Adaptive Re-Configuration of Network Monitoring Applications

Falko Dressler

Autonomic Networking Group
Dept. of Computer Science 7
University of Erlangen-Nuremberg, Germany

Abstract. The rapid growth of network traffic and available bandwidth forced additional research in the area of network monitoring for network security solutions as well as for accounting and traffic engineering applications. Autonomic networking solutions are considered to counteract identified scalability issues. In this paper, we focus on the amount of measurement data obtained by a monitoring probe that has to be transmitted to and analyzed by an analyzer, e.g. an attack detection system. We question the possibility of re-configuring the monitoring part in order to adapt to the computational resources of the analyzer as well as to the current network behavior. We propose an adaptive mechanism for preventing single systems from being overloaded while providing as much monitoring information as possible for post-processing. Using two simulation models, we verified the applicability of our approach and show the speedup gained by adapting the parameters of the monitoring solution.

1 Introduction

Monitoring of high-speed communication networks has become a major research area during the last years. Driven by the needs of network security installations but also by the demands of accounting and charging systems, network monitoring methods and techniques have been standardized by several organizations, first of all by the IETF (Internet Engineering Task Force). In order to cope with the steadily increasing amount of data and the very high bandwidths in nowadays backbone networks, the reduction of monitor data is a key issue for successful monitoring solutions. Developed methodologies include flow accounting, i.e. statistical measures of end-to-end network flows described for example by the IP-5-tuple, and packet sampling, i.e. selection and analysis of a few selected packets.

In the following, we concentrate on attack detection as one of the major applications of network monitoring. Usually, monitoring entities are employed to collect flow or packet data and transmit them to an analyzer, i.e. the detection system. In figure 1, a simple scenario is depicted. Several monitoring probes (P) are used to obtain traffic statistics as well as to capture selected packets. This information is forwarded to accociated attack detection systems (IDS), which in turn analyze the data and close the loop by configuring firewall systems to

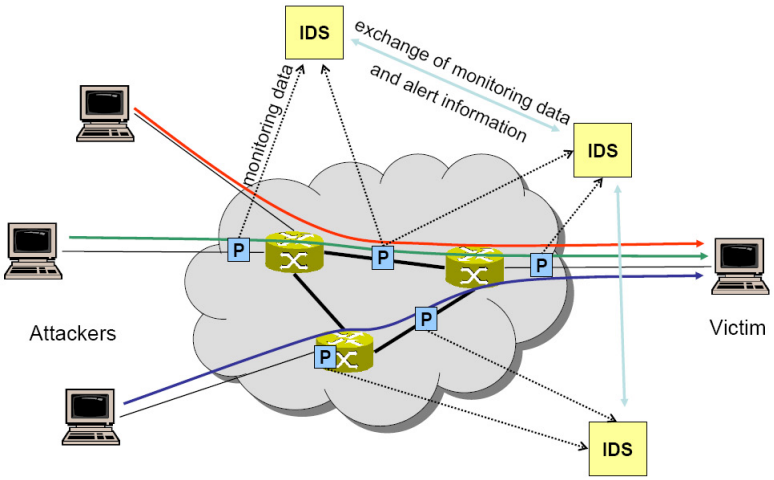

Fig. 1. Network security scenario showing monitoring probes (P) and associated attack detection systems (IDS)

counteract identified attacks. Similar scenarios can be thought of for accounting or traffic engineering purposes.

We analyzed the scalability of such monitoring approaches using appropriate simulation models. In this paper, we propose an adaptive mechanism for reducing the amount of monitoring data, or, to be more precise, to adjust the load of involved systems in the entire network security system. The primary goal is to monitor as much packet data as possible in order to achieve most accurate detection results. A threshold is given by the processing capacity of the involved systems, thus, an upper bound is defined. Additionally, this threshold depends on the many parameters including the form and amount of monitoring data.

We address the described goal by using two separate feedback loops as inspired by similar solutions found in nature. The research area of bio-inspired networking has already shown application scenarios for new and unconventional solutions [1]. Most biological methods for self-organization depend on two mechanisms: a positive feedback loop for amplification and a negative feedback loop for regulation [2, 3]. Using similar mechanisms, we built an adaptive feedback mechanism for a self-organized network monitoring scenario.

In section 2, the state-of-the-art of network monitoring is summarized followed by a description of our proposed adaptive control mechanism in section 3. The simulation model and the accomplished simulations are discussed in section 4. Some conclusions summarize the paper.

2 Network Monitoring

Because available bandwidths grow much faster than the processing speed of the monitoring probes and subsequent analyzers, solutions have been developed that allow reducing the processing requirements at the analysis. The primary idea behind all these concepts is to split the monitoring and the subsequent analysis into two independent tasks. This became possible because not all packet information is required for network analysis (whether for accounting or security reasons).

The first concept developed in this context is netflow accounting. The key idea is to store information about netflows and the corresponding statistics instead of individual packets. Thereby, a netflow is defined as a unidirectional connection between two end systems as defined by the IP 5-tupel (protocol, source IP address, destination IP address, source port, destination port). Doing this, a single measurement data set contains information of one up to several thousands of individual packets. For the transmission of the monitoring data to an analyzing system, a special protocol was developed called netflow.vX, where X stands for the version number. Netflow.v9 is the latest version and standardized by the IETF [4]. Its successor is IPFIX (IP flow information export). It was developed by the corresponding IPFIX working group at the IETF and provides sufficient information for a distributed deployment [5, 6]. First implementations are available that support netflow.v9 as well as IPFIX, e.g. nProbe [7] and Vermont [8].

Even if this methodology works well under normal conditions (usual connections consist of about 7.7 packets per flow [9]), there is a major problem during DDoS attacks. Usual attacks are using forged IP addresses, different ones for each attack packet, which results in the creation of individual flows per packet. Thus, in such an attack situation, netflow accounting or IPFIX do not scale well, i.e. they overflow the connection between the monitoring probe and the intrusion detection system (regardless of the computational expense at the analysis). To cope with this problem, recently an aggregation mechanism was introduced [10] that allows to aggregate individual flows into so called meta-flows. This aggregation mechanism allows a free scaling of the amount of monitoring data and provides the basic functionality to build adaptive self-optimizing netflow accounting solutions. Nevertheless, even more functionality is required to cope with the growing amount of network traffic. In the next section, we discuss an approach for adaptively configuring all involved components in the network security scenario.

3 Adaptive Re-Configuration

In the context of this section, we present an adaptive re-configuration scheme for the monitoring environment depending on the current network behavior. We claim to overcome general scalability problems as well as to prevent overload situations at the attack detection systems. The final goal is to show the practicability of our self-organizing monitoring architecture. This architecture fulfills its

role to collect as much as possible packets concentrating on the really necessary data sets. In the following, first, the problem is described including the primary objectives. Secondly, a basic model is provided which fulfills the requirements and builds a basis for the simulation model used for the performance analysis.

3.1 Problem Description and Objectives

The most important issue is to prevent the attack detection system from becoming overloaded by monitoring probes, which are sending too much information. Primarily, two reasons lead to this objective. We need to prevent the detection system from becoming a target itself as well as to increase the availability of the overall system. Even though in a distributed attack detection, each subsystem can perform the detection autonomously, a single overloaded system might miss important packets that build a primary attack accompanied by a large amount of meaningless packets. To achieve this goal, an autonomous behavior of each subsystem is considered. In this paper, we concentrate on the adaptive re-configuration. Self-optimization is an important requirement for autonomous systems in general. In this context, we understand self-optimization as the ability to optimize the detection quality. This can be achieved by exchanging information about already identified attacks or suspicious network connections and also by statistically forwarding parts of collected data packets and network statistics to neighboring probes. Additionally, the autonomously working entities must be capable to adapt to a changing environment. This adaptation, typically realized by re-configuration of several runtime parameters, comprises of changes in the resource management and in the configuration of tasks and processes. Re-configuration is also required in the case of resource shortages. The attack detection must be modified by selecting algorithms and parameters which require less memory, while typically resulting in a lower detection rate (graceful degradation).

3.2 Model and Solution

In figure 2, a model is shown that represents the considered architecture including all necessary components. While focusing on the monitoring part, we want to reduce, or more precisely, to adapt the rate of packets sent to the attack detection system. This can be done in at least three ways as shown in the following.

Compression/encoding – Monitored packet data can be encoded in a way reducing the number of transmitted bytes to the absolute minimum. Netflow accounting / IPFIX can help at this place, especially if aggregation mechanisms are employed. Nevertheless, during a DDoS attack, nearly every packet represents a separate flow and, therefore, the speedup is questionable in this case. Therefore, methodology based approaches are required.

Blacklists/whitelists – Usually, blacklists represent hosts involved in an attack and whitelists represent legitimate traffic. Blacklists in packet filtering systems, i.e. firewalls, represent a functionality having two advantages. First, the packets

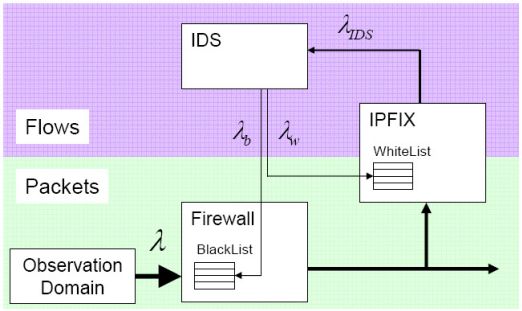

Fig. 2. Basic model for adaptive re-configuration including the main components and the control and data flows

are prevented from reaching the systems under attack and, secondly, these packets no longer reach the monitoring system. Therefore, the data rate being sent the monitoring systems to the attack detection systems is reduced. Additionally, whitelists can be used at the monitoring probes to reduce the amount of data transmitted to the analyzing systems.

Feedback – Finally, a methodological approach can be developed that is based on the parameterization being adapted to the current situation in the network. The attack detection system can communicate its current load to the monitoring probes. These can adapt a number of parameters, usually timeouts, based on the number of packets received from the network, the number of packets reported to the detection systems, and the current load of the analyzers.

3.3 Methodology

As mentioned in section 1, bio-inspired methodologies can be used to create appropriate feedback loops for adapting system parameters. In our system, we want to adapt the parameters of the monitoring environment depending on the load of the detection system and of the current network behavior. Usually, two kinds of feedback loops are used in combination: positive feedback for short-term amplification and negative feedback for long-term regulation. Both loops are depicted in figure 2. The intrusion detection reports detected attacks to the firewall that in turn blocka this traffic and, therefore, reduces the number of packets that have to be monitored. Additionally, the detection system reports legitimate traffic to the monitor. This monitor stops reporting on packets belonging to these flows and, therefore, reduces the number of packets that have to be analyzed. Obviously, both configurations cannot be permanent. Sources sending legitimate traffic might begin to send attack packets at any time. Also, previously attacking machines might become 'corrected and should not be starved by our firewalls.

The adaptation is done using the following formulas for calculating appropriate timeouts. TO_{black} corresponds to the firewall system and TO_{white} to the monitoring probe. The input parameters are summarized in table 1.

λ	packet arrival rate
λ_b	arrival rate of 'black' packets
λ_{bi}	arrival rate of 'black' packets belonging to the i-th flow
λ_w	arrival rate of 'white' packets
λ_{IDS}	maximum capacity (throughput) of the IDS

Table 1. Input parameters

$$TO_{black} = C_1 \underbrace{\frac{\lambda_{bi}}{\lambda}}_{t_1} + C_2 \left(\underbrace{\frac{\lambda_b}{\lambda}}_{t_2} + \underbrace{\frac{\lambda}{\lambda_w}}_{t_3} + \underbrace{\frac{\lambda}{\lambda_{IDS}}}_{t_4} \right) \tag{1}$$

$$TO_{white} = C_3 \left(\underbrace{\frac{\lambda}{\lambda_{IDS}}}_{t_5} + \underbrace{\frac{\lambda}{\lambda_b}}_{t_6} \right) \tag{2}$$

The single terms ($t_1..t_6$) are discussed in the following. In principle, all terms belonging to one timeout are summed up and scaled by a constant ($C_1..C_3$). In our simulation experiments, we tried to find appropriate values for these constants. In a next step, the constants themselves can be adapted to the current scenario.

$t_1 = \frac{\lambda_{bi}}{\lambda}$ – Ratio of the i-th attack flow to the overall attack rate. Used for penalizing previously discovered attack flows. This term must be scaled separately using C_1 because it is usually very small.

$t_2 = \frac{\lambda_b}{\lambda}$ – Similar to t_1 but it defines the ratio of arriving attack traffic to the overall throughout. The larger t_2 is, the more aggressive the attack.

$t_3 = \frac{\lambda}{\lambda_w}$ – This term describes the safety of the arriving traffic. The larger the amount of 'white packets, the smaller the requirement for large timeouts at the firewall.

$t_4 = \frac{\lambda}{\lambda_{IDS}}$ – This term is a measure for the overload of the attack detection system.

$t_5 = \frac{\lambda}{\lambda_{IDS}}$ – Similar to t_4 but used at the monitor instead of the firewall system.

$t_6 = \frac{\lambda}{\lambda_b}$ – Similar to t_3 but defining the risk of arriving packets.

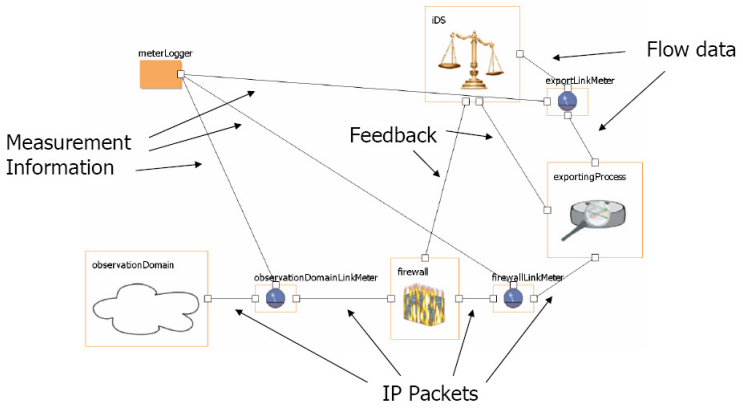

Fig. 3. Simulation model including message types on the communication channels: a) IP packets, b) flow data, c) feedback, d) measurement information

Each term is represented by a fraction of two rates. Therefore, approriate countermeasures have to be applied to prevent a 'division by zero. Therefore, each of the terms has to be read in the following way:

$$t_i = \begin{cases} \frac{\lambda_j}{\lambda_k} & \text{if } \lambda_k \neq 0 \\ 0 & \text{if } \lambda_k = 0 \end{cases} \tag{3}$$

4 Modeling and Simulation

In order to show the potentials of the described feedback-based adaptation, we executed a set of simulations showing the reduction of packet data that is to be received and processed by the attack detection systems. For a reasonable simulation, we decided to use 'trace-based input modeling. We accumulated several traces in front of our workgroup server and directly at the Internet gateway of our university. The data presented in this paper rely to the utilization seen at the university border gateway.

The simulation model is depicted in figure 3. This model was implemented using AnyLogic, a well-established discrete-event simulation tool. The primary goal was to find adequate values for the scaling factors C_1, C_2, and C_3. In the following, selected simulation results are presented and discussed. In multiple experiments, we evaluated candidate values for the scaling factors. The parameters used for the presented simulation results were $C_1 = 9 * 10^8$, $C_2 = 236$, and $C_3 = 120$. The used simulation parameters are summarized in table 2.

As already mentioned, we monitored the traffic at the border gateway of our university. The utilization was quite constant (58.6kpps $\pm$ 1.8kpps) as shown in figure 4, left. Due to few attacks, the firewall removes only few flows (around

C_1	$9 * 10^8$
C_2	236
C_3	120
λ_{IDS}	0.6
E[white]	0.1
E[black]	0.01

Table 2. Simulation parameters

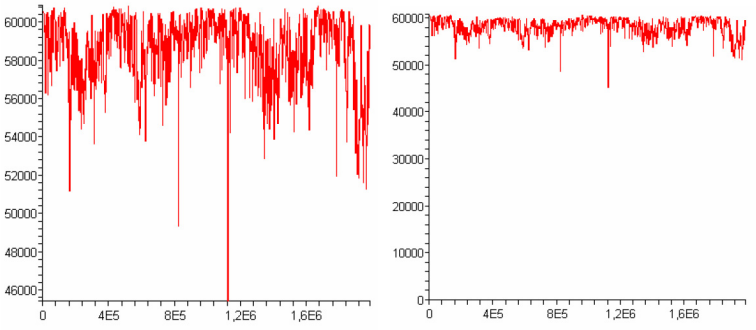

Fig. 4. Observed traffic rate: global (left) and at the monitor (right)

0.5%). Therefore, the packet rate arriving at the monitor is similar but shaped (58.2kpps ± 1.8kpps, figure 4, right).

Even though the number of discarded packets at the firewall is very small, this is primarily a result of the small timeout of each entry. The attack detection system is not overloading, therefore, it may analyze detected attacks over and over again. The behavior of the firewall is depicted in figure 5. In steady-state, around 6000 blacklist entries exist with an average timeout of 260s.

Due to the key ideas of netflow accounting, the summarization into flows, the number of exported data sets is quite low (figure 6). Nevertheless, there is an enormous advantage compared to mechanism without adaptation. In this case, the timeouts have to be predefined and can only correspond to an assumed behavior of the network.

Finally, the numbers of whitelist entries and the corresponding timeouts have to be examined. As shown in figure 7, in steady-state, the number of whitelist entries is around 60000 and the average timeout is 320s. The number corresponds to the detection quality of legitimate traffic which is about ten times higher then for attacks. The timeout oscillates around the requested value.

5 Conclusion

We presented a method for adaptive parameter control in network monitoring environments. Using an amplifying positive feedback loop and a protecting neg-

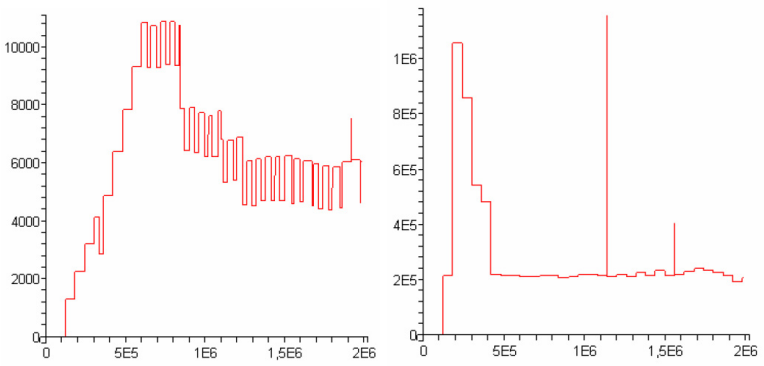

Fig. 5. Number of blacklist entries (left) and blacklist timeouts TO_{black} (right)

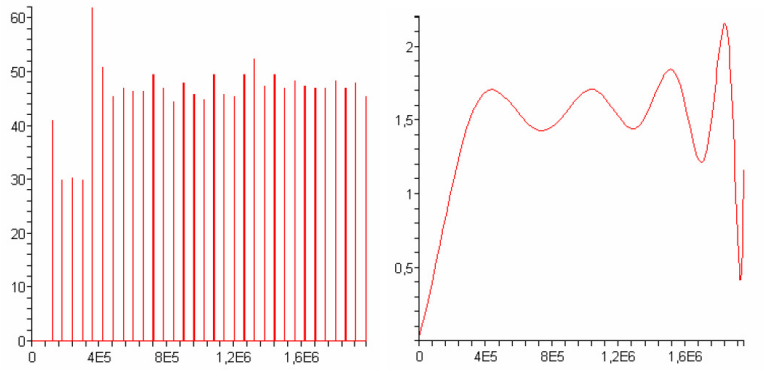

Fig. 6. Number of exported data sets: discrete (left) and averaged (right)

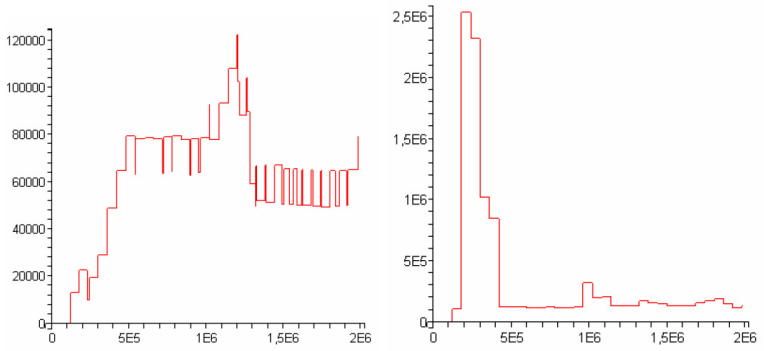

Fig. 7. Number of whitelist entries (left) and whitelist timeouts TO_{white} (right)

ative feedback loop, we achieved a self-organizing behavior of the overall system. Especially the capabilities of the adaptation to reflect the local needs of an analyzing system in combination to the observation of the environment, i.e. the arrival rates at each subsystem make this approach useful for most monitoring scenarios. In a next step, we will evaluate the algorithm in an experimental setup to compare this evaluation to the simulation results.

Acknowledgements

This work is part of a collaborative research project conducted together with Georg Carle and Gerhard Münz from the University of Tübingen, Germany. We also thank our students Alexander Lochschmied and Jing Chen for contributing to this work.

References

1. Krüger, B., Dressler, F.: Molecular processes as a basis for autonomous networking. IPSI Transactions on Advances Research: Issues in Computer Science and Engineering **1** (2005) 43–50
2. Dressler, F.: Bio-inspired mechanisms for efficient and adaptive network security mechanisms. In: Dagstuhl Seminar 04411 on Service Management and Self-Organization in IP-based Networks, Schloss Dagstuhl, Wadern, Germany (2004)
3. Dressler, F.: Efficient and scalable communication in autonomous networking using bio-inspired mechanisms - an overview. Informatica - An International Journal of Computing and Informatics **29** (2005)
4. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational) (2004)
5. Quittek, J., Bryant, S., Meyer, J.: Information Model for IP Flow Information Export. Internet-Draft, draft-ietf-ipfix-info-09.txt (2005)
6. Claise, B.: IPFIX Protocol Specifications. Internet-Draft, draft-ietf-ipfix-protocol-19 (2005)
7. Deri, L.: nprobe: an open source netflow probe for gigabit networks. In: TERENA Networking Conference (TNC 2003), Zagreb, Croatia (2003)
8. Dressler, F., Carle, G.: History - high speed network monitoring and analysis. In: 24th IEEE Conference on Computer Communications (IEEE INFOCOM 2005), Miami, FL, USA (2005)
9. Lee, T.H., Wu, W.K., Huang, T.Y.W.: Scalable packet digesting schemes for ip traceback. In: IEEE International Conference on Communications, Paris, France (2004)
10. Dressler, F., Sommer, C., Münz, G.: IPFIX Aggregation. Internet-Draft, draft-dressler-ipfix-aggregation-02.txt (2005)