$. The goal of the algorithm is to update these capacities so that the routers' throttles are weight-fair and the traffic load at the server converges.

Definition 1. *Router throttling is called **weight-fair** when, for any pair of routers u and v, $b(v) \geq b(u)$ if and only if $l(v) \geq l(u)$ and the total traffic forwarded to v is greater or equal to that forwarded to u.*

Definition 2. *Router throttling mechanism is said to be **converged** if and only if $L - d \leq f(s) \leq L + d$, where $f(s)$ is the current total server load, L is the desired server load, and d is the tolerance of the server.*

For server s, $f(s)$ is the total traffic received at the server at a time. For any router $v \in R$, $f(v)$ is the total traffic passed through the leaky bucket at v at a time. For any $v \in U$, $f(v)$ is the total traffic v wants to send to the server through the router to which v is connected. Let $c(v)$ be the total number of children of $v \in V$ in T_{bfs}. For every node $v \in V$, we define $f_c(v)$ as follows:

$$f_c(v) = min\{f(x) | x \text{ is a child of } v \text{ in } T_{bfs}\}; \tag{2}$$

That is, $f_c(v)$ is the minimum traffic that a child of v wants to forward to v.

In the beginning of the algorithm, the server's capacity is underestimated by the routers so as to protect the server from any sudden initial aggression. Thus, initially, for each router $v \in R$, the size of the leaky bucket $b(v)$ is set as follows:

$$b(v) = c(v) * f_c(v); \tag{3}$$

Thus the initial capacity of the router is set based on the total number of users in $T(v)$. The higher the number of users in $T(v)$ as well as the total traffic forwarded to v, the higher the leaky bucket capacity in v.

The initial capacities of the routers are calculated in a bottom up manner in T_{bfs} in the first round of the algorithm. First every router v that is directly connected only to the users (leaf nodes) calculate its capacity $b(v)$ based on $f_c(v)$. Then every user (good user or attacker) forwards $f_c(v)$ traffic to v. Thus total traffic at v becomes $c(v) * f_c(v)$ and the router gets saturated. Then every router u on the next higher level (nearer to the root) calculates $b(u)$ based on the minimum traffic that one of its children wants to forward to it (i.e. $f_c(u)$). The router u then receives $f_c(u)$ traffic from each of its $c(u)$ children. Thus the whole capacity of each leaky-bucket is utilized in the first round. The traffic ultimately is forwarded to the server. Since every leaky bucket has very small capacity in the beginning, the total traffic received at the server in the first round is usually much less than its desired load L.

The server then calculates a feedback value, r, which triggers the next round of the algorithm. The feedback value r which may be positive, negative, or zero meaning that the routers' capacities should be increased, decreased, or left unchanged, respectively, is calculated as follows :

$$r = (L - f(s)/l(s); \tag{4}$$

where $l(s)$ is the total number of hosts in the network.

The value r signifies that the excess load the server is receiving or the amount of load the server can still receive beyond its current load is equally distributed among the users. Specifically, if r is positive then every user, if necessary, can increase its traffic rate by r, and if r is negative then every user should decrease its traffic rate by r. The server s first sends the feedback value r to its child routers. This feedback is then propagated downward in the tree in the form of a message to increase or decrease the leaky bucket capacity. Let $p(u)$ be the parent node of $u \in V$ in T_{bfs}.

Remark 1. *If $u = s$ (the server) i.e. the root of T_{bfs}, then $p(u) = null$.*

Remark 2. *The server s has no leaky-bucket but, for generalization, we consider that the value $b(s) = L$.*

Every router u, after receiving r from its parent node, updates its leaky bucket capacity based on the total number of users (leaf nodes) in $T(u)$. However, even if some router has too many users under it, its capacity is never increased beyond the capacity of its parent router so that the packets that will be dropped ultimately are no longer forwarded. Therefore, every router u updates its capacity $b(u)$ according to Equation 5:

$$b(u) = min\{b(p(u)), b(u) + r * l(u)\}; \tag{5}$$

Thus the capacities of the leaky buckets in the routers having more users within their subtrees are increased (decreased, respectively) more when r is positive (negative, respectively). Once the capacities are updated, the traffic from the users towards the server through the routers are updated. The entire process is continued until the server load $f(s)$ is between $L - d$ and $L + d$ i.e. $L - d \leq f(s) \leq L + d$.

The idea can be illustrated by an example. Figure 1 shows a breadth-first search tree T_{bfs} rooted at the server s. In the figure, the round nodes are the routers and the square nodes are the users. The value beside every node v shows the number $l(v)$, the number of leaf nodes in $T(v)$, calculated using Equation 1. For routers C, G, B, F, E, D, A, and the server s, $l(C) = 1, l(G) = 2, l(B) = 1, l(F) = 3, l(E) = 4, l(D) = 5, l(A) = 6$, and $l(s) = 7$, respectively.

Figure 2 shows the execution of the first round of the algorithm. In the example, $L = 26$, $d = 3$. The users u_4 and u_6 are considered as attackers. The traffics that users $u_1, u_2, u_3, u_4, u_5, u_6$ and u_7 want to send to their respective routers are 5, 2, 3, 40, 5, 50, and 4, respectively. For the routers, according to Equation 2 and Equation 3:

$f_c(C) = min\{5\} = 5$; $b(C) = c(C) * f_c(C) = 1 * 5 = 5$;

$f_c(G) = min\{3, 40\} = 3$; $b(G) = c(G) * f_c(G) = 2 * 3 = 6$;

$f_c(B) = min\{4\} = 4$; $b(B) = c(B) * f_c(B) = 1 * 4 = 4$;

$f_c(F) = min\{6, 5\} = 5$; $b(F) = c(F) * f_c(F) = 2 * 5 = 10$;

$f_c(E) = min\{10, 50\} = 10$; $b(E) = c(E) * f_c(E) = 2 * 10 = 20$;

$f_c(D) = min\{2, 20\} = 2$; $b(D) = c(V) * f_c(D) = 2 * 2 = 4$;

$f_c(A) = min\{5, 4\} = 4$; $b(A) = c(A) * f_c(A) = 2 * 4 = 8$;

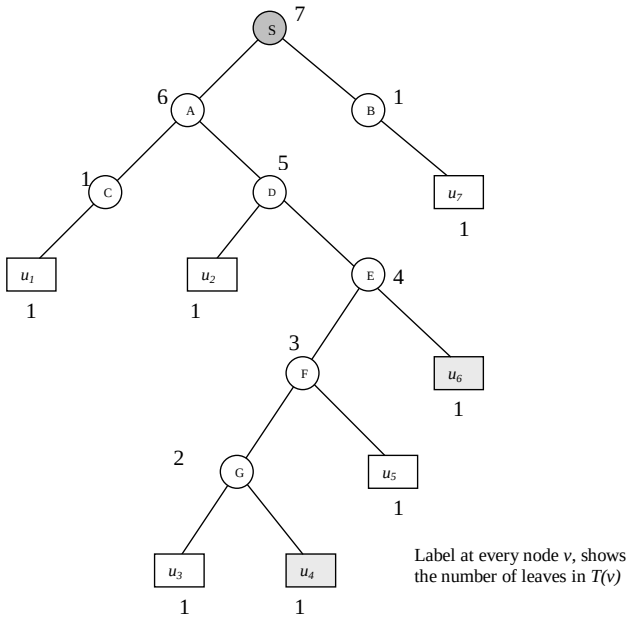

Figure 1: T_{bfs} showing number of leaf nodes in each subtree

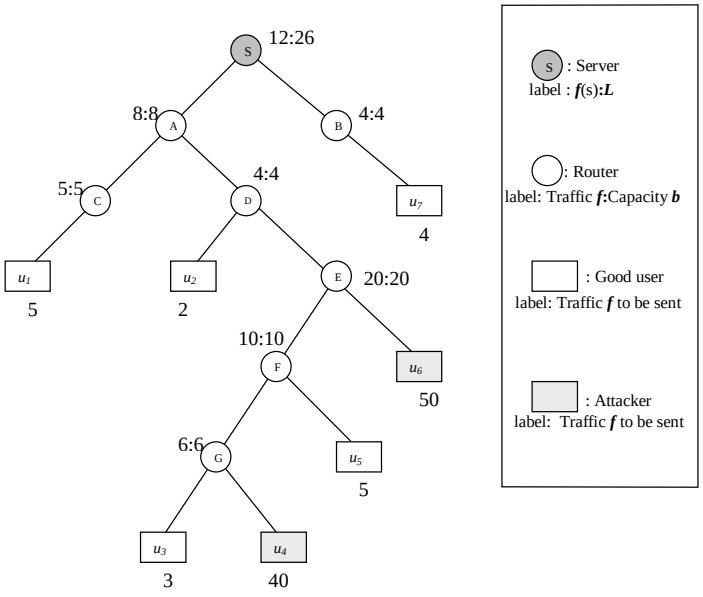

Figure 2: Round 1

The total flow at the server s is 8+4=12. This load is much less than the server's desired load. That is, s can receive more loads. It then calculates feedback r according to Equation 4 and r is propagated to all other routers.

$r = (26 - 12)/7 = 2$;

The second round of the algorithm is shown in Figure 3. Based on $r = 2$, all the routers update their capacities according to Equation 5:

$b(A) = min\{b(s), b(A) + 2 * l(A)\} = min\{26, 8 + 2 * 6\} = 20$;

$b(B) = min\{b(s), b(B) + 2 * l(B)\} = min\{26, 4 + 2 * 1\} = 6$;

$b(C) = min\{b(A), b(C) + 2 * l(C)\} = min\{20, 5 + 2 * 1\} = 7$;

$b(D) = min\{b(A), b(D) + 2 * l(D)\} = min\{20, 4 + 2 * 5\} = 14$;

$b(E) = min\{b(D), b(E) + 2 * l(E)\} = min\{14, 20 + 2 * 4\} = 14$;

$b(F) = min\{b(E), b(F) + 2 * l(F)\} = min\{14, 10 + 2 * 3\} = 14$;

$b(G) = min\{b(F), b(G) + 2 * l(G)\} = min\{14, 6 + 2 * 2\} = 10$;

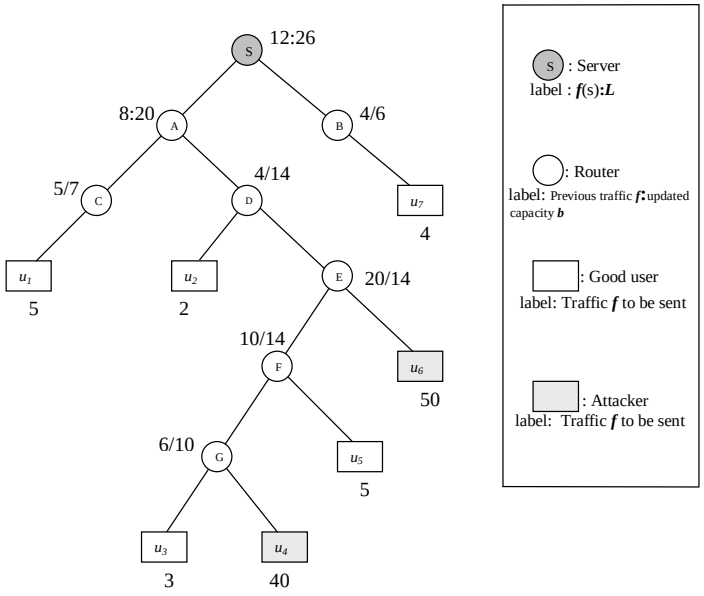

Figure 3: Round 2

Once the capacities are updated, the traffics through the routers are updated in the third round as shown in Figure 4. The updated traffics through the routers, C, G, B, F, E, D, and A are 5, 10, 4, 14, 14, 14, and 19, respectively. Thus the total traffic at s is 19+4=23. Since the tolerance d is 3, the server load is within its tolerable range (i.e. $23 \leq 23 \leq 29$). Hence the system is said to be converged.

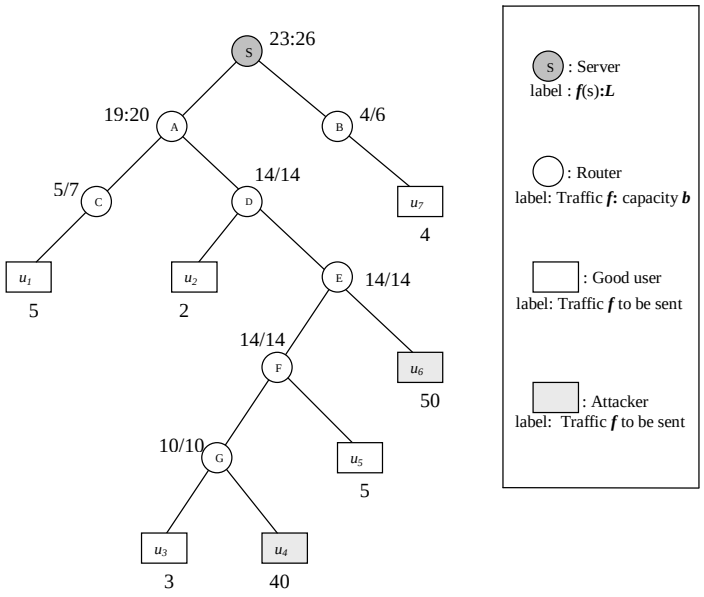

Figure 4: Round 3

4 Conclusion

Our proposed idea is the first to deal with weight-fairness issue. Furthermore, this will ensure more security than that of the recursive push-back mechanism proposed in [13] and the router throttle mechanism proposed by Yau et al. [21]. Above all, the convergence of our algorithm is expected to be much faster than that of Yau et al. [21]. We will demonstrate our weight-fair throttling technique by showing empirical results using the NS-2 [1] simulator in the future version of the paper.

References

[1] www.isi.edu/nsnam/ns/.

[2] Change-point monitoring for the detection of dos attacks. *IEEE Trans. Dependable Secur. Comput. 1*, 4 (2004), 193–208. Member-Haining Wang and Member-Danlu Zhang and Fellow-Kang G. Shin.

[3] D-ward: A source-end defense against flooding denial-of-service attacks. *IEEE Trans. Dependable Secur. Comput. 2*, 3 (2005), 216–232. Member-Jelena Mirkovic and Member-Peter Reiher.

[4] BARFORD, P., KLINE, J., PLONKA, D., AND RON, A. A signal analysis of network traffic anomalies. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment* (New York, NY, USA, 2002), ACM, pp. 71–82.

[5] BLAZEK, R. A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods. *Proc. IEEE Workshop Information Assurance and Security* (2001), 220–226.

[6] CARL, G., KESIDIS, G., BROOKS, R. R., AND RAI, S. Denial-of-service attack-detection techniques. *IEEE Internet Computing 10*, 1 (2006), 82–89.

[7] CHEN, S., AND CHOW, R. A new perspective in defending against ddos. In *FTDCS '04: Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 186–190.

[8] CHEN, S., AND SONG, Q. Perimeter-based defense against high bandwidth ddos attacks. *IEEE Trans. Parallel Distrib. Syst. 16*, 6 (2005), 526–537.

[9] CHEN, S., TANG, Y., AND DU, W. Stateful ddos attacks and targeted filtering. *Journal of Network and Computer Applications (Special Issue on Distributed Denial of Service and Intrusion Detection) 30*, 3 (2007), 823–840.

[10] CHEN, Y., HWANG, K., AND KU, W.-S. Distributed change-point detection of ddos attacks: experimental results on deter testbed. In *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007* (Berkeley, CA, USA, 2007), USENIX Association, pp. 7–7.

[11] FEINSTEIN, L., SCHNACKENBERG, D., BALUPARI, R., AND KINDRED, D. Statistical approaches to ddos attack detection and response. *discex 01* (2003), 303.

[12] GAVRILIS, D., AND DERMATAS, E. Real-time detection of distributed denial-of-service attacks using rbf networks and statistical features. *Comput. Netw. ISDN Syst. 48*, 2 (205), 235–245.

[13] MAHAJAN, R., BELLOVIN, S. M., FLOYD, S., IOANNIDIS, J., PAXSON, V., AND SHENKER, S. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev. 32*, 3 (2002), 62–73.

[14] MIRKOVIC, J., PRIER, G., AND REIHER, P. Attacking ddos at the source. *icnp 00* (2002), 312.

[15] MIRKOVIC, J., AND REIHER, P. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev. 34*, 2 (2004), 39–53.

[16] MIRKOVIC, J., ROBINSON, M., AND REIHER, P. Alliance formation for ddos defense. In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms* (New York, NY, USA, 2003), ACM, pp. 11–18.

[17] MOORE, D., VOELKER, G. M., AND SAVAGE, S. Inferring internet Denial-of-Service activity. pp. 9–22.

[18] PARK, K., AND LEE, H. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2001), ACM, pp. 15–26.

[19] SCHUBA, C. L., KRSUL, I. V., KUHN, M. G., SPAFFORD, E. H., SUNDARAM, A., AND ZAMBONI, D. Analysis of a denial of service attack on tcp. In *SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 1997), IEEE Computer Society, p. 208.

[20] SUNG, M., AND XU, J. Ip traceback-based intelligent packet filtering: A novel technique for defending against internet ddos attacks. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 302–311.

[21] YAU, D. K. Y., LUI, J. C. S., LIANG, F., AND YAM, Y. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Trans. Netw. 13*, 1 (2005), 29–42.