Optimal Control of DDoS defense with Multi-Resource Max-min Fairness

Wei Wei, Yabo Dong, Dongming Lu
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
{weiwei_tc, dongyb, ldm}@zju.edu.cn

Guang jin
College of Information Science and Engineering
Ningbo University
Ningbo, China
d05jinguang@zju.edu.cn

***Abstract*—Distributed defense of DDoS (Distributed Denial of Service) attack has been extensively researched in recent years and control-based defense is a hopeful way. However, existed methods only deal with bandwidth protection. The paper takes defense of DDoS flood as a kind of Processing and Bandwidth Resources allocation and solves it using control theory. Our defense mechanism FFDRF (Feedback Filtering with Dual-Resource Fairness) sets up filters in edge routers of AS and adjusts the filtering thresholds through feedback between these routers and the victim. The simulation results show that FFDRF can make the legitimate traffic keep high survival rate while is stable and converges quickly even in case of heterogeneous flow sources and link conditions. Compared with level-k max-min fairness defense, FFDRF is more effective against CPU-consuming flood. And an implementation of FFDRF in a linux router indicates that FFDRF is feasible in real-life routers.**

***Keywords*—DDoS, max-min fairness, filtering**

I. INTRODUCTION

DDoS is a serious threat to Internet security nowadays and packet flood is a prevalent form of DDoS. Because attack flows exhaust the bandwidth in bottleneck link near victim, the local defense by victim is usually not effective. As a result, distributed defense was proposed. Its principle is to coordinate between several upstream routers of victim and filter out attack traffic as close to sources as possible. Most of existed methods need complicated communication between routers and involve lots of routers when deployment approaches the attacking sources. In this situation, AS (Autonomous System) based defense is a feasible solution and it involves limited number of routers. More importantly, it avoids deployment across ASes and is more practical than most of other solutions. In the method, when under packet flood, the victim sends filter messages to edge routers and then traffic to the victim is filtered accordingly in every edge router. The victim periodically checks the rate of traffic to itself and sends changed filter messages to edge routers if necessary.

But existed edge-router based methods only take bandwidth into considerations. While packet flood may not only consumes bandwidth, but also the processing time of victim, and in some conditions the attacking flood consumes more processor time than bandwidth, e.g. S. Kumar [1] points out that ping flood mainly waste CPU cycles even it also waste bandwidth. As a result, we consider bandwidth and CPU at the same time. Actually, we take the dispatching of filtering messages as a dual-resource allocation problem and name it FFDRF (Feedback Filtering with Dual-Resource Fairness). While multi-resource allocation is a hot problem in AQM, we provide our revised algorithm which will be given in session IV. Through simulations in NS2, we find that even in a network of heterogeneous link condition and flows from heterogeneous source, FFDRF is stable and has fast convergence rate. And legitimate traffic has high enough survival ratio even in heavy attacks. Compared to level-*k* max-min fairness in [13], FFDRF is effective even in circumstance where level-k max-min fairness fails. At last, we implement FFDRF in a linux based router and find it is applicable even when up to thousands of filters are installed.

Funded by Zhejiang Provincial Science and Technology Program (2007C21034)

As stated above, FFDRF has the following advantages:

Deployment. All routers are inside one AS under single administration region. Different from earlier methods, ISPs have enough motivation to deploy. And not all hosts and routers need deploying, except edge routers, only vulnerable sites need to pay for protection.

Costs. Through analyzing internet topology, we find that most ASes have reasonable number of edge routers, which means their deployment costs could be restricted.

Effectiveness. Because inside one AS, the edge routers are the nodes nearest the attacking sources, their filtering the most effective. Additionally, FFDRF is able to defend several kinds of packet flood.

Expansibility. Filtering in one edge router equals to punishment to the corresponding neighbor AS which has not deployed FFDRF. So neighbor ASes have the motivations to deploy. And neighbor ASes could share edge routers to push filtering point closer to sources.

The rest of paper is organized as follows. In section II we discuss related work. FFDRF system model and related definitions are presented in section III. A detailed description of FFDRF algorithm is provided in section IV. Then we give simulations in section V to test FFDRF. Section VI shows the performance of FFDRF implementation. And a conclusion is given at last.

II. Related Work

There are some effective researches in distributed defense of DDoS. K. Wan et al consider deploying local detection system in routers of several Internet core ISPs, coordination between these systems lead to filtering attack traffic in the right incoming ports of right routers [2]. R. Mahajan et al [3] provide methods for filtering aggregate flow, and the defense router cooperates with upstream routers to improve the effectiveness of filtering.

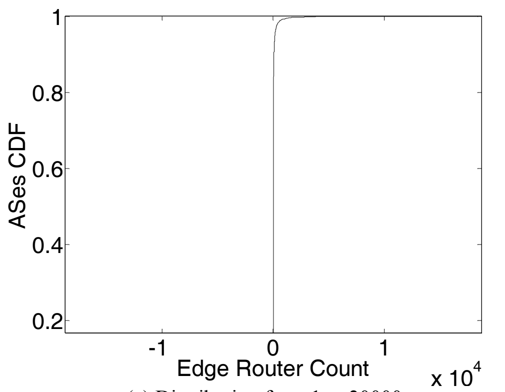
To mitigate DDos attack, much of current researches focus on distributed overlay network, A. Keromytis et al [4] present the conception of Secure Overlay Services (SOS) overlay network through which the legitimate traffic is sent. The SOS network is able to change overlay topology dynamically to avoid DDoS and can survive in case that some key nodes are attacked. D. Xuan et al improve SOS to counter intelligent attack aiming at the architecture of SOS itself [5]. A. Stavrou et al [6] present a stateless multipath SOS to make it harder to attack the SOS architecture.

Except SOS, path identification (PI) is also a feasible way to filter and trace back DDoS. A. Yaar et al [7] propose that one packet is marked in every router it passes and all the marks become path identification (Pi) when packet arrives at destination. The host can filter packets according to Pi. S. Savage et al suggest that router mark packets with a probability and when attacking traffic flood, the victim could get enough packet to construct attack path and trace back the attacking sources [8].

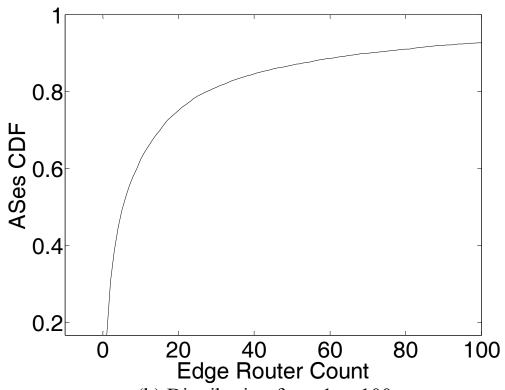
Additionally, capability is designed to prevent internet architecture from attack. X. Yang et al [9] propose that every sender should send request to destination for permission and get a certification called capability, and then the sender can added capability in packet header to send a certain bytes in a certain time range.

The above methods use a significant proportion of routers in Internet. And in some other methods only edge routers are needed. Z, Duan et al [10] pick out packets with faked source address in edge routers using BGP update information. K. Park et al [11] setup two filters in edge router, one only checks source address and the other check both the source and destination address.

Here we emphasize some defense methods combined with feedback control. S. Chen et al propose two defense mechanisms and related trace-back algorithm in edge routers. Their mechanisms can be incrementally deployed between ASes [12]. D. Yau et al [13] propose filtering deployed in level-k upstream router of victim and introduce max-min fairness in traffic control. They have also proved the stability and convergence of AIMD (Additive Increase Multiplicative Decrease) adjustment. Because above methods are only for bandwidth resource, they may fail in case of CPU-consuming attacks, e.g., ping flood and udp flood. And current QoS researches suggest taking both bandwidth and processing time into consideration [14, 15]. To improve defense effectiveness, we propose FFDRF.

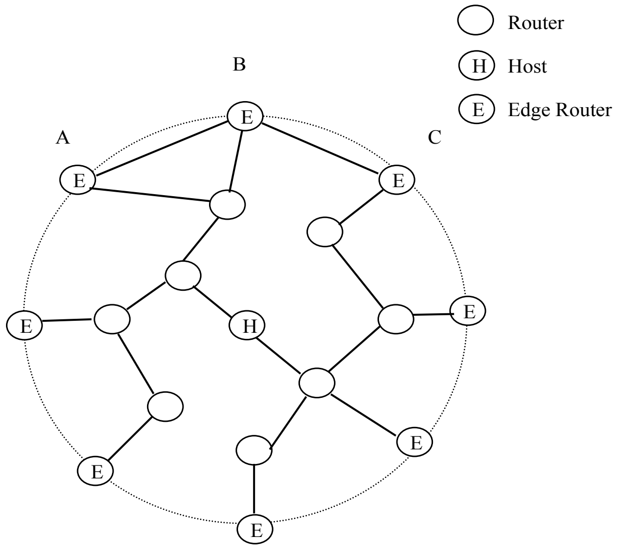


(a) Distribution from 1 to 20000



(b) Distribution from 1 to 100

Figure 1. The cumulative distribution of edge router count of ASes in internet



.Figure 2. The inner topology of AS

III. System Model

A. AS Topology

Edge route is routers that sit at the periphery edge of a network. Internet composes of AS regions and each of them are usually under the administration of single ISP (Internet Service Provider). In contrast with core routers that are in the middle of a network, edge routers are at the periphery of ASes and help to connect to other ASes. The advantages of edge routers are that they are under one single administration region.

Fig. 1 is the cumulative distribution of the count of ASes in internet which is calculated from CAIDA 2003 ITDK data [18]. (a) is the distribution with AS inside edge routers count from 1 to 20000 while (b) is the distribution from 1 to 100. We find ASes with 1 to 100 edge routers has summed up to 92% of all ASes in Internet. That indicates that edge router based defense is feasible for most of ASes.

And about the topology inside AS, there are some misunderstandings. S.chen et al [12] think nearly every host has only one upstream edge routers, but it may not be the case, e.g., host in Fig. 2 connects to several edge routers through several middle routers. In [13], because the fan out of one host is nearly always less than the count of edge routers, the packets from edge routers converge in some middle routers. However, different from the model in [13], where there is no link between level-*k* nodes, links exist among edge routers which has been proved by CAIDA data. The representative topology from one host to edge routers is in Fig. 2 where there is a link between edge routes B and C.

B. Assumptions of DDoS Attack

In DDoS, large amount of packets are sent to victim in a short time and most of legitimate packets are dropped because of the congestion in upstream links of victim. And it probably costs victim a lot of time to process attack packets, actually, sometimes even dropping flood packets can bring up victim's CPU utilization significantly. As a result, legitimate requests can not be served. Here we only think of attacks which do not exploit application vulnerabilities, like in [12] and [13].

And DDoS traffic may be a mixture of several kinds of packets. It means that the impact of DDoS may vary a lot and the defense architecture should response quickly. And at the same time, we only consider attacks from AS outside where most DDoS attacks come from.

C. Multi-Resource Max-Min Fairness

Max-min fairness is proposed by J. Jaffe [19] and the aim is to give an optimized bandwidth distribution to several flows sharing limited links. The contribution of [19] is the principle of fairness and algorithm to achieve it. But max-min fairness is only for homogeneous resources, as several other kind of max-min fairness, e.g. proportional fairness [20] and utility fairness [21]. The max-min fairness for heterogeneous resources is still an open problem, and its background is given as below.

With l flows and m kinds heterogeneous resources, set S_i ($0 \le i \le m$) as the number of instance of resource i and every flow use one or several instances of every kind of resources. Set $C=(C_{ij} \mid 0 \le i \le m,\ 0 \le j \le S_i)$ as the capacity matrix of these resources where C_{ij} is the capacity of instance j of resource i. Set $U=(U_{kij} \mid 0 \le k \le l,\ 0 \le i \le m,\ 0 \le j \le \max_i(S_i))$ as usage matrix of resources and if the flow k uses the instance j of resource i, $U_{kij}=1$, otherwise it is 0. Set $A=(A_{kij} \mid 0 \le k \le l,\ 0 \le i \le m,\ 0 \le j \le \max_i(S_i))$ as the allocation matrix and $A_{kij} \le C_{ij}$ is the amount of instance j of resource i allocated to flow k. Here a rational allocation is constrained as next:

$$\sum_{k=1}^{l}\sum_{i=1}^{m}\sum_{j=1}^{S_i} U_{kij}A_{kij} \le C_{ij} \tag{1}$$

In max-min fairness, $m=1$ and one flow has same requirement of bandwidth in all instances of link resource, so element of A can be written as A_k. Set $\mathrm{T_A}$ as the set of A that satisfied (1).Given U and C, the allocation A' meeting max-min fairness is:

$$A' = \arg\operatorname{asc}(\max\{\operatorname{asc}(A) \mid \forall A \in \mathrm{T_A}\}) \tag{2}$$

Here the function asc is to rearrange the elements of A in dictionary order and function max returns the biggest element of its parameters.

Max-min fairness can not be directly expanded to circumstance of multi-resource. There has been some researches on it. Y. Zhou et al [14] discuss the fairness with constraint that every resource is essential and has only one instance, namely, $\{S_i=1 \mid 0 \le i \le m\}$ in (1) and then the element of the allocation matrix A can be written as A_{ki} and the element of C is C_i. Here if the requirement of one resource does not decrease even the requirement for other resources increase, the resource is essential and you can think it as the basic resource in a network, such as bandwidth. Y. Zhou's fairness principle FERA (Fair Essential Resource Allocation) is based on basic fairness such as max-min, and the fair allocation of A under basic fairness F is in next:

$$A' = \{\mathrm{LNA}(A) \text{ satisfies } \mathrm{F} \mid \forall A \in \mathrm{T_A}\} \tag{3}$$

Here LNA(A) is the Largest Normalized Allocation vector of A, which is:

$$\mathrm{LNA}(A) = \{[D_k] \mid 1 \le k \le l, D_k = \max_i(A_{ki}/C_i)\} \tag{4}$$

But the A' in (3) may not be unique and Y. Zhou et al discussed the precondition of basic fairness F which leads to a unique A'. To be practical, they provided an algorithm used for fairness with single CPU and single link and testified that the algorithm satisfy FERA. .

In the meantime, M. Shin et al [15] give out DRQ, an approximated proportional fairness algorithm for a shared processor and a shared link. They calculate a fixed processing density for every flow involved. Here one flow's processing density is the average number of CPU cycles required per bit of this flow.

Through comparison between above multi-resource fairness schemes, Y. Zhou's principle FERA is suitable for our defense algorithm, and the algorithm will be given in section IV.

IV. ALGORITHM: FFDRF

Our aim is to use max-min fairness to coordinate throttling of attack traffic in edge routers. Our fairness also aims to allocate one processor and one link to several flows. Here the link resource is the incoming link of the victim where attack packets cause congestion and the processor resource is the processor of victim. And single flow is the aggregate which is from AS outside and destined for the victim and transits one edge router. If one flow enters edge router from a port which connects to a router outside the AS, we consider the flow is from AS outside. So, the coordination of throttling could be solved as a problem of fairness among those flows sharing one processor and one link. To throttle effectively, we must get the resources consumption rate of every flow. However, processor consumption rate is hard to measure in edge routers because the actual consumption of a flow is unknown until it is processed by the destined host, except some particular circumstances, e.g., in work of [14] the processing consumption of one packet can be precisely measured in routers. In [17], P. Pappu et al consider two kinds of router application: header-processing applications and payload-processing applications. And for four given sub-kind applications, the processing time of one packet is a fixed value. But the problem is sophisticated here because the processing time per packet in victim is uncertain and related to specific application circumstance. For DDoS, there are mainly 3 kinds of attack packets: TCP, UDP, and ICMP. When the victim receives ICMP attack packets, the reaction is to drop or reply an echo. Similarly, when the victim receives TCP or UDP attack packets, the victim's reaction is dropping or reply an error message. Since for every TCP packet, the victim will search the connection table, so the processing cost of attacking TCP packets may be much larger than that of UDP and ICMP packets while the actual cost is uncertain and related to the size of connection table. So we think it is hard to give a certain processor cost for different attack packets as in [17]. Besides, we do not need the actual cost of one packet but one measurement for FFDRF to balance the CPU cost from different edge routers. If we can identify attack packets, we would rather set their CPU cost a large value to block them and set legitimate packet‘s cost a small value to encourage the acceptance. However, we do not identify attack packets from legitimate packets in FFDRF. Because the legitimate packets do not have absolute more CPU cost than attack packets, e.g. a legitimate http request may need lots of computing in web server. It is feasible to set every packet's CPU cost the same value and we consider this kind of value will help defense effectively and will illustrate it in our simulations.

Given the measurement above, we proposed our FFDRF (Feedback Filtering with Dual-Resource Fairness) algorithm based on Y. Zhou‘s FERA principle. The system using FFDRF functions as a control system: the incoming packet rate of edge routers is input and ingress traffic rate of victim is the output. Victim is the controller and periodically checks ingress rate and sends feedback to edge routers.

Because edge routers may be involved in several DDoS defense, the algorithm in edge routers should be stateless. In FFDRF, the edge routers' function is just to get filter message from victim and install or uninstall filter or change the filtering threshold. The filter message R‘s structure is as below:

$$R = (t, b, d) \tag{5}$$

Here t is the type of R, can be INSTALL or UNINSTALL that tell an edge router to install or uninstall filter related to victim. If t is INSTALL, b and d are respectively the byte rate and packet arrival rate that is permitted through. And if t is UNINSTALL, just uninstall the filter.

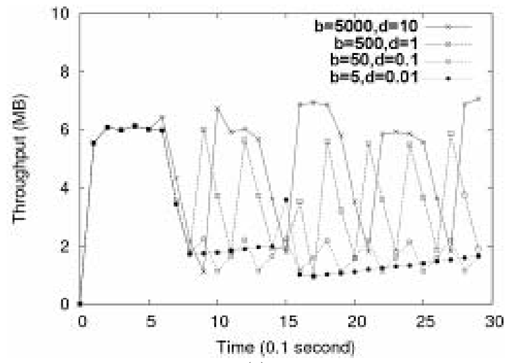
In the mean time, victim periodically check the incoming rate and CPU utilization and feedback correspondingly. If average flow rate v is out of boundary $[L_v,U_v]$ or CPU utilization c is out of boundary $[L_c,U_c]$, the throttle value b or p is adjusted and sent to every edge routers. We use AIMD (Additive Increase Multiplicative Decrease) to adjust the filtering value, i.e., increase filter value by adding a step value and decrease it by dividing 2. The algorithm is given in Fig.3.

```
Procedure FFDRF_allocate{
        Groupcast initial R to edge routers;
Do{
        Get incoming traffic rate v and average CPU utilization rate c;
Rule 1:
        if($v < L_v$ and $c < L_c$) then {
                        if(attack stopped) then set message type as UNINSTALL
                        else increase b and d
        }
Rule 2:
        If( ($U_v \geq v \geq L_v$ and $c < L_c$ ) or ($U_c \geq c \geq L_c$ and $v < L_v$)) then {
                Check which of v and c is throttling bottleneck and increase it
        }
Rule 3:
        If( ($v > U_v$ and $c \leq U_c$ ) or ( $c > U_c$ and $v \leq U_v$)) then {
                If($v > U_v$) decrease throttling threshold b
                If($c > U_c$) decrease throttling threshold d
        }
Rule 4:
        If($v > U_v$ and $c > U_c$) then {
                Decrease b and d
        }
Rule 5:
        If ($U_v \geq v \geq L_v$ and $U_c \geq c \geq L_c$) then {
                Do nothing;
                                }
        Groupcast R to edge routers;
}While(1);
```

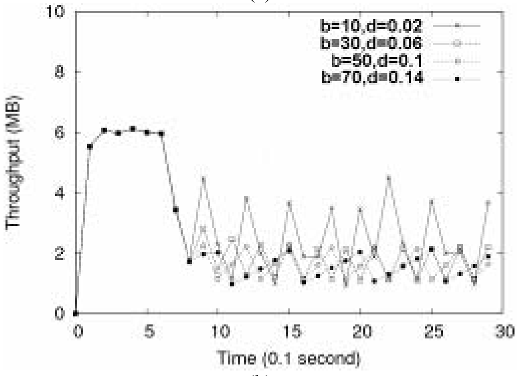
Figure 3. The FFDRF algorithm.

V. SIMULATIONS

We use the topology of AS 15412 from CAIDA ITDK data [18] and there are 134 edge routers of 148 routers. Because FFDRF bases on periodical feedback of incoming traffic, its parameters and incoming traffic fluctuation will impact its stability and convergence. We will check it at first. Then the effect of FFDRF under different attack scenario is given. At last we compare FFDRF with level-k max-min fairness in [13]. The default traffic rate is in units of KB/S and default packet arrival rate is in units of one thousand packets/S.

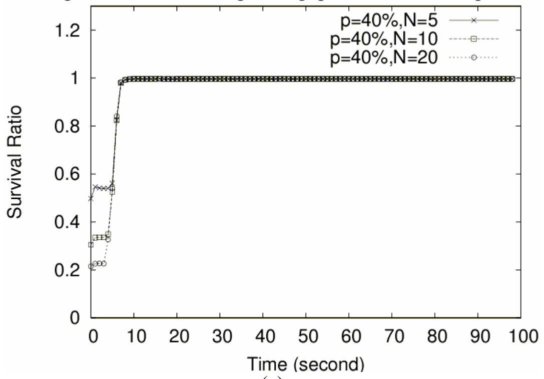


(a)

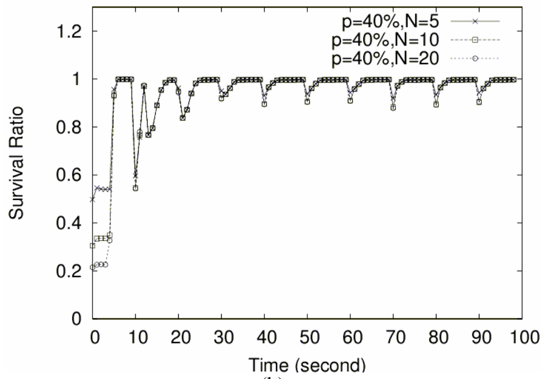


(b)

Figure 4. The incoming throughput in different steps.



(a)



(b)

Figure 5. The legitimate survival ratio in simulation 1.

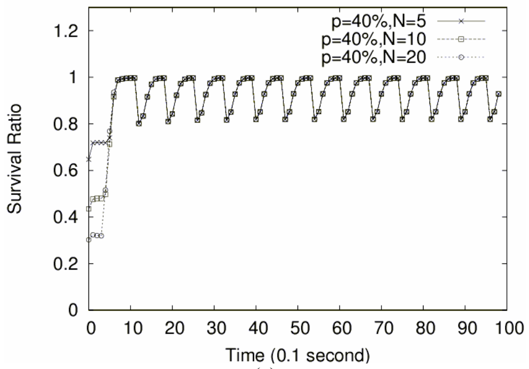
A. Stability and Convergence

(1)Stability

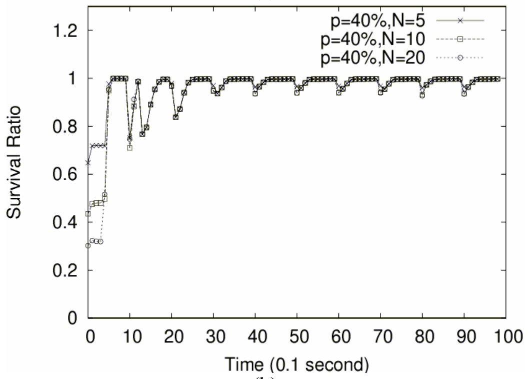
120 attackers outside AS send flood to the victim. 80 of them send constant flows with rate of 50 and 20 of them send flows with Pareto distribution (both on and off interval are 500 ms, on rate is 100 and off rate is 50). The final 20 attackers send synchronized square-pulse flow with low rate of 50, high rate of 100 and half period of 5 seconds. The average packet length is 500 bytes and the incoming bandwidth of victim is 10000. Here we consider four kinds of steps: b=5000 and d=10, b=500 and d=1, b=50 and d=0.1, b=5 and d=0.01. The bound is L_v=65000,U_v=75000 with L_c=130,U_c=150. The result is shown in Fig. 4(a). We can see that the incoming rate is unstable in big step and is stable in small step.

(2)Convergence

Here we use four steps close to each other: b=10 and d= 0.02, b=30 and d=0.06, b=50 and d=0.1, b=70 and d=0.14. The bound is L_v=68000, U_v=72000 with L_c=136, U_c=144. Fig. 4(b) shows that the step should neither be too small, otherwise, the incoming rate is stable enough but may not utilize bandwidth well, and more important, my not converge quickly. So a medium step may be a good choice. In the simulation a good choice is b=50 and d=0.1.



(a)



(b)

Figure 6. The legitimate survival ratio in simulation 2.

B. Legitimate survival ratio

Here attack agents outside AS initiate a distributed flood to one host inside AS and the onset lasts for 100 seconds. In different attack scenarios, CPU or bandwidth is the bottleneck (the resource is jammed firstly) or both are bottlenecks. Furthermore, the ratio of the average CPU cost between legitimate and attack packets may be larger or less than or equal to 1. We give five corresponding simulations.

(1) Simulation 1: The CPU cost ratio is larger than 1 and bandwidth is the bottleneck

The incoming rate of victim is 100000 and average packet size is 500 bytes. For legitimate traffic, the rate is between 50 and 100 and the CPU consumption of single packet is randomly between 1 CPU cycle and 200 cycles. The attack rate is between 50*N* and 100*N* (N is the multiples) and CPU consumption is between 1 and 100 cycles. The peak processing capacity of victim is 2500000 cycles per second. The control bound in victim is between 2400000 cycles and 2450000 cycles with byte rate of 96000 and 99200, i.e., two resources would be jammed and bandwidth is exhausted at first. And attacking traffic come through 40% edge routers where the ratio is represented as p. The simulation results with p=40% and N=5,10 and 20 are plotted in Fig.5 where (a) is constant rate attack and (b) is variable rate attack. In variable rate attack, byte rate of attacking traffic randomly changed inside the bound every 10 seconds.

(2) Simulation 2: The CPU cost ratio is larger than 1 and CPU is the bottleneck

Here the simulation scenario is a little different from simulation 1, i.e. for legitimate traffic, the average packet size is 500 bytes and for the attack traffic, the average packet size is 100 bytes, as a result, when traffic come from edge routers at full speed, CPU is jammed at first. The simulation results are plotted in Fig.6 where (a) is constant rate attack and (b) is variable rate attack.

(3) Simulation 3: The CPU cost ratio is less than 1 and bandwidth is the bottleneck

The incoming rate of victim is 100000 and average packet size is 500 bytes. For legitimate traffic, the rate is between 50 and 100 and the CPU consumption of single packet is between 1 CPU cycle and 50 cycles. The attack rate is between 50*N* and 100*N* and CPU consumption is between 1 and 100 cycles. The peak processing capacity of victim is 2500000 cycles per second. The control bound in victim is between 2400000 cycles and 2450000 cycles and 96000 and 99200. The simulation results are plotted in Fig.7 where (a) is constant rate attack and (b) is variable rate attack.

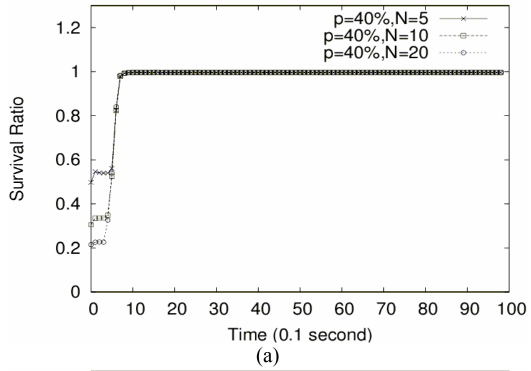
(4) Simulation 4: The CPU cost ratio is less than 1 and CPU is the bottleneck

For legitimate traffic, the average packet size is 500 bytes and for the attack traffic, the average packet size is 100 bytes. And the rest is the same to simulation 3. The simulation results are plotted in Fig.8 where (a) is constant rate attack and (b) is variable rate attack.

In above 4 simulations, we can see that no matter how attack traffic changes, as legitimate traffic is not aggressive, the survival ratio can reach up to 90% around.

C. *Legitimate survival ratio in meek DDoS attack*

The filter can effectively depress attack flows with high rate. But when one attack consists of many low rate flows which act like legitimate flow, FFDRF may not work very well because it punishes legitimate and attack flows equally. We setup a simulation to test the effect under meek attack. Here the incoming bandwidth of victim is 50000 and average packet size is 500 bytes. The CPU of victim could process at most 12500 packets per second. The attack rate is the same to legitimate



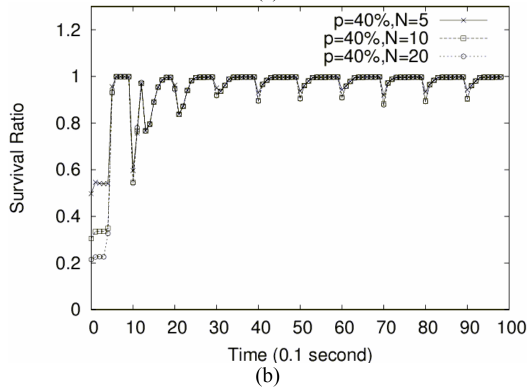
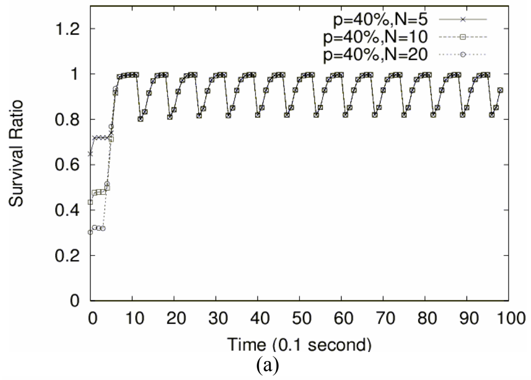


Figure 7. The legitimate survival ratio in simulation 3.



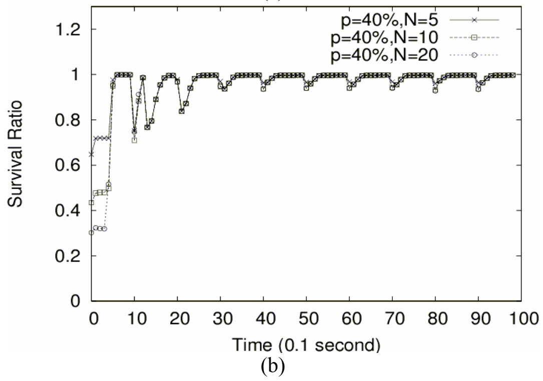


Figure 8. The legitimate survival ratio in simulation 4.

rate,is between 50 and 100. Fig 9 is the survival ratio when p=20% and 40%. We could see that the survival ratio drops to a low level and similar situation happened in level-k max-min. However, the victim is at least in service, not overwhelmed by attack traffic.

D. Comparison between FFDRF and level-k max-min

Because level-k max-min only considers bandwidth flood, when attack traffic mainly consume processor time, such as, ping flood [1], the level k max-min would leave victim dropping packets. To compare FFDRF and level-*k* max-min, the simulation scenario is as below: The incoming bandwidth of victim is 100000 and average size of legitimate packet and attack packet are 500 bytes and 100 bytes. The CPU of victim could process at most 25000 packets per second. The legitimate

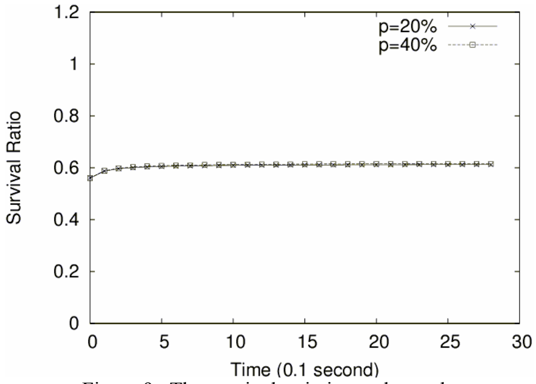
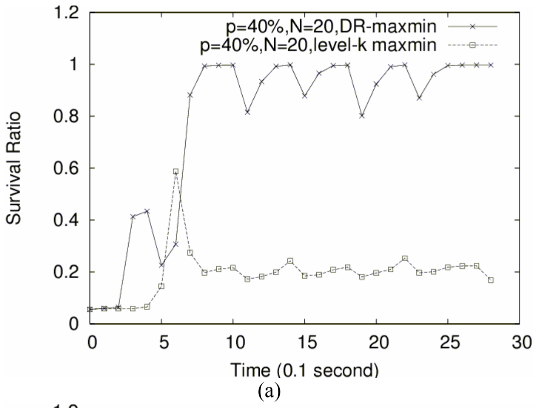


Figure 9. The survival ratio in meek attack.



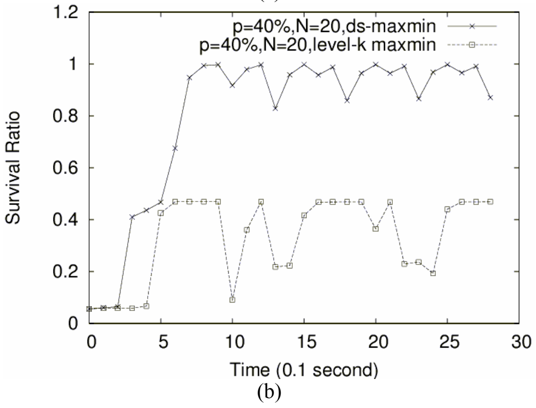


Figure 10. The comparison between level-k max-min and FFDRF.

is between 50 and 100 and the attack rate is between 50*N* and 100*N*. In FFDRF the bound is L_v=96000, U_v=100000 and L_c=24000, U_c=25000. And in level-*k* maxmin it is just [96000, 100000]. Fig.10 shows the comparison of survival ratio while in (a) *p*=40%, *N*=20 and attack rate is constant and in (b) *p*=40%, *N*=20 and attack rate is variable. We can see a clear difference in result.

VI. IMPLEMENTATION

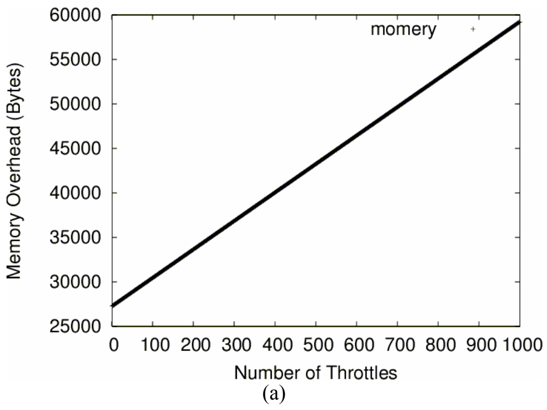
We implement FFDRF in a linux router. The router is implemented in a Sun Server with two 2.0GHz CPUs and 2GB memory. An application level daemon routes incoming packets and FFDRF filter is implemented as a kernel module. When FFDRF filter requests arrive, routing daemon reads the parameters and sends it to FFDRF filter module to install or uninstall a filter.

To test the feasibility of FFDRF, we firstly test the memory overhead of up to 1000 filters are installed, which is shown in Fig. 11(a). We can see memory overhead is linear to the filter count and could still keep low when thousands of filters are installed.

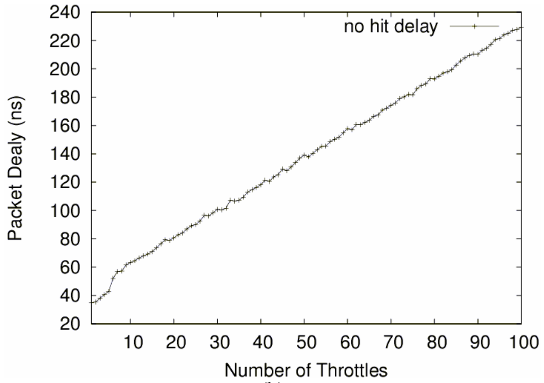
In the other hand, we test the processing time per packet and total throughput with different number of filters installed. When one packet arrives, the filter list is searched linearly. Here we give tests in two situations. One is that the filter list is searched through and no matched filter (no hit), another is the matched filter is in the tail of list (last hit). The results are given in (b) (c) (d) (e) of Fig. 11. It is shown that the simplest implementation of FFDRF could have high throughput and low processing delay.

VII. CONCLUSIONS

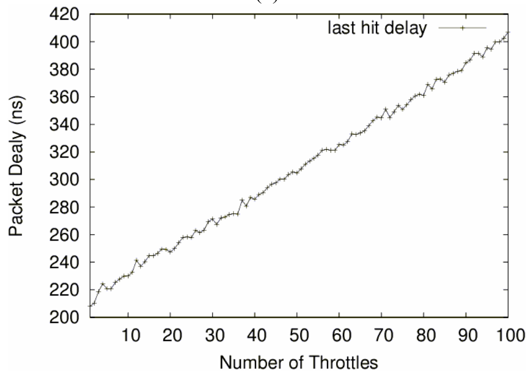
In the paper, we present a new countermeasure against DDoS flood. Here the defense is considered as a resources allocation problem. Compared to earlier work [13], our defense mechanism takes both bandwidth and processor time into considerations. Based on a newly proposed principle, we present our max-min fairness algorithm FFDRF. To implement the fairness filter, only edge routers is needed. The simulation results show that it is effective against common constant and variable rate attacks and have advantages over



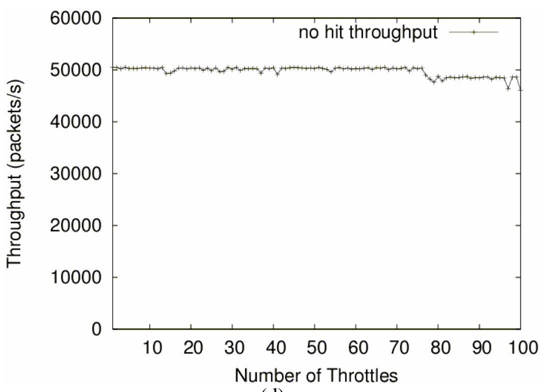
existed single resource method. The implementation in a linux router testifies the feasibility.



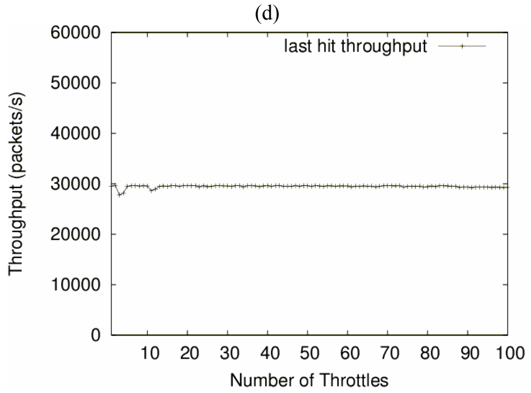
(b)



(c)



(d)



(e)

Figure 11. The performance of FFDRF implementation.

But like existed methods, our mechanism is less effective in meek DDoS attack. And attack from AS inside can not be effectively filtered. We will improve it in the future.

REFERENCES

[1] S. Kuman. PING attack – How bad is it? Computers & Security, Jan 2006.

[2] K.K.K. Wan, R.K.C. Chang. Engineering of a global defense infrastructure for ddos attacks. In proceedings of 10th IEEE international conference on networks (ICON 2002), pp. 419-427, 2002.

[3] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker. Controlling high bandwidth aggregates in the network. ACM SIGCOMM Computer Communication Review, vol. 32, no. 3, July 2002.

[4] A.D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. ACM SIGCOMM Computer Communication Review, vol. 32, no. 4, Aug 2002.

[5] D. Xuan, S. Chellappan, X. Wang, S. Wang. 25. Analyzing the secure overlay architecture under intelligent ddos attacks. In proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), 2004

[6] A. Stavrou, A. D. Keromytis. Intrusion detection and prevention: Countering DoS attacks with stateless multipath overlays. In proceedings of the 12th ACM conference on Computer and communications security (CCS '05), 2005.

[7] A. Yaar, A. Perrig, D. Song. Pi: a path identification mechanism to defend against ddos attacks. In proceedings of Symposium on Security and Privacy, pp. 93-107, May 2003.

[8] S. Savage, D. Wetherall, A. Karlin, T. Anderson. Practical Network Support For IP Traceback. In proceedings of ACM Sigcomm 2000.

[9] X.W. Yang, D. Wetherall, T. Anderson. A DoS limiting Network Architecture. Computer Communication Review, vol. 35, no. 4, p241-252, oct 2005.

[10] Z. Duan, X. Yuan, J. Chandrashekar. Constructing Inter-Domain Packet Filters to Control IP Spoofing Based on BGP Updates. In proceedings of IEEE Infocom, 2006.

[11] K. Park, H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. ACM SIGCOMM Computer Communication Review, vol. 31, no. 4, 2001.

[12] S. Chen, Q. Song. Perimeter-Based Defense against High Bandwidth DDoS Attacks. IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 6, June 2005.

[13] D. K. Y. Yau, J. C. S. Lui, F. Liang, Y. Yam. Defending Against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles. IEEE/ACM Transactions on Networking, vol. 13, no. 1, Feb 2005.

[14] Y. Zhou, H. Sethu. On Achieving Fairness in the Joint Allocation of Processing and Bandwidth Resources: Principles and Algorithms. IEEE IEEE/ACM Transactions on Networking, vol. 13, no. 5, Oct 2005.

[15] M. Shin, S. Chong. Dual-resource TCP/AQM for processing-constrained networks. In proceedings of IEEE Infocom, 2006.

[16] T. Wolf and M. A. Franklin. CommBench - a telecommunications benchmark for network processors. In proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), p154–162, Apr 2000.

[17] P. Pappu and T. Wolf. Scheduling processing resources in programmable routers. In proceedings of IEEE Infocom, pp.104–112, June 2002.

[18] http://www.caida.org/home/

[19] J.Jaffe. Bottleneck Flow Control. IEEE Transactions on Communications. vol. 29, no. 7, pp. 954-962, July 1981.

[20] F. Kelly. Charging and rate control for elastic traffic. Europen Transactions on.Telecommunications, vol. 8, no. 1, pp. 33–37, Jan 1997.

[21] Z. Cao, E.W. Zegura. Utility max-min: an application-oriented bandwidth allocation scheme. In proceedings of IEEE Infocom, 1999.