Accurate Anomaly Detection through Parallelism

Shashank Shanbhag and Tilman Wolf, University of Massachusetts

Abstract

In this article we discuss the design and implementation of a real-time parallel anomaly detection system. The key idea is to use multiple existing anomaly detection algorithms in parallel on thousands of network traffic subclasses, which not only enables us to detect hidden anomalies but also to increase the accuracy of the system. The main challenge then is the management and aggregation of the vast amount of data generated. We propose a novel aggregation process that uses the internal continuous anomaly metrics used by the algorithms to output a single system-wide anomaly metric. The evaluation on real-world attack traces shows a lower false positive rate and false negative rate than any individual anomaly detection algorithm.

Monitoring the correct operation of a network is an essential network management task. Anomaly detection algorithms attempt to profile the "normal" behavior of the network. Understanding the normal behavior of a network makes it possible to identify situations where network characteristics deviate. In case of such *anomalies*, it may be possible to infer the cause of the abnormal behavior and take corrective action. The causes for network anomalies are numerous and varied. Examples include link failures that cause changes in routing and traffic loads on other links, flash crowds that cause increased traffic loads due to sudden popularity of some content that is accessed via the network, and distributed denial of service (DDoS) attacks that intentionally increase the traffic load on some links.

Note that the definition of what constitutes an anomaly is often not easy to translate into a clear quantitative definition. For example, a DDoS attack increases the traffic volume, but determining the threshold where it exceeds normal fluctuations in network traffic volume is difficult. Therefore, there is no perfect anomaly detection algorithm that can identify all anomalies correctly. A direct implication is that *accuracy* is a crucial metric for any anomaly detection system, as false positives (i.e., alarms on normal traffic) and false negatives (i.e., no alarms on anomalous traffic) are detrimental for operational reasons.

Research in anomaly detection therefore aims to find a detection process that provides high accuracy. In prior and related work, many different algorithms have been proposed using a variety of techniques. Typically, these algorithms exhibit different levels of sensitivity to different types of anomalous traffic. A key challenge for a designer of an anomaly detection system is therefore to select the most suitable algorithm. A related challenge is the question of what subset of monitored traffic to consider for the anomaly detection process. For example, attacks that swamp a victim with TCP SYN requests are typically of low volume in comparison to all the traffic on a link. Detecting changes in TCP SYN traffic volume can therefore not be done effectively unless only the subset of TCP SYN traffic is considered. We address these two problems in our work by proposing a parallel anomaly detection (PAD) system.

The key idea of PAD is to exploit recent technological advances, implementing multiple algorithms on a single system and monitoring multiple traffic subsets in parallel. The key benefits of such a design (compared to traditional anomaly detection systems) are:

- Higher detection accuracy: Combining multiple existing anomaly detection algorithms with the normalization and aggregation process described in this article, PAD achieves higher accuracy than any single algorithm.
- Higher sensitivity to a range of anomalies: Monitoring numerous traffic subsets in parallel allows PAD to be sensitive to anomalies that are specific to a particular traffic class.

The PAD design we propose has become feasible only in recent years as high-performance embedded processing systems have become available for network systems (e.g., network processors on routers). These processing systems allow real-time processing of monitoring information (rather than collecting traffic traces and post-processing them). Thus, anomaly detection results become available to a network operator practically instantaneously.

However, there are several fundamental challenges that need to be addressed to make PAD a practical system. In particular, it is important to determine how to manage and interpret the data from multiple anomaly detection algorithms. The contributions of this article are as follows:

- A system design for parallel anomaly detection
- An aggregation process that increases the accuracy of the overall system beyond that of any single algorithm
- A detailed evaluation of a prototype system that illustrates the feasibility and effectiveness of PAD, and shows that it achieves higher detection accuracy than any single anomaly detection algorithm

The remainder of this article is organized as follows. The

0890-8044/09/$25.00 © 2009 IEEE

next section discusses related work. We then introduce the general design of the PAD system. We discuss how we aggregate information from multiple anomaly algorithms that are run in parallel. The following section presents the evaluation result from our prototype PAD system. The final section summarizes and concludes this article.

Related Work

Anomaly detection is an active area of research and a number of different anomaly detection algorithms have been proposed. These anomaly detection algorithms differ either in the technique used to detect deviations from normal behavior or in the type of data from the packet stream that is used for analysis. In practically all cases, traffic is monitored at regular intervals to obtain time-series data. These data are then interpreted using different approaches. Examples are:

- Rate-based detection: In rate-based detection, the history of previously observed data rates is used to compute an estimate for the current interval. If the observed data rate differs from the estimate (plus/minus a certain range of permissible variation), an anomaly is reported. Examples of such rate-based detection algorithms are Holt-Winter forecasting [1], adaptive threshold [2], average over window [3], exponential weighted moving average [4], and cumulative sum [5]. Since this kind of anomaly detection is most widely used, we focus our work on this class of algorithms.
- Frequency information-based detection: In frequency-based detection, packet rates are transposed into the frequency domain and analyzed for anomalies using signal processing methods (e.g., wavelets [6]).
- Entropy-based detection: In entropy-based detection, traffic is split into different classes. Anomalies in traffic rates are determined by changes in the entropy compared to a baseline [7].

Other anomaly detection algorithms are based on machine learning, clustering, and related techniques.

The work presented in this article is based on some of our own prior work. The use of embedded processing systems for online network measurement has been discussed in [8]. A detailed analysis of aggregating anomaly detection algorithms has been presented in [9]. The parallel anomaly detection system has been evaluated with realistic attack traffic on a testbed in [10]. In addition to the work carried out in [9] and [10], we also discuss the various techniques used to aggregate anomaly detection algorithms in this work.

Exploiting Parallelism in Anomaly Detection Systems

The processing performance of an anomaly detection algorithm in an operational network is an important consideration. While anomaly detection accuracy is the most important design goal, it is also important to consider real-time processing constraints. To permit continuous operation of an anomaly detection system, the processing performance must be such that traffic can be handled at the peak rate of the monitored link. If this rate cannot be sustained, monitoring information will eventually backlog and be discarded. Thus, available technology dictates what kind of anomaly detection systems are practically feasible.

Technology Trends

Traditional anomaly detection systems often operate in what can be described as a batch or offline mode. In such systems, packet traces from the monitored link are collected and transferred to a workstation for anomaly detection processing. The main drawbacks of this design are twofold:

- Large trace files need to be transferred and stored (at least temporarily).
- Anomaly detection results are only available after an inherent delay (depending on the batch size).

Also, the complexity of the anomaly detection algorithm is limited to the amount of processing the workstation computer can accomplish while the next packet trace is collected.

Advances in embedded processor design have made it possible to integrate significant amounts of processing power in network systems. These embedded processors are often multicore systems with a handful to dozens of processor cores. Examples range from routers with conventional server processors to high-performance network processors [11]. Using such processing resources, the following two changes can be made to the design of anomaly detection systems:

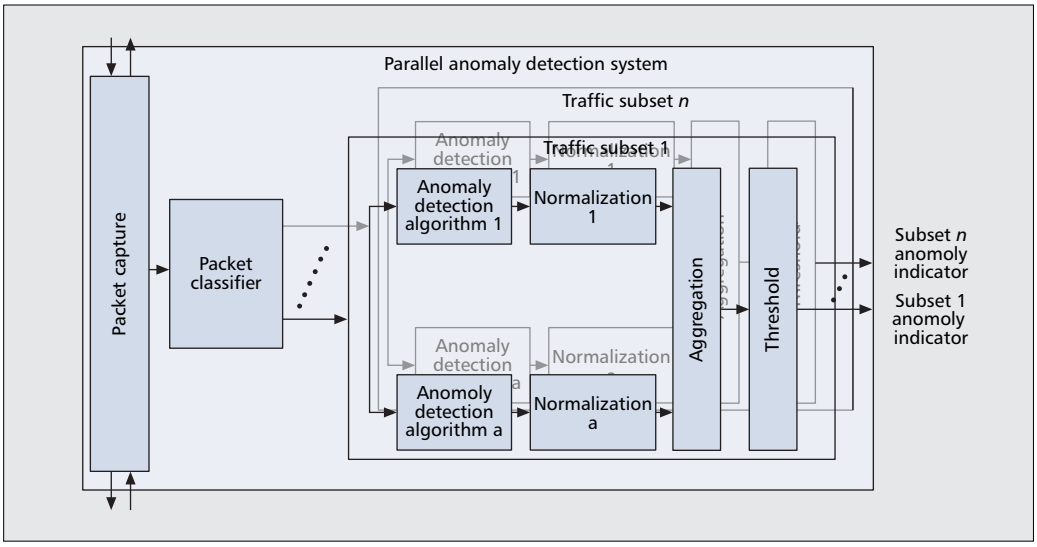
- Online processing: Processing resources that are located on the line cards of routers can be used to perform network measurement and anomaly detection processing. Therefore, the collection and transfer of packet traces is no longer necessary. Instead, every packet is processed as it traverses the network system. Such online processing is discussed in more detail in [8].
- Processing power: Embedded multicore processors can dedicate more processing power to anomaly detection than conventional workstation computers. One of the reasons is the large number of processor cores (e.g., 16 cores on the Intel IXP2400). Another reason is the improved handling of traffic I/O. In many conventional workstations, the operating system consumes much of the processing resources when reading packet traces at high data rates [12]. In contrast, network processors are optimized to handle data rates on the order of gigabits per second. Also, network processors usually do not use operating systems and thus do not incur these overheads. This availability of processing power is the premise for our PAD system.

Based on these technology trends, we can develop a system architecture that utilizes these advances to improve anomaly detection.

System Architecture

The key idea of our PAD system is to run multiple existing anomaly detection algorithms on multiple subsets of traffic in parallel. The design is based on utilizing the available processing resources in network systems to perform online anomaly detection processing for all algorithms and traffic classes in real time. The general architecture of parallel anomaly detection is illustrated in Fig. 1. Packet headers captured from the monitored link are classified into different subsets by the packet classifier, and the data pertaining to those subsets are then extracted. Specifically, we maintain packet and byte counts per observation interval. Each anomaly detection algorithm processes this data for each subset to find volume anomalies characterized by unexpected changes in traffic volume. The continuous anomaly metric output by each algorithm is further normalized. The normalized anomaly metrics for all algorithms for a particular subset are then aggregated to produce an anomaly score that represents the severity of the anomaly. Finally, a binary decision is made based on whether the anomaly score exceeds the threshold.

As our results show, using multiple anomaly detection algorithms on multiple traffic subsets improves the detection sensitivity and accuracy of the system. The main challenge in this process is to determine how the results from multiple algorithms can be combined effectively.



■ Figure 1. *System architecture. Observe that all algorithms process data for every monitored subset.*

Normalization and Aggregation of Anomaly Detection Information

The key technical challenge in our design is the aggregation of the information from multiple anomaly detection algorithms. Existing anomaly detection algorithms internally use a continuous metric, before applying a threshold and generating a binary output (i.e., a 1 for anomaly and 0 for no anomaly). One could choose to aggregate these binary outputs by using majority decision or similar binary functions. However, such a method is too coarse. Instead, our system aggregates the internal continuous anomaly metrics and applies the threshold as late as possible. The internal continuous metric produced by an algorithm depends on its characteristics: internal parameters and the technique used to process the data. Therefore, normalization of these metrics is necessary to ensure that every algorithm has equal influence on the final anomaly metric. The aggregation process reduces the amount of information for each traffic subset by combining the normalized outputs of all algorithms and providing a single indicator of an anomaly.

Notation

Let s be the total number of different traffic subsets S_i, $1 \le i \le s$, and a be the number of anomaly detection algorithms A_j, $1 \le j \le a$. During an observation interval $[t,t + \tau)$, where parameter τ is assumed to be a preset fixed time interval that decides the granularity of the data, let c_i be the packet count observed for the subset S_i, and $p_{i,j}$ be the prediction produced by each algorithm A_j based on the history of packet counts for the subset S_i. Then our system uses the metric $m_{i,j} = c_i/p_{i,j}$ for the observation interval $[t,t + \tau)$.

Normalization Scheme

Our system lays emphasis on detecting volume anomalies that are characterized by an increase in bit rates or packet counts. Some examples of such anomalies are DoS attacks, flash crowds, and port-scanning traffic. Each algorithm j outputs a prediction $p_{i,j}$ that depends on the characteristics and internal parameters used by the algorithm. Thus, normalization is necessary to ensure equal influence of each algorithm on the aggregate. We define a normalization function, N, that produces the normalized metric $n_{i,j}$, for algorithm j and subset i at time t. This function normalizes $m_{i,j}$ to the continuous interval [0,1], where 0 represents the condition of no anomaly and 1 represents anomaly. The normalization function is as follows:

$$n_{i,j} = N(m_{i,j}, \alpha_j) = \min(1, \max(0, 0.5 \cdot \alpha_j \cdot m_{i,j})). \quad (1)$$

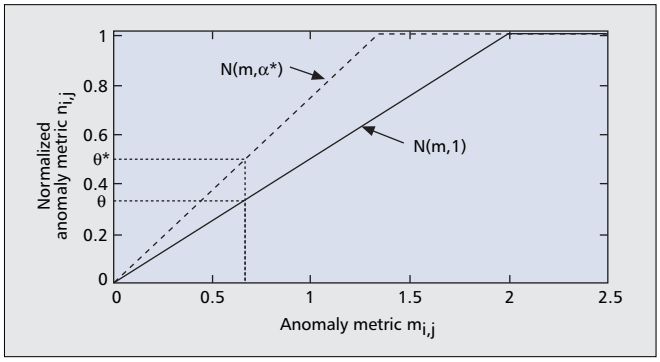
Parameter α_j determines the slope of the normalization function and is unique to a particular algorithm A_j. It is adjusted such that the boundary between the anomalies and no anomalies falls exactly in the middle of the range [0,1] at $\theta^* = 0.5$ for

$$\alpha_j^* = \frac{n^*}{\theta_j}.$$

(Refer to Fig. 2.) In other words, the normalization scheme seeks to transform the algorithm-specific thresholds, θ_j, to θ^*. This ensures that we do not have to use a system-specific threshold, and the final aggregated value can be compared with $\theta^* = 0.5$.

Aggregation Scheme and Anomaly Decision

During the observation interval $[t,t + \tau)$, for each subset S_i, the output of the normalization module is the normalized anomaly metric $n_{i,j}$ for every algorithm A_j. The aggregation module then uses the aggregation function G on these values to determine the final aggregated anomaly metric g_i for each subset i at time t. We have evaluated arithmetic mean, geo-



■ Figure 2. *Normalization function* N(m, α). α* *ensures that the boundary between anomalies and no anomalies occurs at* θ* = *0.5.*

Trace	Source	Trace duration	Anomaly type	Description
T1	Los Nettos Trace 4 [14]	439 s	TCP SYN flood	Persistent low packet rate traffic from five sources targeting a single victim IP on four ports.
T2	Los Nettos Trace 18 [14]	1057 s	TCP SYN	Persistent traffic of different intensities from large number of sources targeting a single port on seven victim IPs.
			TCP No-Flag	High intensity attack targeting a single IP on 65,530 ports from 788,820 sources.
			UDP flood	Persistent flooding of three ports on a single IP from five source IPs and 250 source ports.
T3	Los Nettos Trace 29 [14]	956 s	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 four victim IPs from a large number of sources.
T4	Code Red II [15]	248 s	TCP SYN portscan	Single IP scans HTTP port on a large number of victim IPs.
T5	MIT Lincoln Labs DDoS [16]	6167 s	TCP RST flood	Single IP flooded by a large number of spoofed source IPs.

■ Table 1. *Packet traces used in experiments.*

metric mean, median, minimum, maximum, and so on as aggregation functions; but the following function, which is the average of the maximum and mean as proposed by Evangelista *et al.* in [13], is the most effective:

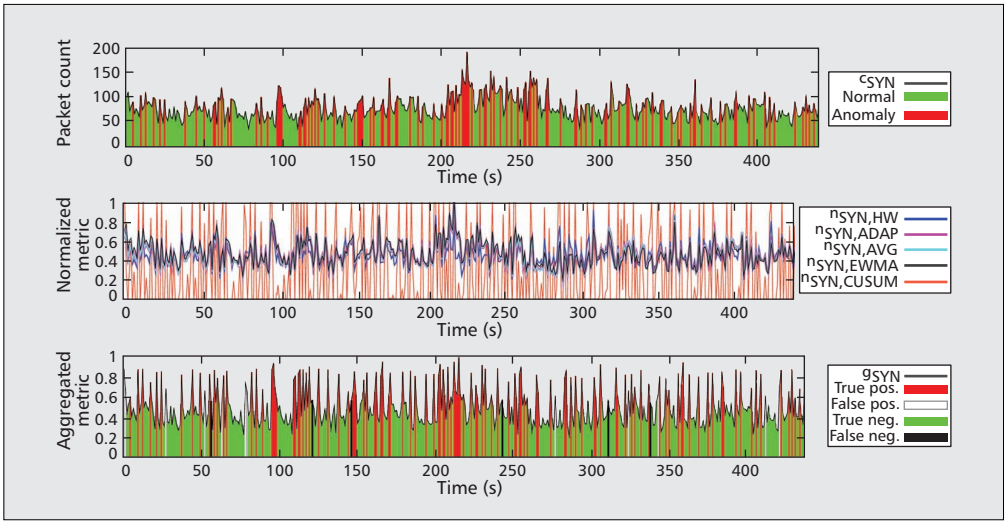
$$g_i = G(n_{i,1},\ldots n_{i,a}) = \frac{1}{2}\left(\left(\frac{1}{a}\sum_{j=1}^{a} n_{i,j}\right) + \max_j n_{i,j}\right). \quad (2)$$

A performance comparison of the various aggregation functions is presented later. Different algorithms have different characteristics and thus behave differently under different traffic conditions. For example, an algorithm may be sensitive to high-volume anomalies but may fail to detect low-volume floods. The mean aggregation functions work by assigning equal weights to the various algorithms independent of whether a particular algorithm is good or bad. An alternative approach is to determine which of the algorithms have poor performance under certain conditions and assign weights. The max function selects the algorithm that is most sensitive and produces the highest normalized value at any particular instance. Therefore, we use the max function in conjunction with the *arithmetic mean* to make sure that the aggregation function is robust in all cases, thus improving overall detection performance.

Finally, a binary decision is made to determine if there is an anomaly by comparing the aggregated anomaly metric with the threshold, θ^* as follows:

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

The binary decision is then reported to the operator.



■ Figure 3. *Behavior of metrics over time for trace 1.*

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

■ Table 2. *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.*

Prototype Implementation and Experimental Results

Our prototype implementation is based on an online measurement node [8] developed on an Intel IXP2400 network processor platform. The system makes use of multiple data path processors to classify packets into different subsets and update packet counts for each subset of traffic. These counts are stored in a shared SRAM. For each observation interval $[t,t+\tau)$, the XScale control processor reads these SRAM locations to obtain c_i. Each algorithm then outputs a prediction p_i based on the previously observed history of packet counts. The XScale control processor then computes the $m_{i,j}$, $n_{i,j}$, g_i, and r_i values. Our system uses $\tau = 1$ s. The summary of g_i and r_i can then be transmitted to the user via a socket interface. 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 and RST), and port numbers (as done in [7]). The $a = 5$ different anomaly detection algorithms we implement in our prototype are:

- HW: Holt Winter Forecasting Model [1]
- ADAP: Adaptive Threshold Algorithm [2]
- AVG: Average over Window [3]
- EWMA: Exponential Weighted Moving Average [4]
- CUSUM: Cumulative Sum Algorithm [5]

We test the ability of our system to identify anomalies in packet traces by replaying the traces shown in Table 1 using `tcpreplay`. This ensures that the PAD system receives realistic traffic and the results are reproducible. Each time interval of the trace is labeled manually. If during any interval of $\tau = 1$ seconds any packets match the anomaly description shown in Table 1, the interval is marked as anomalous. The labeling is compared to the anomaly decision r_i made by the PAD system. Figure 3 illustrates the working of the PAD system over time for trace T_1 which consists of a TCP SYN flood attack.

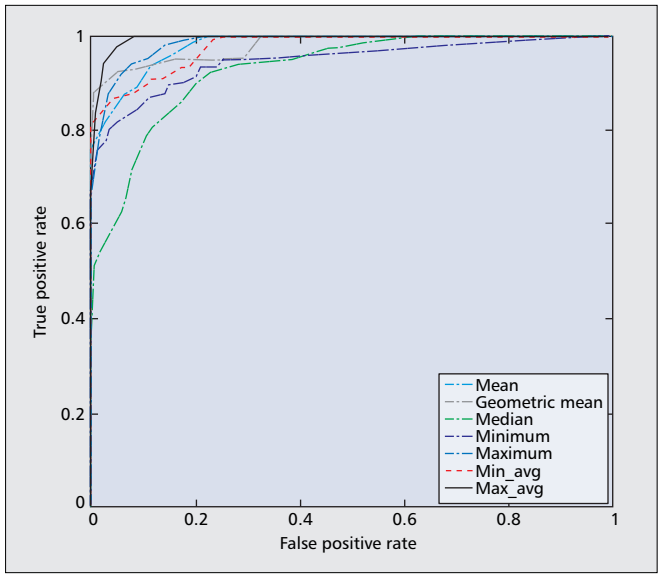
The top graph in Fig. 3 shows the TCP SYN packet count, c_{SYN}, and the manual classification into normal and anomalous time intervals. The middle graph shows the normalized anomaly metrics, $n_{SYN,j}$, provided by the five anomaly detection algorithms we have implemented on the prototype system. The bottom graph shows the aggregate anomaly metric, g_{SYN}. Also shown are the binary classification results, r_{SYN}, indicating by color if an anomaly is reported. For trace T_1, the PAD system yields a false positive rate of 4.4 percent and a false negative rate of 3.3 percent.

Normalization Parameters

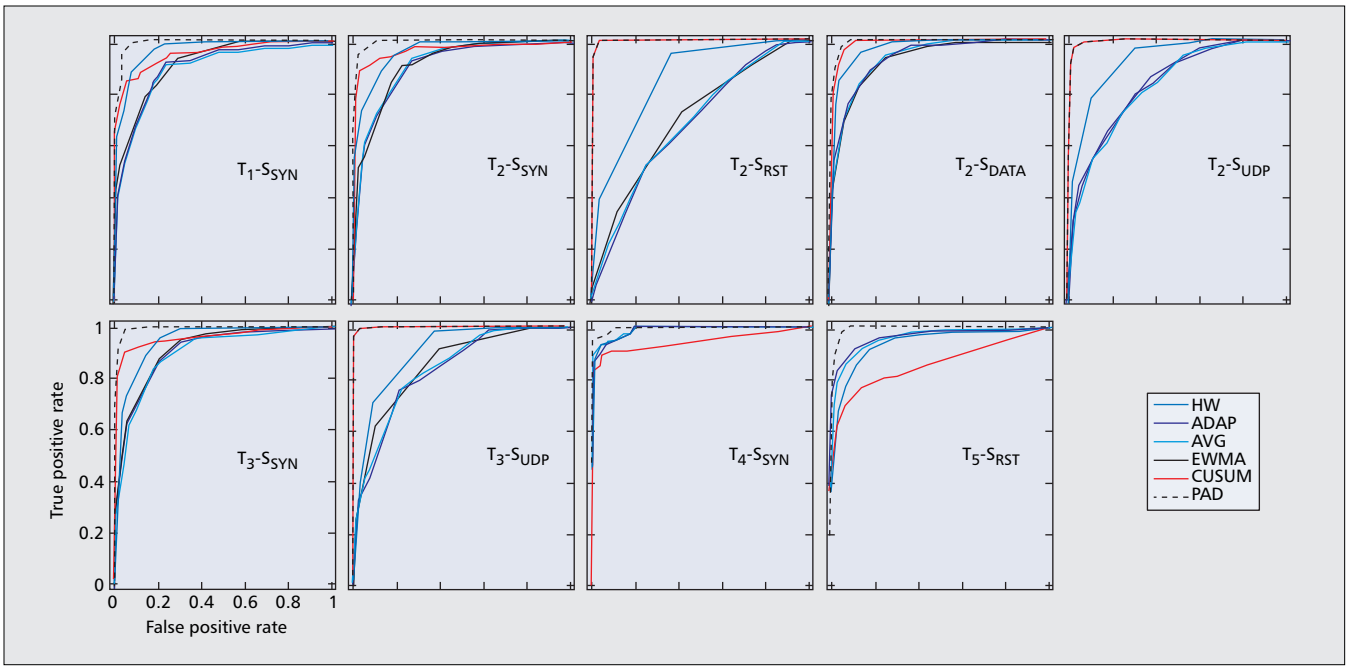
For each algorithm A_j, we need to determine the best normalization function shape parameter α_j^*. The results for each trace are shown in Table 2. We can observe that most parameter values for a given algorithm are similar across different traces. The aggregate parameters used in the PAD implementation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

Detection Performance of PAD

To prove that the aggregated anomaly metric indeed performs better than any individual algorithm in terms of detection performance, we compare the receiver operating characteristic (ROC) curves for the algorithms and the PAD system. A ROC curve is a graphical plot of the true positive rate (y-axis) vs. the false positive rate (x-axis) for a binary classifier (i.e., an anomaly detection algorithm) as its threshold values (i.e., θ) is varied. An ideal binary classifier will yield a point in the upper left corner of the ROC space, which represents a 100 percent true positive rate and a 0 percent false positive rate. Thus, when comparing the detection performance of algorithms, the



■ Figure 4. *ROC curves comparing detection performance for different aggregation Functions for Trace 1. Min_avg is the average of the minimum and average. Max_avg has the best performance and is used by PAD.*



■ Figure 5. *ROC curves comparing detection performance for all traces and all anomalous traffic subsets.*

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
PAD		.044/.033	.043/.036	.027/.000	.052/.025	.047/.009	.055/.000	.018/.011	.000/.053	.066/.004
Improvement		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%

■ Table 3. *False positive rate/false negative rate of individual algorithm and PAD system.*

algorithm that yields the maximum distance between its ROC curve and the diagonal is considered to perform better than other algorithms. Figure 4 compares the performances of the various aggregation functions we have evaluated. Clearly, the aggregation function shown in Eq. 2 outperforms the other functions evaluated.

Figure 5 shows the ROC curves for all individual algorithms (using the best possible parameters shown in Table 2) and the ROC curve for the PAD system using aggregated parameters. Observe that the PAD system outperforms any individual algorithm, even though a single set of parameters is used for all traces.

Table 3 shows the false positive rates and false negative rates for the PAD system as well as individual algorithms. The entries shown in bold are the lowest false positive or false negative rates achieved by any algorithm. Observe that the PAD system provides a lower false positive and false negative rates than any given algorithm, the only exception being the subset S_{SYN} in trace T_4. The difference in these rates is shown as a percentage in the last row. Trace T_4 consists of Code Red II scans with very low packet rates (in tens). This is barely distinguishable from normal traffic resulting in most algorithms having very poor detection performance for this trace. However, Holt-Winter forecasting performs exceptionally well and detects every instance of the anomaly. CUSUM has the worst detection performance of all algorithms. Since the averaging term in the aggregation function assigns equal weights to all the algorithms, the other algorithms tend to have some influence on the aggregate and thus the deterioration in performance of the aggregate for trace T_4. However, the aggregate still performs better than the other algorithms. The median improvement is a 10.0 percent lower false positive rate and a

37.9 percent lower false negative rate. Thus, the PAD system performs significantly better than any single anomaly detection algorithm.

Conclusion

We have discussed the design of a parallel anomaly detection system that uses multiple existing anomaly detection algorithms in parallel to detect traffic anomalies in real time. The prototype implementation on an Intel IXP2400 network processor is evaluated using real-world attack traces. The main idea of our parallel anomaly detection system is to implement multiple existing anomaly detection algorithms on multiple traffic subsets in parallel. The availability of advanced processing resources in network systems helps us achieve this task. Using multiple anomaly detection algorithms improves the accuracy and detection sensitivity of an anomaly detection system, and running these algorithms on multiple subsets of traffic in parallel allows us to detect subsets that are causing the anomaly. Our experiments on real-world attack traces show that parallel anomaly detection achieves lower false positive and false negative rates than any individual algorithm.

Acknowledgment

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

References

[1] J. D. Brutlag, "Aberrant Behavior Detection in Time Series for Network Monitoring," *Proc. 14th Sys. Admin. Conf.*, New Orleans, LA, Dec. 2000, pp. 139–46.
[2] V. A. Siris and F. Papagalou, "Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks," *Proc. IEEE GLOBECOM*, Dallas, TX, Nov. 2004, pp. 2050–54.
[3] C. Schwarzer, "Prediction and Adaptation in a Traffic-Aware Packet Filtering Method," Master's thesis, EPFL, Lausanne, Switzerland, Mar. 2006.
[4] S. Deshpande *et al.*, "A Statistical Approach to Anomaly Detection in Interdomain Routing," *Proc. 3rd Int'l. Conf. Broadband Commun., Netw., and Sys. (BROADNETS)*, San Jose, CA, Oct. 2006.
[5] H. Wang, D. Xhang, and K.G. Shin, "Change-Point Monitoring for the Detection of DoS Attacks," *IEEE Trans. Dependable Secure Comp.*, vol. 1, no. 4, Oct. 2004, pp. 193–208.
[6] P. Barford *et al.*, "A Signal Analysis of Network Traffic Anomalies," *Proc. 2nd ACM SIGCOMM Wksp. Internet Measurement* , Marseille, France, Nov. 2002, pp. 71–82.
[7] Y. Gu, A. McCallum, and D. Towsley, "Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation," *Proc. 5th ACM SIGCOMM Conf. Internet Measurement*, Berkeley, CA, Oct. 2005.
[8] T. Wolf *et al.*, "An Architecture for Distributed Real-Time Passive Network Measurement," *Proc. 14th Annual Meeting IEEE/ACM Int'l. Symp. Modeling, Analysis, and Simulation Comp. and Telecommun. Sys.*, Monterey, CA, Sept. 2006, pp. 335–44.
[9] S. Shanbhag and T. Wolf, "Massively Parallel Anomaly Detection in Online Network Measurement," *Proc. 17th IEEE ICCCN*, St. Thomas, U.S. Virgin Islands, Aug. 2008.
[10] S. Shanbhag and T. Wolf, "Evaluation of an Online Parallel Anomaly Detection System," *Proc. IEEE GLOBECOM*, New Orleans, LA, Dec. 2008.
[11] T. Wolf, "Challenges and Applications for Network-Processor-Based Programmable Routers," *Proc. IEEE Sarnoff Symp.*, Princeton, NJ, Mar. 2006.
[12] S. Bunga and T. Wolf, "A Characterization of High-Performance Network Monitoring Systems and Workloads," *Proc. IEEE Wksp. High Perf. Switching and Routing*, Brooklyn, NY, May 2007.
[13] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, "Data Fusion for Outlier Detection Through Pseudo-ROC Curves and Rank Distributions," *Proc. Int'l. Joint Conf. Neural Net.*, Vancouver, BC, July 2006, pp. 2166–73.
[14] A. Hussain, J. Heidemann, and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," *Proc. SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003, pp. 99–110.
[15] D. Moore, C. Shannon, and J. Brown, "Code-Red: A Case Study on the Spread and Victims of an Internet Worm," *Proc. 2nd ACM SIGCOMM Wksp. Internet Measurement*, Marseille, France, Nov. 2002, pp. 273–84.
[16] R. P. Lippmann *et al.*, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation," *Proc. DARPA Info. Survivability Conf. Expo.*, Hilton Head, SC, vol. 2, Jan. 2000, pp. 12–26.

Biographies

SHASHANK SHANBHAG (sshanbha@ecs.umass.edu) is a doctoral student at the University of Massachusetts, Amherst. He received his B.S. degree from Visvesvaraya Technological University, India, and his M.S. degree from the University of Massachusetts, Amherst. His research interests span the areas of network measurement, anomaly detection, and next-generation Internet architectures.

TILMAN WOLF (wolf@ecs.umass.edu) is an associate professor in the Department of Electrical and Computer Engineering at the University of Massachusetts, Amherst. He received a D.Sc. in computer science from Washington University in St. Louis in 2002. His research interests include network processors, their application in next-generation Internet architectures, and embedded system security. He has been active as a program committee member and an organizing committee member of numerous professional conferences, including IEEE INFOCOM and ACM SIGCOMM.