Real-time detection and containment of network attacks using QoS Regulation

Seong Soo Kim and A. L. Narasimha Reddy
Department of Electrical Engineering
Texas A&M University
College Station, TX 77843-3128, USA
{skim, reddy}@ee.tamu.edu

Abstract–**In this paper, we present a network measurement mechanism that can detect and mitigate attacks and anomalous traffic in real-time using QoS regulation. The detection method rapidly pursues the dynamics of the network on the basis of correlation properties of the network protocols. By observing the proportion occupied by each traffic protocol and correlating it to that of previous states of traffic, it can be possible to determine whether the current traffic is behaving normally. When abnormalities are detected, our mechanism allows aggregated resource regulation of each protocol's traffic. The trace-driven results show that the rate-based regulation of traffic characterized by protocol classes is a feasible vehicle for mitigating the impact of network attacks on end servers.**

Keywords–**Network measurement, Experimentation with real networks, Denial of service, Attack detection, Quality of service, Weighted Fair Queuing, Traffic Control for QoS.**

I. INTRODUCTION

Many kinds of DoS (Denial of Service) attacks and worm based attacks have been recently staged on popular servers in the Internet. These network attacks prevent the servers from providing valuable services to the legitimate users. Furthermore, in severe situations, the attacked servers may be brought down by these attacks. Therefore, the necessity of protecting the servers has increased and much recent work has focused on mitigating or thwarting such DOS attacks. One of the feasible solutions is the class-based buffer management proposed in [1]. In this approach, instead of identifying the attacks, the resources that need protection are identified. When DoS attacks are staged over a single protocol, the traffic volume of that protocol significantly increases from normal traffic. As a result, it seems possible to provide protocol-based buffering and regulation of traffic in order to control the attack traffic. Rate control as in [8], window control and weighted fair queuing can play a significant role in protecting servers from anomalous traffic in this way.

However, class-based buffer management requires more processing time and resource consumption than non class-based buffer management because it is necessary to parse every packet header from the input traffic, to assign these packets to designated buffers that are classified by protocols. Moreover, if predefined fixed rates are employed to regulate the different protocols, the dynamics of the traffic cannot be flexibly managed. Hence, we take a two-part approach in our work presented here. During normal traffic, we monitor the volume of traffic based on protocols. If the volume of traffic for a particular protocol changes abnormally, our scheme switches to a class-based regulation mechanism in order to control the attack traffic volume.

II. OUR APPROACH

2.1 The Nature of Network Attacks

Typical network attacks are the TCP SYN flood, ICMP directed broadcast, Smurf, Ping of Death [3, 4, 6, 7]. It has been shown that more than 90% of the DoS attacks use TCP [2]. However, various kinds of traffic attacks that use the vulnerability of different protocols are often observed. Table I summarizes the relationship between various recent network attacks and exploited protocols. We consider an approach based on observing the composition of different protocols in the network traffic. This is based on the observation that during the attacks, the protocol employed by the attack traffic should see considerably more traffic than during normal traffic. For example, the recent SQL Slammer worm infected over 90% of the vulnerable hosts employing UDP protocol [7]. In this approach, we monitor the amount of ICMP traffic $i(t)$, TCP traffic $t(t)$, UDP traffic $u(t)$ and ETC. traffic $e(t)$ as a fraction of the total traffic volume at each sampling interval. Protocol composition has $O(1)$ cost per packet and $O(n)$ storage cost per sample, where n is the number of protocols monitored.

2.2 Our Approach

Our approach first detects anomalies in traffic and when an attack is detected, resource management techniques are

TABLE I
TYPICAL ATTACKS AND THEIR PROTOCOLS

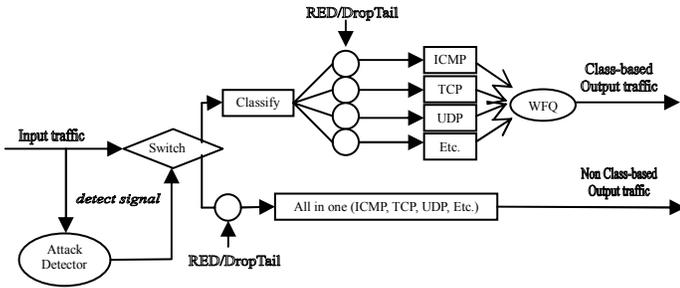| Protocol	Anomalies and Attacks
TCP	TCP SYN Flooding
	ACK Scan
	Telnet Scan
	TCP session hijacking (Hunt, Juggernaut)
	WinNuke
	Christmas Tree
	Code Red
UDP	Echo-Chargen
	Trin00
	Nimda
	SQL Slammer
ICMP	Smurf
	ICMP echo reply
	Ping of Death
	RingZero
	TFN (Tribe Flood Network)
	WinFreeze
	Loki

Figure 1. The structure of our flexible buffer management

employed to effectively control the system. Fig. 1 illustrates the structure of our mechanism. Most of the traffic attacks send malicious packets to victims using a specific protocol and also they are designed to consume specific network resources like network bandwidth, protocol state buffer, kernel interrupt processing cycles, memory, or CPU cycles. When a specific DoS attack is staged, the proportion of the specific protocol exploited increases abruptly. Therefore, we are able to design an attack detector by monitoring the variation of the input traffic by each protocol. We can use the detection signal to switch to a different resource management technique to control the detected anomalous traffic. For example, we could employ buffer management techniques to control the traffic of the protocol employed by the attack traffic. We can switch to a class-based buffer management when network attack is detected and we can switch back to normal (non class-based) buffer management when no anomalies are present. Such a dual-mode approach potentially allows us to provide more efficient packet handling during normal traffic as the processing required for class-based resource management is only necessary during the periods of attack.

III. Implementation

3.1 Traces and Attacks

To evaluate the validity of our approach, we run our method on 2 kinds of network traces. First, to investigate on backbone links, we examine the tool on KREONet2 traces from Oct. 12, 2003 to Oct. 26, 2003 which contain actual worm attacks [9]. In the traces employed here, there are 5 major attacks as described in Table II and a few instantaneous probe attacks. Additionally, to inspect the sensitivity of our tool on various configurations, we evaluate our method on the NLANR (National Laboratory for Applied Network Research) traces [5]. These traces were anonymized, but preserved the protocol numbers.

3.2 The Weighted Fair Queuing

Recent studies have shown that the traffic can have strong

TABLE II
THE DESCRIPTIONS OF FIVE ATTACKS IN KREONET2 TRACES

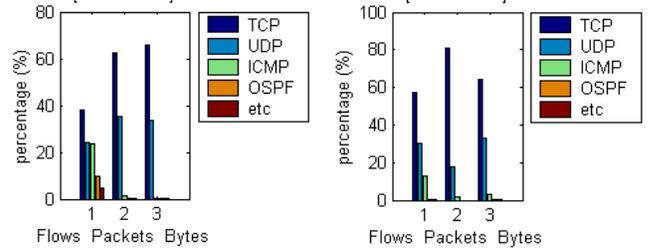|	1	2	3	4	5
Duration	5.3 h	4.5 h	4.1 hours	12.3 h	3.6 h
IP	semi-random	random[a]	random	semi-random random	random
Protocol	TCP	UDP	TCP/UDP	TCP/UDP/ICMP	UDP
Port	#80	#1434	random/#1434	#80 / #1434 / #0	#1434
Size	48B	404B	random/ 404B	48B / 404B/ 28B	404B

a. SQL Slammer

Figure 2. The proportion of major protocols over two traffic traces

patterns of behavior over several timescales [10], and our previous work has shown the possibility of analysis of wide-sense stationary (WSS) property in network traffic [11]. By observing the traffic and correlating it to the previous states of traffic, it may be possible to see whether the current traffic is behaving in an anomalous manner.

For recognizing the proportions of traffic by protocol, we observed the protocol ratios of the real traces. Fig. 2 shows traffic-volume ratios by each protocol over two different NLANR traces. The number of flows, packet counts and byte numbers of a specific traffic are closely related to each other. The descending order of protocol proportion is TCP, UDP, ICMP, OSPF and etc.

For the class-based buffer management, we use WFQ (Weighted Fair Queuing). We classify the protocol in 4 classes as ICMP, TCP, UDP and etc. Based on the proportion of each protocol in the KREONet2 traces during normal time, we can set weights for each class (protocol). Initially we set weights of 4:71:24:1 respectively for ICMP, TCP, UDP, and etc. Moreover, these weights are adjustable over time according to the input traffic volume.

3.3 Thresholds

We employ 2 kinds of thresholds, which are a high threshold T_H indicating that the fraction of volume of one of the network protocols increased abnormally and a low threshold T_L indicating that the fraction of the traffic volume of the network protocol decreased inordinately. We set the high threshold (T_H) for detecting abnormal increase as (1).

$$T_H(p,t) = 1 / \sqrt{r(p,t)} \qquad (1)$$

, where p is individual protocol

$r(p,t)$ is the current proportion of the protocol

The threshold (T_L) detects decreases in traffic of that protocol. Because the proportion of each protocol in traffic is closely interrelated to each other, the increase of the proportion of one protocol makes the proportions of other protocols to decrease. By using this reciprocal property we are able to detect anomalies of other protocols. For example, when we observe the abrupt decrease of proportion of traffic other than TCP, we are able to notice TCP-based traffic anomalies. We set the low thresholds by inversing high thresholds as $T_L(p,t) = 1/T_H(p,t)$.

When each protocol proportion in current input traffic is larger (or lower) than the product of average proportion and the high (or low) threshold for individual protocol as in (2), the detector declares anomalies. It indicates that the balance of traffic by protocol changed significantly.

$$\text{If } r(p,t) \begin{cases} > MA(p,t-1)*T_H(p,t) \\ < MA(p,t-1)*T_L(p,t) \end{cases}, \Lambda(p,t) \text{ is attack} \quad (2)$$

Otherwise $\qquad\qquad\qquad\qquad, \Lambda(p,t)$ is normal

The higher the protocol's proportion of normal traffic is, the lower the setting for its high threshold is. This improves sensitivity. Initially the high thresholds are 5.0: 1.2: 2.0: 10.0 for ICMP, TCP, UDP, etc. The reason why the high threshold for ICMP is set high is that ICMP traffic is very low at normal time, so that a little increase of ICMP traffic can even cause false alarm if we set the detection threshold for ICMP to be low. Therefore, we set the T_H threshold for ICMP much higher than the other protocols. Similarly, the threshold for TCP is the lowest among them. TCP traffic is continuously high at normal time and it occupies most of bandwidth. Therefore, if we set the detection threshold for TCP high, we may not be able to detect network attacks even though they may still exist. As a result, we set the detection threshold for TCP much lower than those for other protocols for higher sensitivity. And we initially set the low thresholds as 1/5: 1/1.2: 1/2: 1/10.

3.4 Exponential Weighted Moving Average

We employ exponential weighted moving averages for accommodating the dynamics of traffic. By using moving averages of each protocol proportion, we can filter out shorter-term noises. We can define the moving-average time series as follows.

$$MA(p,t) = \alpha \cdot r(p,t) + (1-\alpha) \cdot MA(p,t-1) \quad (3)$$
$$\text{, where } MA(\cdot,1) = r(\cdot,1)$$

Currently, we set a forgetting factor α to 0.1 for this experiment. It is worthy to note that it is not reasonable to calculate the moving average at all times because the moving average becomes abnormally high during the long duration of network attacks. It degrades the correctness of network attack detector. Therefore, we use only the values that are measured during the absence of network attacks for moving averages. In particular, we do not calculate moving average during traffic attacks. In this way, we can preserve the sensitivity of network attack detector.
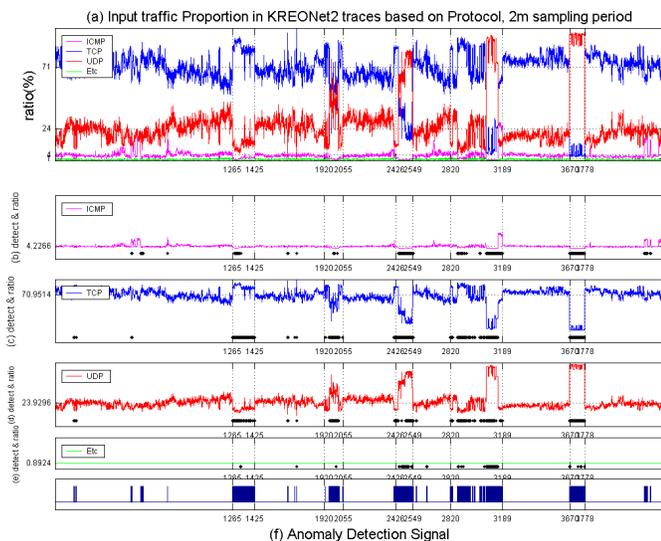

Figure 3. Input traffic propotion by protocol and detection signal

3.5 Operation Modes

There are two modes of operation used in our design. One is class-based buffer management and the other one is non class-based buffer management. For the class-based buffer management, we design the buffers to be operated by RED (Random Early Detection) or Drop-Tail with weighted round robin to de-queue packets. For the non class-based, we used FCFS (First-Come First-Served) scheme to de-queue packets instead of weighted round robin.

IV. EXPERIMENT & DISCUSSION

4.1 Input Traffic By Protocol and Detection

Fig. 3 shows the input traffic proportions of each protocol with attacks. Fig. 3(a) sub-picture shows the proportions of all protocols and remnant sub-pictures show the proportion of each protocol individually. The vertical lines show the five salient network attack periods and the black dots at the bottom of Fig. 3(b) through 3(e) sub-pictures mean that the traffic attacks are detected independently.

Fig. 3(b) shows the variation of ICMP proportion signal $i(t)$ with time. ICMP takes about 4.2% of total proportion at normal state. However, the 4th ICMP-based attack between the 3158th and the 3189th sampling period makes ICMP proportion increase up to 29.2% abruptly. Fig. 3(c) shows the variation of TCP proportion $t(t)$ with time. TCP occupies about 71% of the total at normal time. The traffic in 1st, 3rd and 4th attack cases show a little variation in TCP traffic since TCP already takes a high portion of the total traffic. Even though TCP-based attacks do not show a significant change in TCP traffic, we can detect the attack indirectly. Fig. 3(d) shows the UDP proportion $u(t)$. It shows the UDP-based SQL Slammer worm attack in the 2nd, 3rd, 4th, and 5th cases. Especially, in the 5th attack case between the 3670th and 3778th period has a protocol proportion of 0.4: 3.6: 95.9: 0.1 for ICMP: TCP: UDP: Etc. respectively, as the statistical median. Furthermore, it detects all TCP-based attacks by using a low detection threshold for TCP. The proportion of each protocol is closely interrelated. Therefore, if TCP takes a higher portion of the traffic, other protocols correspondingly take a smaller proportion. For example, the 1st attack case between the 1265th and 1425th period has a protocol proportion of 2.1: 85.5: 12.2: 0.3 as the median. In this way, we can detect the anomalous attacks of other protocols. That is why we exploit the lower detection threshold as well. Fig. 3(e) shows that the other protocols $e(t)$ occupied about 1% of network traffic.

4.1.1 Evaluation of Anomaly Detection Performance

Fig. 3(f) shows the generated attack signal which is combined (logical OR) from the attack detection signals of each protocol. To reduce false alarms due to instantaneous changes in one protocol, we could use a majority vote over the detection signal of each protocol to detect anomalies. The detection signal is used to control the switching of modes for back-end buffer management. When network attacks are detected, our scheme switches the buffer management mode to class-based queuing. Through traffic engineering on two different traces, we evaluate that our approach has a detection rate of about 89.8% (detects 702 out of 782 suspicious symptoms) to 92.5%, and a false alarm rate of about 0.48% to

2.30% (82 false alarms in 3563 sampling periods) with *one out of four* majority detection as shown in Table III. This shows that protocol composition could be a useful signal.

Table III. Results of protocol composition signals

majority	T.P. β [1]	F.P. α [2]	LR [3]	NLR [4]
1 out of 4	92.5% 767/829	0.48% 17/3516	191.4	0.08
2 out of 4	80.1% 664/829	0.17% 6/3516	455.2	0.20
1 out of 4	89.8% 702/782	2.30% 82/3563	39.0	0.10
2 out of 4	82.4% 644/782	0.73% 26/3563	112.9	0.18

1. True Positive rate
2. False Positive rate
3. Likelihood Ratio by β/α, ideally it is infinity
4. Negative Likelihood Ratio by $1-\beta/1-\alpha$, ideally it is zero

4.2 The Non-class based Scheduling

Fig. 4 shows the output traffic proportion when we use FCFS (First-Come First-Served) and Drop-Tail as a buffer management. This is a non class-based scheme. For the second sub-picture of ICMP, when the 4th ICMP-based attack is staged, we can observe abrupt increase of ICMP output traffic that is much higher than regular 4%. This is natural because we do not provide any specific regulation for anomalous traffic mode. As a result, the ICMP-based anomaly results in high output ICMP traffic.

On the other hand, during the 1st, 3rd and 4th attacks, we can notice that the proportion of the TCP protocol increases. Moreover, we can observe the abrupt decrease of other protocols' proportions by reaction. On the other hand, when we inspect UDP, it remains about 24% on average. However, it shows abrupt increase of the proportion during the 2nd, 3rd, 4th, and 5th attacks with SQL Slammer worms. While anomalies are staged by UDP or ICMP, TCP undergoes corresponding decrease in the output rate.

In general, when a specific protocol-based attack happens, it results in the abrupt increase of the proportion of the
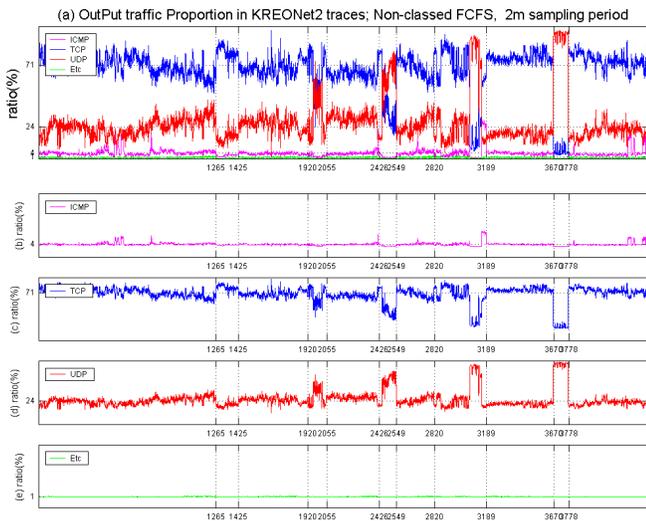

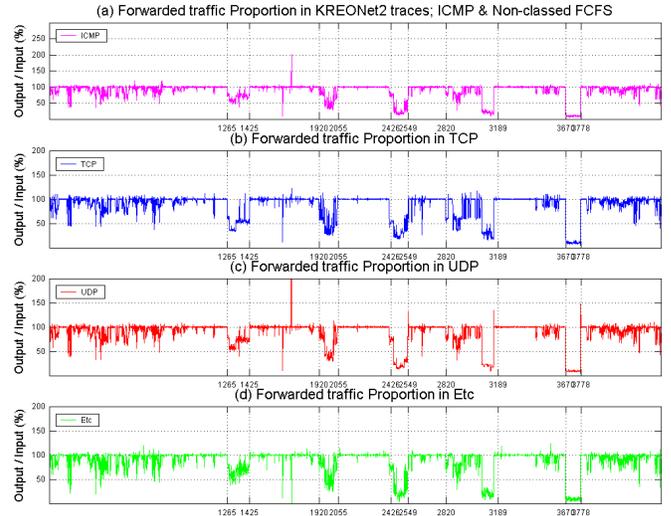Figure. 4. Output traffic proportion by protocol without regulation.

Figure. 5. Forwarded traffic proportion by protocol without regulation.

corresponding protocol because non class-based buffer management doesn't provide any regulation or differentiation that can protect the system from abnormal traffic. Actually the output of the traffic is nearly similar to the input of the traffic in non class-based as shown in Fig. 3 and Fig. 4.

Fig. 5 shows the forwarded traffic proportions of the FCFS scheme. The forwarded proportion is defined as the ratio of output to input in percentage. Ideally, the ratios stay around 100% at normal times. Because the FCFS scheme does not regulate the output rates for each class, it results in uniform packet drops for every protocol during all the attacks. As a result, it leads to low forwarding rates for all classes.

4.3 The Flexible Buffer Management

Fig. 6 shows the result of our flexible buffer management scheme. It operates with FCFS buffer management during the absence of network attacks. On the other hand, it changes the mode to class-based buffer management during network attack periods. Ideally, we expect the performance close to the class-based buffer management performance during the attack periods. In Fig. 6, the proposed approach shows performance close to class-based buffer management i.e., within established weights.

The output rate of each protocol remains around the weight which we set in spite of many network attacks. There is some latency between the onset of an attack and the generation of the attack detection signal. This latency results in instantaneous peaks in output traffic at the beginning of the attack periods. However, the instantaneous increase of output traffic is immediately suppressed by the output control in real-time. Especially, TCP experiences instantaneous decrease of the output rate as a result of attacks staged by other protocols like UDP and ICMP. However, when the system starts to apply the class-based management, TCP can return to its normal proportion of traffic.

Fig. 7 shows the forwarded traffic proportions of the proposed approach. Our scheme detects network attacks in real-time and sends the control signal to switch to class-based buffer management. Consequently, it regulates the output rates

for each class to its assigned weight during attack periods and results in containment of attack packets through a higher rate of packet drops of that protocol. Simultaneously it protects the output of other innocent protocols. As a result, it leads to low forwarding rates for the protocol employed by the attack traffic.

Our experiments on NLANR traces with nine simulated attacks of various configurations show that the proposed approach performs equally well on different types of attacks as shown in Fig. 8.

V. CONCLUSION AND FUTURE WORK

DoS attacks tend to exploit a specific protocol. Therefore, when network attacks are staged, the input traffic of the corresponding protocol increases substantially. We used this property to design a network attack detector. During the normal time, we use non class-based buffer management to diminish the system load and during the attack time, we use class-based buffer management techniques to contain attack traffic. Our trace-driven evaluation shows that the proposed approach is effective in both detecting and mitigating the detected attacks.

Our design does not consider which application (port number) is responsible for the detected network attack. When attack is detected, our system just processes packets at a predefined weight for all the applications employing that protocol. On the other hand, if we are able to exactly distinguish which port number is employed by the by the attack, we could regulate only the attack traffic.

REFERENCES

[1] A. Garg and A. L. Narasimha Reddy, "Mitigation of DOS attacks through QOS regulation", in *Proceedings of IWQOS*, May 2002.
[2] D. Moore, G. Voelker and S. Savage, "Inferring Internet Denial of Service Activity", in *Proceedings of USENIX Security Symposium'2001*, August 2001.
[3] CERT@ Coordination Center, "*CERT@ Advisory CA-1997-28 IP Denial-of-Service Attacks*", December 1997.
Available : http://www.cert.org/advisories/CA-1997-28.html
[4] Danilo Bruschi, Emilia Rosti, "Disarming offense to facilitate defense",
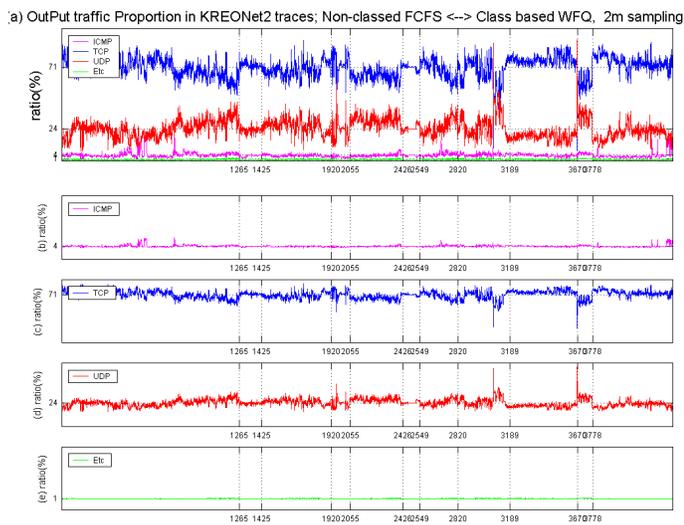
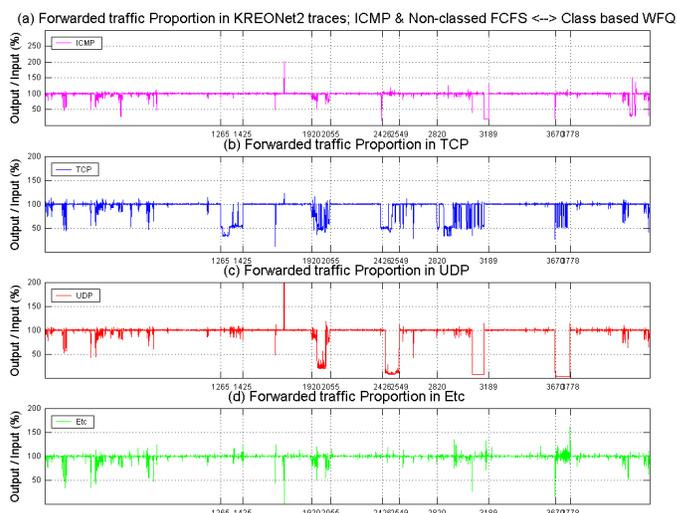Figure. 6. Output traffic proportion by protocol with the proposed approach.

Figure. 7. Forwarded traffic proportion by protocol in the proposed approach.

in *Proceedings of the 2000 workshop on New security paradigms p. 69 - 75*, Ballycotton, County Cork, Ireland, 2000.
[5] National Laboratory for Applied Network Research (NLANR), measurement and operations analysis team, "*NLANR network traffic packet header traces*".
Available : http://pma.nlanr.net/Traces/
[6] Haining Wang, Danlu Zhang, Kang G. Shin, "Detecting SYN Flooding Attacks", in *Proceeding of IEEE INFOCOM'2002*, New York City, NY, 2002.
[7] D. Moore, V. Paxon, S. Savage, C. Shannon, S. Staniford and N. Weaver, "The Spread of the Sapphire/Slammer Worm", in *The technical report of CAIDA*.
http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html
[8] Cynthia Wong, Chenxi Wang, Dawn Song, Stan Bielski and Gregory R. Ganger, "Dynamic Quarantine of Internet Worms", in *The International Conference on Dependable Systems and Networks (DSN-2004), pp. 62-71*, Florence, Italy, June 28 – July 01, 2004.
[9] KREONet2 (Korea Research Environment Open NETwork2). Available : http://www.kreonet2.net
[10] P. Barford, J. Kline, D. Plonka and A. Ron, "A Signal Analysis of Network Traffic Anomalies," in *Proc. of ACM SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, November 2002.
[11] Seong Soo Kim, A. L. Narasimha Reddy and Marina Vannucci, "Detecting traffic anomalies through aggregate analysis of packet header data", in *Proceedings of Networking 2004, LNCS 3042, pp. 1047-1059*, Athens, Greece, May 2004.
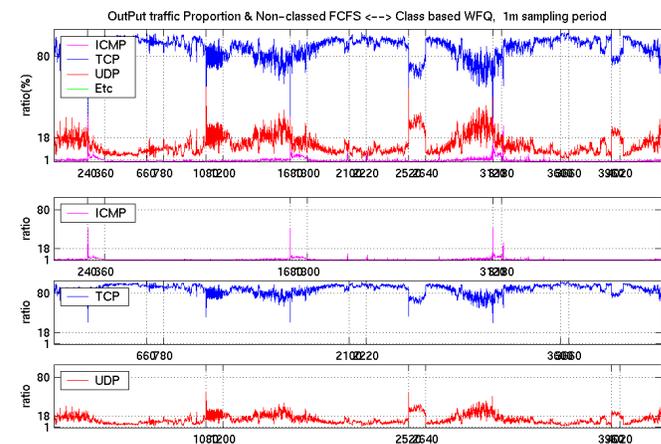
Figure. 8. Output traffic proportion by protocol with the proposed approach in NLANR traces. The three major protocols, ICMP, TCP and UDP, are exploited in turn.