Massively Parallel Anomaly Detection in Online Network Measurement

Shashank Shanbhag and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts Amherst, MA, USA
Email: {sshanbha,wolf}@ecs.umass.edu

***Abstract*—Detecting anomalies during the operation of a network is an important aspect of network management and security. Recent development of high-performance embedded processing systems allow traffic monitoring and anomaly detection in real-time. In this paper, we show how such processing capabilities can be used to run several different anomaly detection algorithms in parallel on thousands of different traffic subclasses. The main challenge in this context is to manage and aggregate the vast amount of data generated by these processes. We propose (1) a novel aggregation process that uses continuous anomaly information (rather than binary outputs) from existing algorithms and (2) an anomaly tree representation to illustrate the state of all traffic subclasses. Aggregated anomaly detection results show a lower false positive and false negative rate than any single anomaly detection algorithm.**

***Index Terms*—Network measurement, anomaly detection, data aggregation, network processor.**

I. Introduction

One important goal of passive network measurement is to monitor the operational characteristics of a network. Detecting anomalies in network traffic is one of the key functionalities of a measurement system in this context. Anomalies can arise from a number of different causes, ranging from benign changes in traffic patterns (e.g., routing changes, novel networking applications and protocols) to malicious denial of service attacks or intrusion attempts (e.g., TCP SYN attacks, port scans). To detect such changes, numerous anomaly detection algorithms have been developed. However, practically all existing algorithms are limited in what anomalies they can detect.

To expand the capabilities of anomaly detection, we present a system that can implement multiple anomaly detection algorithms that monitor a large number of traffic classes in parallel. Such a massively parallel anomaly detection (MPAD) system has become technologically feasible in recent years as high-performance embedded multi-core network processors have been developed and used for online passive measurement [1]. A key capability of this system is to monitor traffic for anomalies in real-time (i.e., online, as traffic traverses the node, rather than processing trace files offline).

Our contributions towards making such MPAD systems a reality are: (1) a normalization and aggregation process that effectively combines and interprets the results from multiple different anomaly detection algorithms; (2) a visualization of anomaly levels in different traffic classes through the use of anomaly trees; and (3) results from a prototype system based on an Intel IXP2400 network processor that illustrates the detection accuracy of MPAD systems.

The remainder of this paper is organized as follows. Section II discusses related work including some anomaly detection algorithms that we use in our system. Section III describes the overall MPAD architecture. Section IV presents our normalization and aggregation process. Results from our prototype system are shown in Section V. Section VI summarizes and concludes this paper.

II. Related Work

There exists an extensive body of work on anomaly detection algorithms. These algorithms use different types of information from the packet stream: Holt-Winter uses a volume-based forecasting model [2]; Barford et al. use frequency information to determine anomalies from a signal processing point of view [3]; Gu et al. have proposed a maximum-entropy-based anomaly detection algorithm [4]; Lakhina et al. [5] use a subspace method to identify anomalies in byte counts, packet counts, and IP-flow counts; Siris and Papagalou [6] present a statistical anomaly detection algorithm that identifies SYN flooding attacks using adaptive threshold and cumulative sum. We use several of these algorithms in our MPAD prototype. While most algorithms consider anomalies in terms of traffic volume, there are also anomalies caused by transmission timing and packet header values [7], which we do not consider in this work.

III. Parallel Online Anomaly Detection

Before describing the system design of our massively parallel anomaly detection node, we briefly discuss our motivation for online operation and parallelism.

A. Online Operation and Parallelism

An online anomaly detection system must be able to process all observed traffic in real-time in order to inform the user of anomalies during online operation. This leads to strict performance requirements on how quickly processing needs to be complete before the next packet arrives, but it also simplifies the system design. It is not necessary to collect and transfer large trace files (which can easily amount to tens of Gigabytes per hour for headers alone) from the measurement node to a storage facility for off-line processing. Instead, only

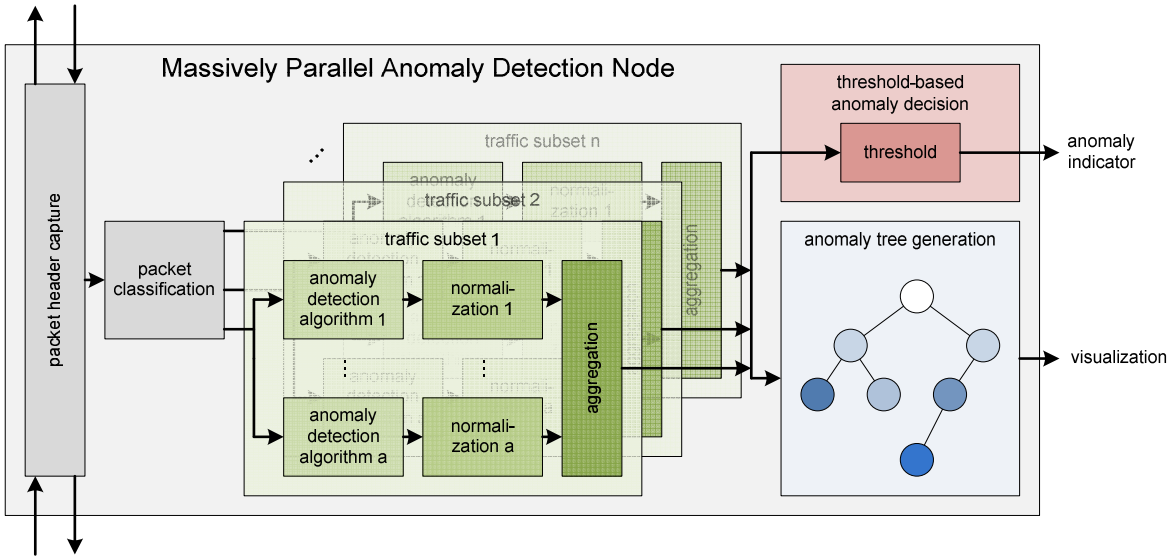

Fig. 1. Data Flow Through MPAD Node

the summary information of the anomaly detection process is reported to the user.

One drawback of such an approach is that most information about the network traffic is not retained in the summary information provided by the anomaly detection system. Therefore it is crucial that the anomaly detection system considers as many characteristics of the network traffic as possible. For this purpose, we exploit two dimensions of parallelism in anomaly detection:

- Parallelism across multiple anomaly detection algorithms: The MPAD node implements several existing anomaly detection algorithms that analyze traffic in parallel to leverage their capabilities and to make aggregate decisions.
- Parallelism across multiple traffic subsets: The MPAD node applies all anomaly detection algorithms to numerous different subsets of traffic to detect targeted anomalies that would be difficult to notice in the overall traffic.

We focus on finding anomalies that can be characterized by changes in traffic volume and occur in conventional subsets of traffic (e.g., TCP SYN packets, traffic with certain port ranges, etc.). It would be conceivable to extend MPAD and attempt finding new anomalies by randomized search over new traffic subsets, but this is beyond the scope of this paper.

B. MPAD Node Design

The architecture of our MPAD node design is shown in Figure 1. After packet headers are captured from the monitored link, a packet classification module divides traffic into different subsets. Any given packet may belong to multiple subsets. For each subset to which the packet belongs, all anomaly detection algorithms process the packet. The outputs from all algorithms are normalized and aggregated. The aggregated information is used to visualize the state of the network traffic and to notify users of anomalies.

IV. Normalization and Aggregation

With a general understanding of the MPAD architecture, we now turn towards the key technical challenge: how to aggregate information from multiple anomaly detection algorithms and how to visualize this information.

A. Continuous Anomaly Metric

When using multiple different anomaly detection algorithms, it is important to consider how outputs from different algorithms can be aggregated into a single metric. Existing algorithms typically yield a binary decision (i.e., a '1' if an anomaly is detected and '0' if not). One could aggregate such binary outputs by using majority decision or similar binary functions. However, we believe that such an approach is too coarse.

Instead, we observe that practically all anomaly detection algorithms use continuous metrics internally, before applying a threshold and generating the binary output. Therefore, our system uses these continuous metrics for aggregation and applies the threshold as late as possible (e.g., just before the output as shown in Figure 1). Since different algorithms use different internal metrics, it is crucial to normalize these metrics before aggregation. This normalization process is explained after we introduce the necessary notation.

B. Notation

The anomaly detection algorithms we consider in this paper use traffic volume information to determine anomalies. Thus, we denote the packet count during the interval $[t, t+\tau)$ with c^t ("packet count"). We assume τ to be a fixed time interval (e.g., τ=1 second). The packet classifier classifies these packets into one or more of s different traffic subsets S_i, $1 \leq i \leq s$.

The packet count for each subset is c_i^t. The MPAD system uses a parallel anomaly detection algorithms A_j, $1 \leq j \leq a$. These algorithms use some metric, $m_{i,j}^t$, that determines the

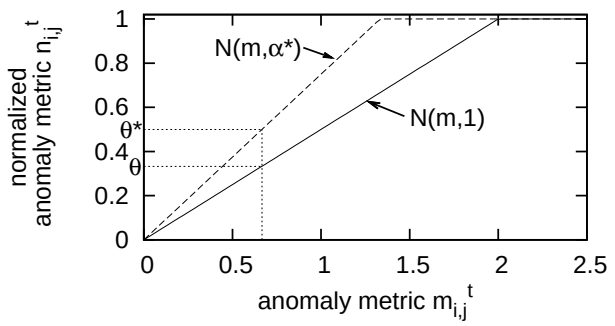

Fig. 2. Normalization Function $N(m, \alpha)$. Parameter α^* ensures that the optimal separation between anomalies and non-anomalies is at θ^*=0.5.

level of anomaly determined by algorithm j for subset i at time interval $[t, t+\tau)$. For most algorithms, this metric is based on a prediction of traffic volume (in number of packets), $p_{i,j}^t$, that is considered "normal." This prediction is based on the history of previous packet counts $c_i^{t_k}$, $t_k < t$. Once the actual packet count, c_i^t, for this interval is available, it is compared to the prediction, $p_{i,j}^t$. The continuous anomaly metric is then $m_{i,j}^t$=$c_i^t/p_{i,j}^t$, which relates the actual packet count to the predicted volume.

C. Normalization and Aggregation of Algorithm Outputs

Before aggregating the anomaly metrics from all algorithms, we need to normalize them to ensure equal influence on the aggregate value. We normalize to an interval $[0, 1]$ with 0 representing no anomaly to 1 representing an anomaly. We define a normalization function, N, that provides the normalized anomaly metric, $n_{i,j}^t$, for algorithm j and subset i at time t:

$$n_{i,j}^t = N\left(m_{i,j}^t, \alpha_j\right) = \min(1, max(0, \frac{1}{2} \cdot \alpha_j \cdot m)). \quad (1)$$

Parameter α_j determines the shape of the normalization function. We use this parameter to adjust the normalized intensity metric such that optimal separation between anomalies and no anomalies occurs exactly in the middle of the range $[0 \ldots 1]$ at θ^*=0.5 for $\alpha_j^* = \frac{n^*}{\theta_j}$ (as illustrated in Figure 2).

We have explored other normalization function (e.g., $N(m,\alpha)$=$\max(1-e^{-\alpha \cdot \ln m}, 0)$, $N(m,\alpha)$=$1-e^{-|\alpha \cdot \ln m|}$), but the function shown in Equation 1 provides the best overall performance.

With the availability of this normalized anomaly indicator, we can aggregate $n_{i,j}^t$ across all algorithms A_j. We use the aggregation function G to determine the aggregated anomaly intensity g_i^t across all a algorithms for subset i at time t. We have evaluated arithmetic mean, geometric mean, median, minimum, maximum, etc., but found the following linear combination of maximum and arithmetic average to be most effective (as suggested in [8]):

$$g_i^t = G\left(n_{i,1}^t, \ldots, n_{i,a}^t\right) = \frac{1}{2}\left(\frac{1}{a}\sum_{j=1}^{a} n_{i,j}^t + \max_j n_{i,j}^t\right). \quad (2)$$

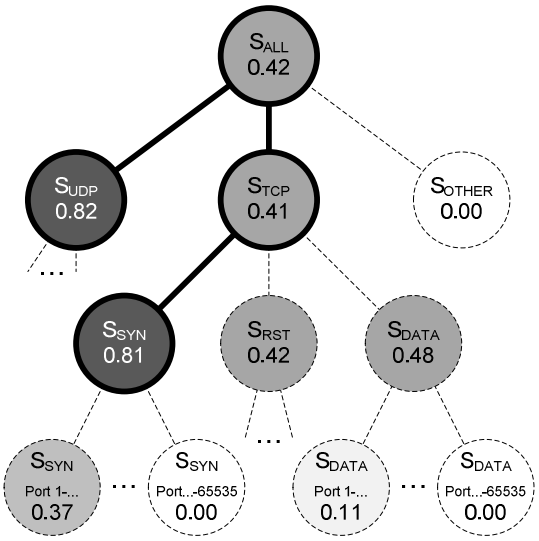

Fig. 3. Anomaly Tree Example. Dashed lines indicate nodes not shown to the user.

Finally, the aggregated anomaly metric is compared to a threshold, θ^*, to make a binary decision regarding the occurrence of an anomaly. This information is provided to the user via report r_i^t with

$$r_i^t = \begin{cases} 0, & \text{if } g_i^t < \theta^* \\ 1, & \text{if } g_i^t \geq \theta^*. \end{cases} \quad (3)$$

D. Visualization

The normalization and aggregation process reduces the amount of information for each traffic subset S_i from a continuous values ($m_{i,j}^t$, $1 \leq j \leq a$) to a single binary anomaly report r_i^t. For a system that monitors a large number of traffic subsets (in the order of thousands), this amount of information may still be too much for a user to understand. Thus, we use a graphical representation that we call "anomaly tree." The tree structure relates all traffic subsets S_i to tree nodes V_i, where V_j is placed into the subtree below V_i if $S_j \subseteq S_i$. Each tree node contains the normalized and aggregated anomaly metric g_i^t. For the visualization of the anomaly tree we use the following rules:

- Node visibility: By default, nodes are hidden. A Node V_i is shown if (1) the anomaly metric is high enough to have caused an anomaly report (i.e., r_i^t=1) or (2) the subtree below node V_i contains a node for which (1) applies. Only nodes that are anomalous and their path to the root node are shown.
- Node coloring: The color intensity of the node is proportional to the aggregated anomaly metric g_i^t. Coloring allows for quick identification of anomalous nodes, the level of anomaly, and relationship between anomalous node.

The tree is dynamically updated every interval τ. An example of an anomaly tree is shown in Figure 3.

TABLE I
PACKET TRACES USED IN EXPERIMENTS

Trace	Source	Trace Duration	Anomaly Type	Description
T_1	Los Nettos Trace 4 [9]	439 sec	TCP SYN flood	Persistent low packet rate traffic from 5 sources targeting single victim IP on 4 ports.
T_2	Los Nettos Trace 18 [9]	1057 sec	TCP SYN	Persistent traffic of different intensities from large number of sources targeting a single port on 7 victim IPs.
			TCP No-Flag	High intensity attack targeting single IP on 65530 ports from 788820 sources.
			UDP flood	Persistent flooding of 3 ports on a single IP from 5 source IPs and 250 source ports.
T_3	Los Nettos Trace 29 [9]	956 sec	TCP SYN	Persistent traffic of different intensities from a large number of source IPs targeting different ports on 5 victim IPs.
			UDP flood	Persistent flooding of 4 victim IPs from large number of sources.
T_4	Code Red II [10]	248 sec	TCP SYN portscan	Single IP scans HTTP port on a large number of victim IPs.
T_5	MIT Lincoln Labs DDoS [11]	6167 sec	TCP RST flood	Single IP flooded by a large number of spoofed source IPs.

V. EVALUATION

We have implemented a prototype MPAD node to evaluate the effectiveness of parallel anomaly detection.

A. Experimental Setup

1) Prototype System: Our prototype is based on an online measurement node that we have developed on an Intel IXP2400 network processor platform [1]. The system uses multiple data path processors to classify packets and update packet counts for each monitored subset of traffic. These packet counts are stored in shared SRAM. At fixed intervals of τ=1 second, the XScale control processor reads the appropriate SRAM locations and computes $m_{i,j}^t$, $n_{i,j}^t$, g_i^t, and r_i^t. The resulting summary information of g^t and r^t can then be transmitted to the user via a socket interface. While it is conceptually possible to implement all functionality of computing summary information and generating the anomaly tree on the embedded control processor on the IXP2400, our current prototype still performs some computations offline (e.g., tree generation).

The a=5 different anomaly detection algorithms that we implement in our prototype are (with examples where these algorithms have been used in practice):

- HW: Holt Winter Forecasting Model (e.g., [2]).
- ADAP: Adaptive Threshold Algorithm (e.g., [6]).
- AVG: Average over Window (e.g., [12]).
- EWMA: Exponential Weighted Moving Average (e.g., [13]).
- CUSUM: Cumulative Sum Algorithm (e.g., [14]).

At runtime the system monitors a total of s=2031 subsets of traffic including subsets that distinguish UDP and TCP, TCP flags (e.g., SYN, RST), and port numbers (as done in [4]).

2) Traces: We evaluate the performance of our prototype system by comparing its ability to identify anomalies in packet traces with known anomalies. We use the traces shown in Table I, where each trace is played back using `tcpreplay`. Thus, the MPAD system receives traffic that is identical to a realistic online scenario while ensuring that results are reproducible.

3) Labeling Methodology: In order to quantify the detection accuracy of MPAD, we need to determine manually when anomalies occur in the traces we use. We have labeled each time interval of each trace by comparing the packets in the trace with the verbal description of the anomaly (shown in Table I). If during any interval of τ=1 seconds any packets match the anomaly description, that interval is labeled as anomalous. This labeling is then compared to the output r_i^t from the MPAD system.

B. Results

1) Metrics over Time: We start the presentation of results with an illustration of the operation of the MPAD system over time. Figure 4 shows the main steps of the MPAD system for trace T_1. Since trace T_1 consists of TCP SYN attacks, we only discuss the TCP SYN traffic subset, denoted as S_{SYN}. The top graph in Figure 4 shows the TCP SYN packet count, c_{SYN}^t, and the manual classification into normal and anomalous time intervals. The middle graph shows the normalized anomaly metrics, $n_{SYN,j}^t$, provided by the five anomaly detection algorithms we have implemented on the prototype system. The bottom graph shows the aggregate anomaly metric, g_{SYN}^t. Also shown are the binary classification results, r_{SYN}^t, indicating by color if an anomaly is reported. In this example, the MPAD system yields a false positive rate of 4.4% and a false negative rate of 3.3%.

2) Normalization Parameters: For each algorithm A_j, we need to determine the best normalization function shape parameter α_j^*. The results for each trace are shown in Table II. We can observe that most parameter values for a given algorithm are similar across different traces. The aggregate parameters used in the MPAD implementation are obtained by averaging across traces.

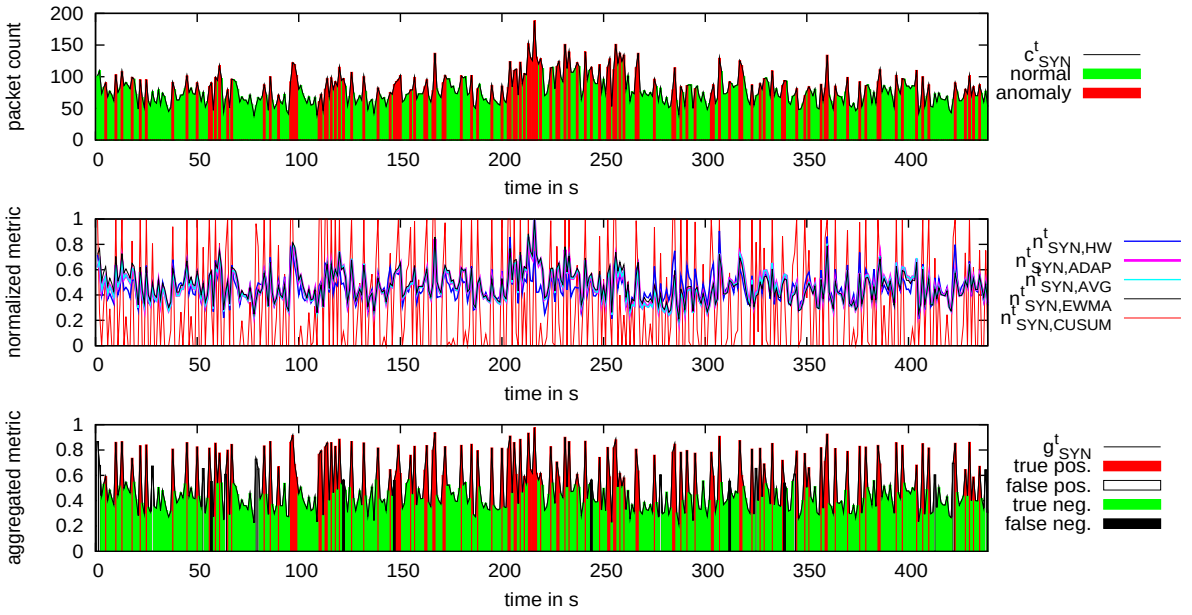

Fig. 4. Behavior of Metrics Over Time.

TABLE II
NORMALIZATION PARAMETER α_j FOR ALL ALGORITHMS AND TRACES. ALL SUBSETS THAT TRIGGER ANOMALIES ARE SHOWN. THE AGGREGATE NORMALIZATION PARAMETER α_j^* (USED IN THE AGGREGATION FUNCTION G) SHOWS THE AVERAGE OF α_j ACROSS ALL TRACES.

Algorithm		Normalization Parameter α_j									Aggregate Normalization
		T_1	T_2				T_3		T_4	T_5	
j	name	S_{SYN}	S_{SYN}	S_{RST}	S_{NOFLAG}	S_{UDP}	S_{SYN}	S_{UDP}	S_{SYN}	S_{RST}	Parameter α_j^*
1	HW	0.95	0.95	0.99	0.94	0.97	0.88	0.97	0.74	0.93	0.927
2	ADAP	1.12	1.14	1.16	1.11	1.12	1.11	1.15	0.63	1.12	1.075
3	AVG	0.95	0.96	0.98	0.94	0.96	0.94	0.97	0.57	0.95	0.915
4	EWMA	0.98	0.99	0.99	0.97	1.01	0.95	0.98	0.61	0.98	0.940
5	CUSUM	1.39	1.22	1.03	1.23	1.18	1.49	1.09	2.86	2.27	1.529

3) Detection Performance of MPAD: The key question about MPAD is if the aggregated anomaly metric indeed provides better detection performance than individual algorithms. To answer this question, we compare receiver operating characteristic (ROC) curves. An ROC curve is an illustration of the sensitivity of a binary classifier (i.e., anomaly detection algorithm) for different threshold values (i.e., θ) [15]. The x-axis shows the false positive rate and the y-axis shows the true positive rate. The point along the ROC curve that is most interesting to us is where distance between the curve and the diagonal (i.e., random guess) is maximized.

Figure 5 shows the ROC curves for all individual algorithms (using the best possible parameters shown in Table II) and the ROC curve for the MPAD system using aggregated parameters. We observe that MPAD outperforms any individual algorithm, even though a single set of parameters is used for all traces.

To quantify this improvement in classification quality in the MPAD system, we show the false positive and false negative rate in Table III. Entries shown in bold are the lowest false positive or false negative rates achieved by any algorithm. Except for the false negative rate for T_4's subset S_{SYN}, the MPAD system provides lower Type I and Type II errors than any given algorithm. The differences in these error rates are shown in percentage in the last row. The median improvement is a 10.0% lower false positive rate and a 37.9% lower false negative rate. These results show that MPAD provides significantly better detection accuracy than any single anomaly detection algorithm.

VI. SUMMARY AND CONCLUSION

We have discussed the design of a massively parallel anomaly detection system that analyzes traffic in real-time and can detect anomalies by using multiple existing anomaly detection algorithms in parallel. The evaluation of our prototype implementation on an Intel IXP2400 network processor shows the effectiveness of this approach as the MPAD system provides false positive rates and false negative rates that are significantly lower than those of any single algorithm.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0325868.

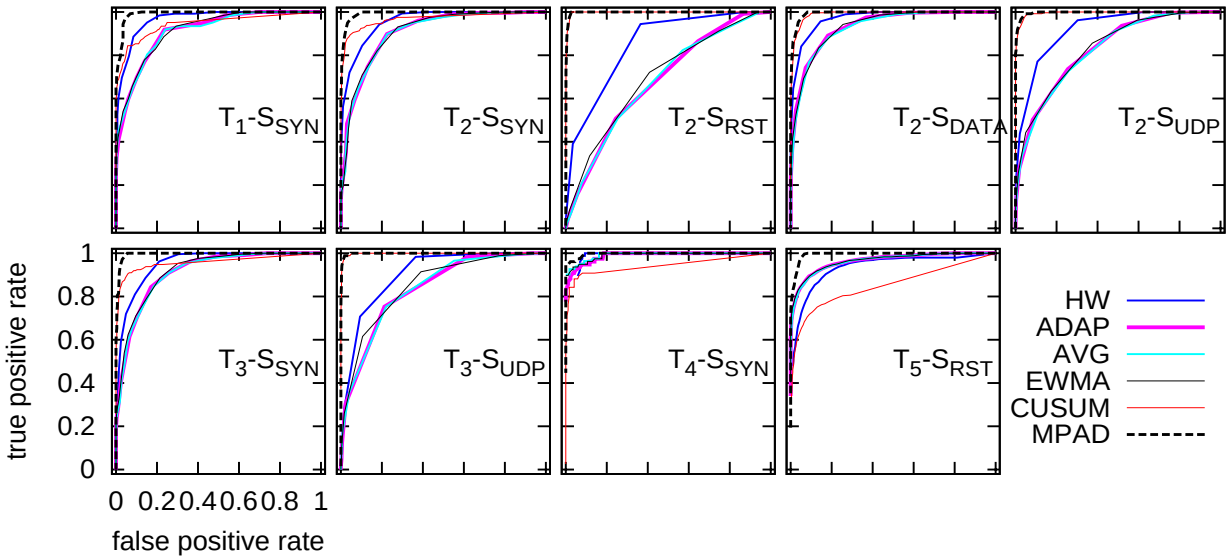

Fig. 5. ROC Curves Comparing Detection Performance for All Traces and All Anomalous Traffic Subsets.

TABLE III
FALSE POSITIVE RATE / FALSE NEGATIVE RATE OF INDIVIDUAL ALGORITHM AND MPAD SYSTEM.

Algorithm		False Positive Rate / False Negative Rate								
		T_1	T_2				T_3		T_4	T_5
j	name	S_{SYN}	S_{SYN}	S_{RST}	S_{NOFLAG}	S_{UDP}	S_{SYN}	S_{UDP}	S_{SYN}	S_{RST}
1	HW	.085/.115	.188/**.058**	.365/.056	.146/.043	.107/.230	.210/**.038**	.364/.017	.093/**.000**	.117/.147
2	ADAP	.237/**.082**	.225/.102	.243/.493	.182/.109	.255/.264	.170/.157	.212/.247	.047/.079	.090/**.107**
3	AVG	.240/**.082**	.226/.102	.253/.493	.187/.116	.259/.272	.185/.152	.221/.247	**.018**/.074	.092/.109
4	EWMA	.233/.115	.277/.069	.409/.279	.136/.167	.375/.145	.214/.114	.390/.086	.035/.064	**.076**/.121
5	CUSUM	**.057**/.156	**.046**/.105	**.029**/**.000**	**.060**/**.036**	**.050**/**.017**	**.058**/.095	**.020**/**.011**	.041/.118	.092/.282
MPAD		.044/.033	.043/.036	.027/.000	.052/.025	.047/.009	.055/.000	.018/.011	.000/.053	.066/.004
Improve-ment		22.8%/ 59.8%	6.5%/ 37.9%	6.9%/ 0%	13.3%/ 30.6%	6.0%/ 47.1%	5.2%/ 100.0%	10.0%/ 0%	100.0%/ -100.0%	13.2%/ 96.3%

REFERENCES

[1] T. Wolf, R. Ramaswamy, S. Bunga, and N. Yang, "An architecture for distributed real-time passive network measurement," in *Proc. of 14th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Monterey, CA, Sep. 2006, pp. 335–344.

[2] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proc. of the 14th Systems Administration Conference*, New Orleans, LA, Dec. 2000, pp. 139–146.

[3] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet Measurement (IMW)*, Marseille, France, Nov. 2002, pp. 71–82.

[4] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, Berkeley, CA, Oct. 2005.

[5] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina, Italy, Oct. 2004, pp. 201–206.

[6] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, Dallas, TX, Nov. 2004, pp. 2050–2054.

[7] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, Aug. 2003, pp. 75–86.

[8] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, "Data fusion for outlier detection through pseudo-ROC curves and rank distributions," in *Proc. of International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Jul. 2006, pp. 2166–2173.

[9] A. Hussain, J. Heidemann, and C. Papadopoulos, "A framework for classifying denial of service attacks," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Karlsruhe, Germany, Aug. 2003, pp. 99–110.

[10] D. Moore, C. Shannon, and J. Brown, "Code-Red: a case study on the spread and victims of an internet worm," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, Marseille, France, Nov. 2002, pp. 273–284.

[11] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX '00)*, vol. 2, Hilton Head, SC, Jan. 2000, pp. 12–26.

[12] C. Schwarzer, "Prediction and adaptation in a traffic-aware packet filtering method," Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, Mar. 2006.

[13] S. Deshpande, M. Thottan, T. K. Ho, and B. Sikdar, "A statistical approach to anomaly detection in interdomain routing," in *Proc. of Third International Conferenc on Broadband Communications, Networks, and Systems (BROADNETS)*, San Jose, CA, Oct. 2006.

[14] H. Wang, D. Xhang, and K. G. Shin, "Change-point monitoring for the detection of dos attacks," *IEEE Transactions on Dependable Secure Computing*, vol. 1, no. 4, pp. 193–208, Oct. 2004.

[15] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006.