# Self-Organized Network Security Facilities based on Bio-inspired Promoters and Inhibitors

Falko Dressler

Autonomic Networking Group
Dept. of Computer Science 7,
University of Erlangen, Germany
dressler@informatik.uni-erlangen.de

**Summary.** Self-organization techniques based on promoters and inhibitors has been intensively studied in biological systems. Promoters enable an on-demand amplification of reactions to a particular cause. This allows to react quickly with appropriate countermeasures. On the other hand, inhibitors are capable of regulating this uncontrolled amplification by suppressing the reaction. In this paper, we demonstrate the applicability of these mechanisms in a network security scenario consisting of network monitoring elements, attack detection, and firewall devices. Previous work identified most existing detection approaches as not suitable for high-speed networks. This problem can be alleviated by separating the methodologies for network monitoring and for subsequent data analysis. In this paper, we present an adaptation algorithm that allows to manage the individual configuration parameters in order to optimize the overall system. We show the advantages of self-regulating techniques based on promoters and inhibitors that lead to maximized security and that gracefully degradate in case of overload situations. We created a simulation model to verify the algorithms. The results of the conducted simulations encourage further studies in this field.

## 1 Introduction

Network security facilities usually include mechanisms for attack detection and appropriate countermeasures. If employed in high-speed networks, a third component is added in order to cope with the steadily increasing amount of data and the very high bandwidths in nowadays backbone networks: network monitoring. All these components interoperate in a distributed environment. Driven by network security demands, network monitoring methods and techniques have been developed and standardized by several organizations, first of all by the IETF (Internet Engineering Task Force). Figure 1 depicts a typical scenario. Monitoring probes are used to obtain traffic statistics and to capture selected packets. This information is forwarded to associated attack detection systems, e.g. intrusion detection systems (IDS), which in turn ana-

lyze the data and close the loop by configuring firewall systems to counteract identified attacks. The same figure also introduces possible extensions to distributed attack detection by interconnecting autonomous IDS systems to share information about ongoing attacks and legitimate traffic. In this scenario, the efficiency of the security mechanisms strongly depends on the performance of all involved components. In the following, we concentrate on attack detection as one of the major applications of network monitoring, especially DDoS (distributed denial-of-service) attacks are concerned [3, 20]. Such attack detection entities rely on the quality of received monitoring data, i.e. their timeliness, correlation, and completeness. 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. In a distributed attack detection environment, each subsystem can perform the detection autonomously. Nevertheless, a single overloaded system can miss packets of a primary attack accompanied by a large amount of meaningless packets.

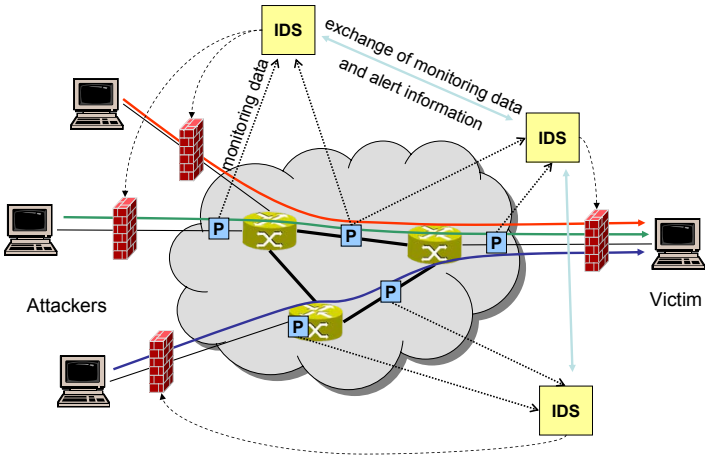


**Fig. 1.** Network security scenario consisting of monitoring probes, associated attack detection systems, and firewall devices

Searching for methods available to reduce the amount of monitoring data, two solutions can be found: flow monitoring and packet sampling. Flow monitoring [1] provides statistical measures of packet flows described by common properties, e.g. by the IP-5-tuple [21]. Depending on the specification of the flow characteristics, the necessary bandwidth to transport all monitoring data can be successfully reduced at cost of granularity and content. On the other hand, packet sampling is used for the selection and analysis of a few selected

packets [13]. Filter are used for deterministic packet selection while statistical operations allow to select packets based on content-independent measures. All these mechanisms must be properly configured in order to achieve an optimal result. In summary, the scalability of the overall system strongly depends on three parameters: the current network load, the amount of flows that was successfully identified to be malicious or legitimate, and the configuration of the involved systems. Similar observations hold for the reliability in terms of overload prevention of subsystems, e.g. monitoring probes or attack detection systems.

Turning to nature, we figured out several commonalities between the structure of organisms and computer networks [9]. This is also true for the cellular signaling pathways and data communication, thus promising high potentials for computer networking in general and adaptive network security in particular [18]. Based on these investigations, we started to adapt mechanisms known from molecular biology to enable a self-organized operation and control in network security scenarios.

We developed and analyzed an adaptive mechanism [6,8] for reducing the amount of monitoring data to adjust the load of involved attack detection systems. 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. Technically, this threshold depends on the many parameters including the form and amount of monitoring data. In order to achieve this goal, we created two separate feedback loops as inspired by similar solutions found in nature [7]. These feedback loops represent promoter / inhibitor functions, i.e. they either stimulate monitoring probes to send dare, i.e. higher quality data, or they suppress this amplification effect if the detection modules approach their maximum capacity.

The following general objectives are addressed with our adaptive reconfiguration scheme. *Self-maintenance* relies on the adaptation of configuration parameters depending on the environmental conditions. The autonomously working entities must be able to adapt to changing environmental conditions. *Self-healing* mechanisms respond to system failures. For example, on-demand re-configuration is required in the case of resource shortages. *Self-optimization* refers to the overall detection quality. This can be achieved by exchanging information about identified attacks or suspicious network connections and also by statistically forwarding parts of collected data packets and network statistics to neighboring probes.

We created appropriate simulation models to analyze the scalability of the developed approaches. Basically, we implemented the behavior of monitoring, firewall, and attack detection systems. In order to allow practically significant simulations and to easily compare different configurations, we used previously monitored data for trace-driven input modeling. We studied the configuration and possible adaptation of individual subsystems to increase the scalability and reliability of the overall system. It turned out that the dynamic recon-

figuration depending on the current network behavior is possible without any global control, i.e. we achieve an optimized system behavior at all times.

The rest of the paper is organized as follows. Sections 2 and 3 reflect the state-of-the-art in network monitoring and present an overview to the investigates biological mechanisms, respectively. The self-organizing adaptive control scheme is depicted in section 4. The used simulation model including the discussion of selected results is presented in section 5. Some conclusions in section 6 summarize the paper.

## 2 Network Monitoring and Attack Detection

### 2.1 High-speed network onitoring

In this section, a general overview to monitoring solutions is provided. Due to the fact that 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 stage. The primary idea behind all these concepts is to split the monitoring and the subsequent analysis into two independent tasks and to discard as much data as possible at the monitoring stage. The first concept that was developed in this context is flow monitoring. The key idea is to store information about flows and the corresponding statistics instead of individual packets. Thereby, a flow is defined as a unidirectional connection between two end systems as defined by common properties, e.g. the IP 5-tuple (protocol, source IP address, destination IP address, source port, destination port). Using flow monitoring, a single measurement data set contains information of one up to several thousand individual packets. For the transmission of the monitoring data to an analyzing system, a special protocol was developed: Netflow.v9 [4]. Its successor, the IPFIX (IP flow information export) protocol provides sufficient information for a distributed deployment [5, 23]. Even if this methodology works well under normal conditions (usual connections consist of about 7.7 packets per flow [19]), there is a major problem during DDoS attacks. Typical 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, flow monitoring does not scale well, i.e. it tends to 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 [10,12] 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 flow-based accounting solutions.

An additional problem is based on the basic principle of flow monitoring: the loss of payload information. For intrusion detection reasons, this information is often required and, therefore, the applicability of flow monitoring is

limited. To support the selection of single but complete packets and transporting them to an analyzer, PSAMP (packet sampling) was developed [14, 26]. It allows the free combination of filters and samplers. Filters are used for deterministic packet selection based on matching fields in the IP packet. Samplers are statistical algorithms that select packets using particular sampling algorithms, e.g. count-based. PSAMP therefore allows to monitor and to export complete packets providing sufficient information for intrusion detection. Additionally, the sampling algorithms, filters, and parameters can be freely defined and re-configured allowing a full-adaptive behavior. The monitoring can be optimized to provide all required data to the analyzer and no more than it is able to process.

To conclude the overview to network monitoring solutions, some strengths and disadvantages of flow accounting and packet sampling are summarized: Flow monitoring provides strong data compression in general. Unfortunately, during ongoing attacks, the number of flow records easily reaches the number ob observed packets and there is no possibility for signature-based attack detection due to the missing payload information. In case of packet sampling, the reduction depends only on the sampling algorithm, i.e. on then current configuration. The methodology-inherent loss of packet information makes the usage of such approaches in accounting and charging scenarios difficult. Therefore, even more functionality is required to cope with the growing amount of network traffic. In the next section, we discuss an approach for adaptively reconfiguration of involved components in the network security scenario.

### 2.2 CATS - Attack Detection using Cooperating Autonomous Detection Systems

The objective of this section is to describe an approach for attack detection using cooperative autonomous detection systems. This system, CATS [11], is one of the first approaches that provide an architecture clearly split into the mentioned three parts: monitoring, analysis, counteracting. Additionally, aspects of distributed operation are included into the monitoring part as well as into the analyzing part.

The architecture of an individual detection system is described in [11]. It consists of an outer part for network monitoring and an inner part for detection. The network monitoring part is responsible for capturing packets and flow statistics from the network, either directly using a connected network interface, or by employing monitoring probes and the standardized protocols IPFIX and PSAMP. This part also performs necessary preprocessing of the gathered data, such as packet filtering or generation of statistical flow measurements needed by the detection part. It is further divided into a layer for packet monitoring and sampling and a layer for statistical measurements. The detection part is divided into two detection engines, one providing statistical anomaly detection and the other applying knowledge-based detection mecha-

nisms. The required packet data and statistical measures are provided by the network monitoring part.

The main reason for separating the network monitoring part and the detection part is to allow for a multi-hierarchy monitoring environment for capturing packets and flow statistics. The metering NSLP protocol [15] can be employed for the configuration of the monitoring environment. This allows for deploying one detection system that analyzes data monitored at different points of the network. Furthermore, a detection system can become itself a source of information to other detection systems by exporting monitoring data.

In the following subsections, the network monitoring part and the detection part of the detection system are described in more detail. This and additional information on CATS can be found in [11].

*Packet monitoring and sampling layer*

The architecture of our detection system allows two ways to capture packet data from the network: by using a directly connected NIC, and by employing PSAMP exporters, which send the collected information in a standardized way. The packet monitoring and sampling layer is responsible for capturing of packet data received via NICs or PSAMP. Moreover this layer may preprocess the packet data. Filters or sampling algorithms may be applied to reduce the amount of packets being further processed. Within the detection system, the collected packet data is used for two purposes. First, it can be directly passed on to the detection part in order to look for known attack signatures. Secondly, it can be forwarded to the statistical measurement layer that generates flow statistics from the packet data. Additionally, the detection system can export packet data to other detection systems using PSAMP.

*Statistical measurement layer*

The statistical measurement layer generates statistical flow measures based on the packet data received by the packet monitoring and sampling layer, and the flow statistics received via IPFIX. Examples for statistical measures are the number of bytes and packets per flow or per aggregate, the number of connections per time, and the number of similar connections. The resulting statistical measures build the basis for further anomaly detections. For instance, an unusually high connection rate may indicate a distributed denial of service attack where typically each connection consists of only a single packet.

The statistical measurement layer does not only provide the data for the local detection mechanisms. It may also export the generated flow statistics via IPFIX. Using the terms of IPFIX, this corresponds to the functionality of an exporter or concentrator.

*Attack detection*

In the detection system, we integrate two separate, independently working detection engines - an anomaly detection engine and a knowledge-based detection engine - in order to achieve high detection rates. The detection of an attack results in the generation of an event that is combined with additional information for characterizing the attack. This information can be exchanged with other detection systems in order to improve the detection performance. On the other hand, it can be used to trigger appropriate countermeasures.

The anomaly detection works on statistical data received from the lower statistical measurement layer. This detection process is looking for unusual behavior without any precognition. It compares long-time behavior to short-time behavior and maintains different profiles, e.g. per destination, aggregate, and others. Potential techniques are statistical tests, neural networks, and Bayes networks. The architecture of our autonomous detection system allows to integrate a variety of other detection algorithms. The knowledge-based approach represents the second main pillar of our detection engine. This engine searches the packet stream for known signatures and misbehaviors. Open-source tools such as Snort [2, 24] and Bro [22], which are widely used in the Internet community, build the basis for this part of the detection.

## 3 Biological Background

In an interdisciplinary team we are identifying appropriate mechanisms in cell biology and to adapt them to networking technology with the focus on self-organization based on adaptive feedback loops. A structural comparison of organisms and computer networks depicts that both show high similarities. Also, the communications between the systems, the signal transduction pathways, follow the same requirements [9]. Here, a specific regulation mechanism is discussed.

Some organs such as the kidney do play a central role on physiological functions and dysfunctions of the organism. For example a descent of arterial blood pressure below a critical value which will have many negative consequences for the whole body is monitored by a small population of cells in the filtration unit of the kidney [25]. As an answer to this information, these cells produce a protein (renin) which has the function to initiate a cascade of conversions and activations, respectively, of another constitutive but quiescent protein (angiotensinogen) produced by the liver and distributed in several organs. The conversion of this protein to a shorter one (now called angiotensin I) is the first step to form the right answer for solving the initial problem. Further proteins are necessary for the formation of this final answer. The protein ACE (angiotensin converting enzyme) further modulates this protein, angiotensin I by cleaving it into the short and potent protein angiotensin II. This protein represents the final answer which now has many effects on different cells in

different organs in order to increase the blood pressure to normal level. On the one hand, angiotensin II stimulates the production of further protein signals in the adrenal gland, which is e.g. a hormone called aldosterone. This protein in turn stimulates the retention of Na+ ions in the kidney which finally has consequences for the blood volume regulation. On the other hand, angiotensin II stimulates the contraction of smooth muscle cells surrounding blood vessels within the kidney. Finally, angiotensin II also activates the production of the hormone vasopressin in the adenohypophysis in the brain which finally plays a role in the blood volume regulation. All these effects enhance the blood pressure in the whole body. The complete procedure is depicted in figure 2.

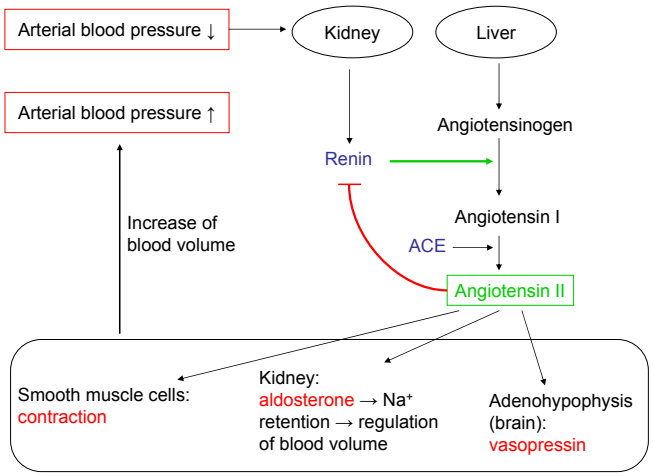


**Fig. 2.** Overview of the regulation of blood pressure (signaling cascades including a molecular negative feedback mechanism)

Looking at one of the target cells of angiotensin II in the kidney or smooth muscle cells, the protein binds to certain receptors on the cell surface. This binding induces an intracellular signal transduction cascade that finally results in the aforementioned actions to increase the blood pressure. A molecular negative feedback mechanism finishes the whole cellular reaction. If all receptor are bound by angiotensin II, the reaction is blocked which in turn also blocks the primary conversion of angiotensinogen to angiotensin II in the way that the initial renin secretion is blocked. Therefore, this mechanism describes a very effective remote and local control of the blood pressure which plays a central role in the body.

In summary, renin is a promoter for the development of angiotensin II, which in turn works as an inhibitor for the production of renin. A smooth self-regulation is the result of this feedback loop [17]. We will describe the

application of this methodology for the envisioned network security scenario in the next section.

## 4 Adaptive Control Scheme

Our developed adaptive control scheme depends on the current network behavior, i.e. on the observed traffic as well as on the current state of the individual subsystems. The primary intention of this approach is to prevent overload situations. In this section, we outline the used models and the application of previously depicted biological promoter / inhibitor principles. The developed model is the basis for the conducted performance measures.

### 4.1 Modeling the Security Solution

The overall goal is to adapt the rate of packets sent to the attack detection system. The following three methods can be used to regulate the rate of monitored data: *Compression/encoding* – Monitored packet data can be encoded in a way reducing the number of transmitted bytes to the minimum. This is done using IPFIX and associated aggregation mechanisms. *Statistical sampling* – In many cases statistically chosen packets can be successfully used to determine possible attacks or to identify legitimate traffic. The PSAMP framework specifies a number of sampling algorithms that fulfill the requirements of anomaly detection systems. *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 are prevented from reaching the systems under attack and, secondly, these packets also no longer reach the monitoring system. Therefore, the data rate being sent by the monitoring systems to the attack detection systems is reduced. Additionally, whitelists are used at the monitoring probes to reduce the amount of data transmitted to the analyzing systems.

Adaptive flow aggregation has been addressed by Hu et al. [16], proposing to adapt the aggregation level dynamically according to the available resources of the monitoring probe. Although this approach avoids flow loss in DDoS attack situations as described above, it does not take into account that arbitrarily defined flow aggregates usually do not meet the requirements of the analyzer.

In figure 3, a model is shown that represents the considered architecture. On the lower half, the packet-oriented monitoring part is shown. The observation domain reflects the network behavior. All data packets regardless of their content are received by the overall system. In a first step, a firewall component is used to filter all packets that belong to previously detected attack flows. A blacklist is involved here that can be configured dynamically by an IDS system. In a subsequent step, the monitoring probe is used to gather packet

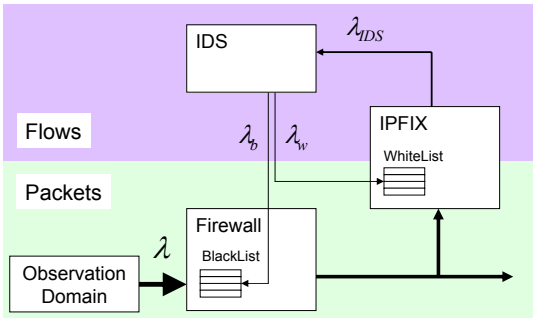


**Fig. 3.** Basic model for adaptive re-configuration including the main components and the control and data flows

information, invoke flow accounting and/or packet sampling algorithms, and to transmit the monitored data sets to the attack detection system. We introduce a whitelist, also configured by the IDS, representing all legitimate flows. Therefore, appropriate filters can be used at the monitoring probe to reduce the amount of data packets to be processed. Finally, an IDS system is used to process the monitoring data. All well-detected flows, either attack or legitimate, are communicated to the firewall or monitoring systems, respectively. The depicted parameters as used in figure 3 represent possible measures to adapt the system behavior. The total arrival rate $\lambda$ can be decomposed into attack traffic $\lambda_b$, legitimate traffic $\lambda_w$, and unknown data flows $\lambda - \lambda_b - \lambda_w$. $\lambda_{IDS}$ is a system parameter describing the maximum data rate that can be processed by the attack detection system. Finally, $\lambda_{bi}$ is the arrival rate of 'black' packets belonging to the i-th flow, i.e. representing the possibility to optimize the system behavior depending on particular attack flows.

### 4.2 Biologically inspired Promoters and Inhibitors

As previously mentioned, bio-inspired methodologies can be used to create appropriate feedback loops for adapting the 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 3. The intrusion detection reports detected attacks to the firewall that in turn blocks 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 pack-

ets at any time. Also, previously attacking machines might become 'corrected' and should not be starved by our firewalls.

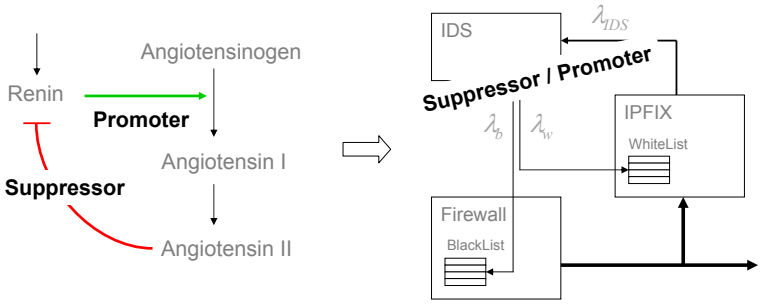


**Fig. 4.** Adapting the biological model to network monitoring; promoter and suppressor mechanisms are needed

In contrast to other system that dynamically configure firewall rules based on attack detection results, we introduce two other basic methods:

1. direct configuration of filters at the monitoring probes corresponding to legitimate data flows and
2. associated timeout values to each blacklist and whitelist entry corresponding to the current overall system load.

The adaptation of biological promoters and inhibitors is depicted in figure 4. The self-regulating process that amplifies the production of angiotensin II by the production of renin is reflected by the measurement of $\lambda_b$ and $\lambda_w$. If there are resources available at the attack detection system, the amplification effect is initiated by decreasing the associated timeouts for entries in the backlists and whitelists. Similarly, the suppressing reaction, i.e. angiotensin II inhibits the production of renin, is modeled by modifying the timeouts in the opposite direction in case of overload situations. Thus, 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.

### 4.3 Mathematical Modeling

Finally, the adaptation is done using the following formulas for calculating appropriate timeouts. $TO_{black}$ as estimated by (1) corresponds to the firewall system. Obviously, this timeout depends on the measured rate of attack and

legitimate flows as well as on the maximum load of the IDS system. Particular interest is expressed in specific attack flows, or more precisely in the long-term behavior of such flows. Therefore, the load of attack flows is measured for each flow separately by $\lambda_{bi}$. We used this measure to punish flows that reappear multiple times in the blacklist. The constants $C_1$ and $C_2$ explicitly define the system behavior. Two different values are required because of the different scaling of $t_1$ and $t_2 ... t_4$.

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

On the other hand, $TO_{white}$ as computed by (2) corresponds to the monitoring probe. Here, we only focus on the behavior of the firewall system and the capacity of the IDS represented by $\lambda_b$ and $\lambda_{IDS}$, respectively. Again, $C_3$ is used to define the system behavior.

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

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.

- $t_1$ – 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.

$$t_1 = \begin{cases} \frac{\lambda_{bi}}{\lambda} & \text{if } \lambda \neq 0 \\ 0 & \text{if } \lambda = 0 \end{cases} \tag{3}$$

- $t_2$ – 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.

$$t_2 = \begin{cases} \frac{\lambda_b}{\lambda} & \text{if } \lambda \neq 0 \\ 0 & \text{if } \lambda = 0 \end{cases} \tag{4}$$

- $t_3$ – 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.

$$t_3 = \begin{cases} \frac{\lambda}{\lambda_w} & \text{if } \lambda_w \neq 0 \\ 0 & \text{if } \lambda_w = 0 \end{cases} \tag{5}$$

- $t_4$ – This term is a measure for the overload of the attack detection system.

$$t_4 = \begin{cases} \frac{\lambda}{\lambda_{IDS}} & \text{if } \lambda_{IDS} \neq 0 \\ 0 & \text{if } \lambda_{IDS} = 0 \end{cases} \tag{6}$$

- $t_5$ – Similar to $t_4$ but used at the monitor instead of the firewall system.

$$t_5 = \begin{cases} \frac{\lambda}{\lambda_{IDS}} & \text{if } \lambda_{IDS} \neq 0 \\ 0 & \text{if } \lambda_{IDS} = 0 \end{cases} \tag{7}$$

- $t_6$ – Similar to $t_3$ but defining the potential risk of arriving packets.

$$t_6 = \begin{cases} \frac{\lambda}{\lambda_b} & \text{if } \lambda_b \neq 0 \\ 0 & \text{if } \lambda_b = 0 \end{cases} \tag{8}$$

## 5 Simulation Model and Evaluation

### 5.1 Simulation model

In order to evaluate the potentials of the described feedback-based adaptation, we implemented a simulation model using the JAVA-based discrete simulation tool AnyLogic. The simulation model is depicted in figure 5. Several meters have been included to measure the performance of the overall system. In all setups, we executed a set of simulations showing the reduction of packet data that is to be received and processed by the attack detection systems. The simulation model has to be interpreted as follows. The observation domain 'creates' packet data (see below) that is inspected by the `observationDomainLinkMeter`. Afterwards, a firewall element is used to filter packets according to the current blacklist configuration. The packets arriving at the monitor are measured by the `firewallLinkMeter`. Depending on the simulation setup, the monitor is providing IPFIX or PSAMP data (after applying the whitelist) that is measured by the `exportLinkMeter`. Finally, the IDS selects packets to belong to attack or legitimate traffic according to a probabilistic scheme. The results are used to adapt the blacklist and whitelist entries. All measured information is transmitted to the `meterLogger` object for subsequent performance analysis.

For reasonable comparability between the simulation environment and real communication behavior, 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. As am example, figure 6 shows the throughput as observed by the monitor at the Internet gateway. The utilization is quite constant (58.6kpps ± 1.8kpps) as shown in figure 6 left. Due to few attacks, the firewall removes only few flows (around 0.5%). Therefore, the packet rate arriving at the monitor is quite the same but shaped (58.2kpps ± 1.8kpps, figure 6 right).

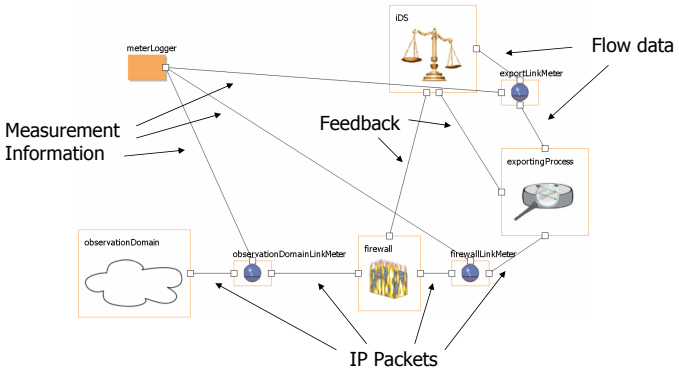


**Fig. 5.** Simulation model including message types on the communication channels: a) IP packets, b) flow data, c) feedback, d) measurement information

**Fig. 6.** Measurement at the Internet gateway: input ratio (left) and input ratio at the monitor (right)

### 5.2 Results using flow monitoring

In a first set of simulations we evaluated the capabilities of the proposed adaptive reconfiguration scheme for flow-based monitoring. Besides the main objective to assess the overall system behavior, 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 other simulation parameters are $\lambda_{IDS} = 0.6$, E[white]= 0.1, and E[black]= 0.01. As already mentioned, we monitored the traffic at the border gateway of our university. The utilization was quite constant (58.6kpps $\pm$ 1.8kpps). Due to few attacks, the firewall removes only few flows (around 0.5%). Therefore, the packet rate arriving at the monitor is similar but shaped (58.2kpps $\pm$ 1.8kpps).

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 7. In steady-state, around 6000 blacklist entries exist with an average timeout of 260s.

Finally, the numbers of whitelist entries and the corresponding timeouts have to be examined. As shown in figure 8, in steady-state, the number of whitelist entries is around 60000 and the average timeout is 320s. The number

**Fig. 7.** Number of blacklist entries (left) and timeouts $TO_{black}$ (right)

corresponds to the detection quality of legitimate traffic which is about ten times higher then for attacks. The timeout oscillates around the requested value.

**Fig. 8.** Number of whitelist entries (left) and timeouts $TO_{white}$ (right)

In figure 9, the measured parameters $\lambda_b$ (left) and $\lambda_w$ (right) are depicted. These figures allow a more convenient analysis of the simulation results.

**Fig. 9.** Measured parameters $\lambda_b$ (left) and $\lambda_w$ (right)

### 5.3 Results using packet sampling

For the packet sampling based measurements, we used the packet trace taken in front of out workgroup server. The monitor is executing a sampling algorithm that selects 50% of the packets (count-based). Obviously, the output packet rate is about one half of the input packet rate and the output byte rate represents the reduction due to keeping only parts of the packet (IP header including transport protocol information).

In the following, we used the implemented blacklist and the whitelist representing the detected attack and legitimate flows. The most important parameters to analyze are: *Timeout* – A timeout value associated to each entry in the blacklist and whitelist. This defined how long an entry as determined by the attack detection system will be valid in the firewall system and the monitoring probe, respectively. *Detection ratio* – We assume a constant detection ratio resulting in new blacklist / whitelist entries. This is a presumption that does not correctly correspond with the behavior in a real network. Nevertheless, it reflects the behavior of the global system pretty well due to the proper configuration of the blacklist / whitelist.

In the following simulation results, we always display the output packet rate as issued by the monitoring probe and the input rate as a reference, because this reflects the corresponding amount of data that is to be processed by the detection system. In the first simulations, we examined the effect of the timeout value associated with the single entries in the blacklist and whitelist. For analyzing this behavior, we statically configured a detection ratio of 10%

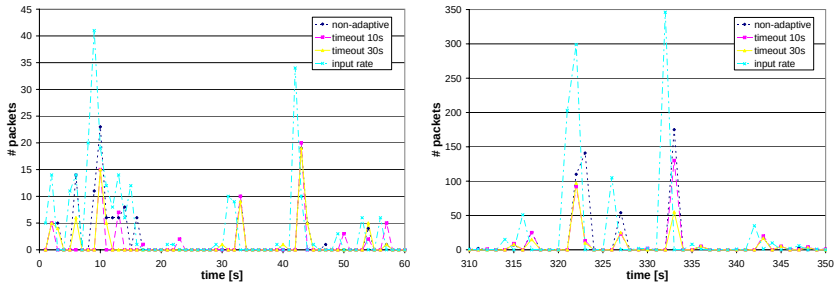


**Fig. 10.** Output rate for variable timeout and fixed detection ratio (10% whitelist, 1% blacklist)

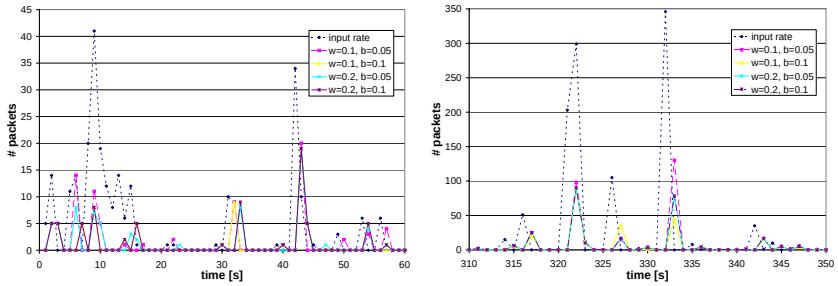


**Fig. 11.** Output rate for variable detection ratio and fixed timeout (10s)

for new whitelist entries and 1% for new blacklist entries. These values seem to be adequate to reflect the behavior in real networks because about ten times as mush flows can be detected as legitimate traffic that as attack traffic and most of the network packets are not directly categorizable.

In figure 10, the simulation results are shown. Obviously, the adaptive configuration has a large impact on the amount of output packets. Additionally, it can be seen that a large timeout, which results also in large blacklist / whitelist tables and therefore, exhaustive search operations, does not lead to a massive reduction of the output packet rate. The reason for this behavior is the relatively short time a flow is lasting in the network. In figure 11, a second simulation run is shown. This time, the timeout was fixed at 10 seconds and the detection ration was modified. Interestingly, especially the whitelist has not that large impact on the amount of packets sent to the detection system as expected by the percentage of "white" packets (up to 20%). Additionally, it can be seen that the amount of data presented by the monitoring probe to the attack detection system can be adapted using an information exchange between all involved systems. An optimal adaptation to the behavior of the network seems to be very important and will lead to an optimized global

system. This optimization step can be executed independently by each participating entity in the network by two meanings: local parameterization and communication / interoperation of neighboring entities. Therefore, we have shown that it is possible to self-organize the complete monitoring and analyzing environment for network security enhancement focusing on the monitoring part.

## 6 Conclusion

In this paper, we studied the adaptation of biologically inspired promoter / inhibitor schemes for adaptive parameter control in network security environments. Using an amplifying positive feedback loop and a suppressing negative feedback loop, we achieved a self-organizing autonomic 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.

## References

1. H.-W. Braun, k. Claffy, and G. C. Polyzos, "A framework for flow-based accounting on the Internet," in *IEEE Singapore International Conference on Networks (SICON'93)*, Singapore, September 1993, pp. 847–851.
2. B. Caswell and J. Hewlett, "Snort Users Manual," The Snort Project, Manual, May 2004.
3. R. K. C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," *IEEE Communications Magazine*, vol. 10, pp. 42–51, October 2002.
4. B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, October 2004.
5. ——, "IPFIX Protocol Specification," Internet-Draft (work in progress), draft-ietf-ipfix-protocol-22.txt, June 2006.
6. F. Dressler, "Adaptive network monitoring for self-organizing network security mechanisms," in *IFIP International Conference on Telecommunication Systems, Modeling and Analysis 2005 (ICTSM2005)*, Dallas, TX, USA, November 2005, pp. 67–75.
7. ——, "Efficient and Scalable Communication in Autonomous Networking using Bio-inspired Mechanisms - An Overview," *Informatica - An International Journal of Computing and Informatics*, vol. 29, no. 2, pp. 183–188, July 2005.
8. F. Dressler and I. Dietrich, "Simulative Analysis of Adaptive Network Monitoring Methodologies for Attack Detection," in *IEEE EUROCON 2005 - The International Conference on "Computer as a Tool"*, Belgrade, Serbia and Montenegro, November 2005, pp. 624–627.

9. F. Dressler and B. Krüger, "Cell biology as a key to computer networking," in *German Conference on Bioinformatics 2004 (GCB'04), Poster Session*, Bielefeld, Germany, October 2004.
10. F. Dressler and G. Münz, "Flexible Flow Aggregation for Adaptive Network Monitoring," in *31st IEEE Conference on Local Computer Networks (LCN): 1st IEEE LCN Workshop on Network Measurements (WNM 2006)*, Tampa, Florida, November 2006, pp. 702–709.
11. F. Dressler, G. Münz, and G. Carle, "Attack Detection using Cooperating Autonomous Detection Systems (CATS)," in *1st IFIP International Workshop on Autonomic Communication (WAC 2004), Poster Session*, Berlin, Germany, October 2004.
12. F. Dressler, C. Sommer, and G. Münz, "IPFIX Aggregation," Internet-Draft (work in progress), draft-dressler-ipfix-aggregation-03.txt, June 2006.
13. N. Duffield and M. Grossglauser, "Trajectory Sampling for Direct Traffic Observation," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 3, pp. 280–292, June 2001.
14. N. Duffield, "A Framework for Packet Selection and Reporting," Internet-Draft (work in progress), draft-ietf-psamp-framework-10.txt, January 2005.
15. A. Fessi, G. Carle, F. Dressler, J. Quittek, C. Kappler, and H. Tschofenig, "NSLP for Metering Configuration Signaling," Internet-Draft (work in progress), draft-dressler-nsis-metering-nslp-04.txt, June 2006.
16. Y. Hu, D.-M. Chiu, and J. C. Lui, "Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks," in *IEEE/IFIP Network Operations and Management Symposium (IEEE/IFIP NOMS 2006)*, Vancouver, Canada, April 2006, pp. 424–435.
17. C. A. Janeway, M. Walport, and P. Travers, *Immunobiology: The Immune System in Health and Disease*, 5th ed. Garland Publishing, 2001.
18. B. Krüger and F. Dressler, "Molecular Processes as a Basis for Autonomous Networking," *IPSI Transactions on Advances Research: Issues in Computer Science and Engineering*, vol. 1, no. 1, pp. 43–50, January 2005.
19. T.-H. Lee, W.-K. Wu, and T.-Y. W. Huang, "Scalable Packet Digesting Schemes for IP Traceback," in *IEEE International Conference on Communications*, Paris, France, June 2004.
20. J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, April 2004.
21. M. Molina, "A scalable and efficient methodology for flow monitoring in the Internet," in *18th International Teletraffic Congress (ITC18)*, ser. Providing Quality of Service in Heterogeneous Environments, J. Charzinski, R. Lehnert, and P. Tran-Gia, Eds., vol. 5a. Berlin, Germany: Elsevier, August 2003, pp. 271–280.
22. V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, December 1999.
23. J. Quittek, S. Bryant, B. Claise, and J. Meyer, "Information Model for IP Flow Information Export," Internet-Draft (work in progress), draft-ietf-ipfix-info-12.txt, June 2006.
24. M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *13th USENIX Conference on System Administration*. USENIX Association, 1999, pp. 229–238.

25. R. F. Schmidt, F. Lang, and G. Thews, *Physiologie des Menschen*, 29th ed. Springer Verlag, 2005.
26. T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection," Internet-Draft (work in progress), draft-ietf-psamp-sample-tech-07.txt, July 2005.