Horizon Project

ANR call for proposals number ANR-08-VERS-010
FINEP settlement number 1655/08

Horizon - A New Horizon for Internet

WP3 - TASK 3.3: Situated View: Identification of Update Needs
(Annex I)

Institutions

Brazil	**France**
GTA-COPPE/UFRJ	LIP6 Université Pierre et Marie Curie
PUC-Rio	Telecom SudParis
UNICAMP	Devoteam
Netcenter Informática LTDA.	Ginkgo Networks
	VirtuOR

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Network virtualization technique allows the execution of multiple virtual networks over the same physical hardware. This functionality is achieved through the use of a control and management entity, which is responsible for multiplexing the hardware access and providing logical slices of resources to the virtualized systems. The main primitives of a virtualized networking system are the creation and destruction of virtual networks, the migration of virtual nodes and the mapping of virtual networks on the physical substrate. These primitives suit any network virtualization platform, such as OpenFlow or Xen, which are used in this project.

In the standard version of Xen and OpenFlow, all of the above mentioned primitives are manually run, which implies scalability and management problems. Hence, we developed in this work package a piloting plane that is able to autonomously execute these primitives. Nevertheless, it is important to know when to use those primitives and also when virtual networks are no longer working as desired and, consequently, require modifications on their properties. We developed a knowledge plane that works together with the piloting plane. This knowledge plane monitors virtual routers, obtain their usage profile, and proactively detects, using prediction mechanisms, possible need for updates in the virtual network configuration. The knowledge plane brings the idea of a plane that takes into account the whole network and retrieves knowledge from it. In this plane, information concerning each virtual network element is stored, allowing management decisions and proactive network maintenance. It is important to notice that, due to scalability issues, this knowledge plane must be distributed in many nodes. Therefore, each node keeps a partial view of the knowledge plane, limited to their neighborhood and the surroundings. This partial view is called the situated view of a node. The main challenges of the knowledge plane are the time scheduling to take decisions and the information update about each network element.

The goal of this task is to provide mechanisms to solve these problems.

Once defined the information that must be stored on the knowledge plane, we can identify the frequency of updates for each "piece of information" and on which nodes this information must be stored, based on performance and QoS metrics. Hence, we can define a set of situated views of the context concerning the piloting plane. Thus, instead of having information about the whole network, we use only local information to decide which action should be executed in the network, improving scalability.

We developed for this task a set of mechanisms that distributedly identify and stores changes in the network state and predicts the evolution of all the variables involved in the networking process, so that virtual network anomalies can be detected and corrected. The proposed mechanisms can be used by both Xen and OpenFlow platforms. For sake of simplicity, we show the utilization of our proposals only in the Xen platform.

The first proposed approach is based on the development of a dynamic allocation system that analyzes virtual router profiles and provides a fair share of resources based on QoS metrics and SLA agreements. The second proposed approach focuses on monitoring and prediction techniques that monitor the environment, provide to the knowledge plane proper and updated information and also detect router misbehavior. These two approaches together constitute a framework for detecting the need for updates and also when SLAs are violated, which may also trigger update alarms.

This report is organized as follows. Chapter 2 details the mechanism to extract and store virtual network profiles. Also, we describe a QoS controller implemented according to the obtained profiles and the defined SLAs that is based on fuzzy logic. Chapter 3 explains in details the monitoring suite and the predictors used to detect the need for changes in many parameters obtained from machines. Chapter 4 concludes the report.

Chapter 2

Dynamic SLA Controller

This chapter defines one of the approaches to detect the need of updates on a virtual router environment. This approach is based on the development of a system that dynamically controls Service Level Agreements (SLA) and provides Quality of Service (QoS) guarantees in a virtualized network environment.

2.1 Related Work

The proposed dynamic allocation mechanisms found in the literature are mainly focused on server consolidation. Sandpiper [1] is a system that monitors virtual machines in a data center and migrates virtual machines to different physical servers in order to achieve a virtual machine distribution that maximizes the performance and reduces the misuse of resources. It also considers the profile generated by each virtual router through the use of time series to avoid misbehaviors such as denial of service attacks. To determine the physical resource utilization of each physical server, the authors propose a volume metric Vol, expressed as

$$Vol = \frac{1}{1 - cpu} * \frac{1}{1 - mem} * \frac{1}{1 - net}, \tag{2.1}$$

where *cpu* is the processor utilization percentage, *mem* means the memory utilization and *net* stands for network utilization. Meng *et al.* propose algorithms that provide the best distribution of virtualized servers in a grid of physical servers [2]. Virtual servers are instantiated on physical servers in order to reduce the distance between servers that are exchanging data with each other, thus optimizing network scalability and providing better utilization of bandwidth in communication links.

Menascé *et al.* apply autonomic computing techniques to control the processor sharing among virtual machines [3]. The authors propose a dynamic allocation algorithm that is validated through simulations.

Xu *et al.* propose control mechanisms based on fuzzy logic to optimize the resource allocation in data centers and also execute performance tests in virtualized web servers with different workloads [4]. A learning system feeds the fuzzy controller in order to understand the web server behavior under different loads.

One important tool that helps the dynamic resource allocation is the migration of virtual machines, which is used in Sandpiper [1].The migration primitive enables migration of virtual machines from a physical server to another, allowing preventive maintenance and energy saving obtained through the re-organization of machines and shutdown of under-utilized servers. Nevertheless, the migration procedure represents a challenge for virtualized network environments due to the packet losses during the period of time where the machine is suspended. Wang*et al.* propose a live migration mechanism without packet loss [5] and Pisa *et al.* implement this proposal in the Xen architecture [6].

The dynamic allocation and control of resources is a challenge in the Xen platform because the I/O virtualization technology is still naive, without isolation, allowing malicious routers to impact in the performance of others. Therefore, XNetMon [7] proposes a secure control system for routing traffic based on data and control plane separation approach to manage the use of I/O resources by the virtual routers.

Keller *et al.* analyze QoS requirements in virtualized environments and propose authoring model to account and guarantee SLAs in virtual routers [8]. Authors explain the need to use authoring mechanisms and propose two main implementation models. The first is based on the monitoring of network parameters while the second is based on the use of trusted platforms and cryptographic keys to make systems tamper proof.

This report presents a dynamic controller based on Service Level Agreements (SLA) for virtual networks. The control is based on the generation and further analysis of router profiles and on the detection and real time punishment of SLA violations. The proposed system monitors load values associated with each virtual network and generates real estimative of use profiles. Those profiles are used to ensure the better router organization to reduce the overload probability. The load function is based on Sandpiper but also includes other important parameters, such as system robustness, operation temperature, and others that can be added if desired. The proposed fuzzy control system aims at providing easy configuration of the weight of each parameter. The fuzzy logic is used to map administrator strategies and

allow them to punish routers that violate SLAs. Besides, the network administrator can easily insert new rules and action strategies. The proposed system supports data/control plane separation and, as a consequence, is fully compatible with XNetMon.

Results obtained from the prototype show the behavior of the control system, the generated profiles and the strategy mechanisms. The controller consumes only a few CPU cycles to execute, allowing the monitoring of many virtual routers in parallel.

2.2 The Proposed Control System

The proposed control system monitors and guarantees Service Level Agreements in virtualized networks. The key idea relies on the generation and analysis of use profiles of each virtual network and on the detection and real-time punishment of violations. In the violation detection, there are parameters that consider the level of punishment applied to each virtual router, according to the current system state and the severity of the violation. The global state of routers and control domain is also characterized by using a fuzzy controller, that takes into account the processor, memory, network and robustness (existence of redundancy and fail-proof mechanisms) when calculating the system load. According to the output of the controller, it is possible to dynamically modify the punishment strategies and the system tolerance to violation actions.

The proposed system follows a distributed management model composed of controller agents. Each agent is associated with a set of physical routers where virtual routers execute, as seen in Figure 2.1. Each controller agent can be associated to a given number of domains and the network manager must decide on how these associations are accomplished. Each physical router has a control domain where a control and monitoring daemon monitors the allocation of physical resources, verifies in real time the conformity of SLAs, and generates the use profiles of each virtual router. Five modules compose the controller agents. The Strategy and Policy Module (SPM) holds management strategies that can be applied over physical routers under its control and updates the current strategy to be applied on each control and monitoring daemon. The Service Level Module (SLM) keeps a database that associates the SLAs with each virtual router. The Knowledge Base Module (KBM) stores the history, use profiles and the description of violations that have occurred. This module can be used to estimate future network migrations, detect the need for updates and re-negotiate contracts, adapting them to the real profile of each virtual router. The Actuator Module executes within

the control and monitoring daemons and retrieves the profiles and statistics of each virtual router. The controllers also have a Communication Module (Comm), which allows the exchange of information among other controllers through secure channels. If necessary, the controllers may use those channels to negotiate actuation domains changes and negotiate virtual elements migration.

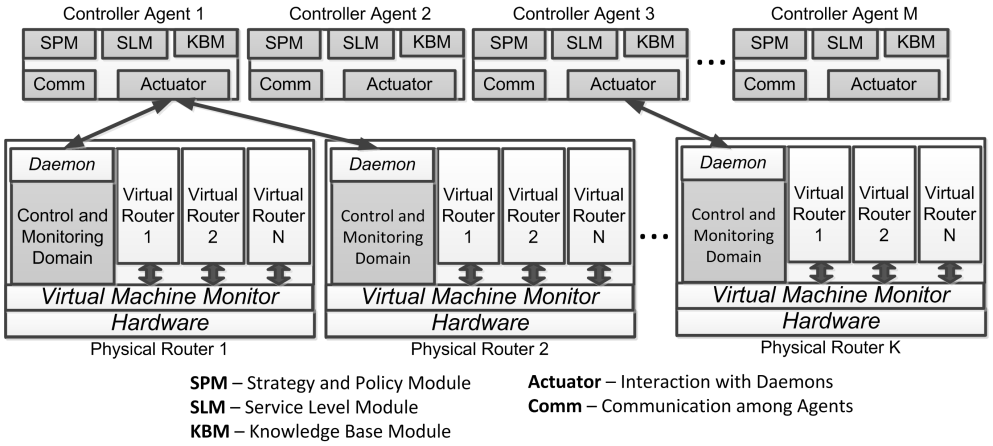
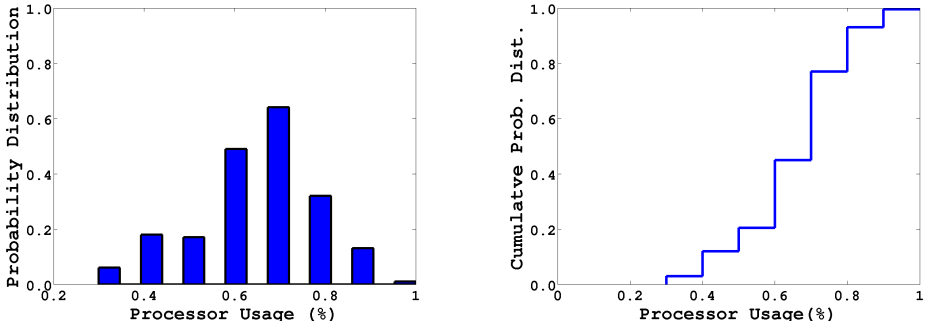


Figure 2.1: The control system architecture. Actuator module of the distributed controller agents interacts with monitoring and control daemons running on physical routers.

The described system has three main mechanisms. The first one is the profile generation mechanism, which provides utilization statistics and SLA violation detection. This information is stored in the Knowledge Base Module (KBM) in order to re-negotiate possible misconfigurations in the contracted SLAs. The second mechanism is the system load estimator, which gives an output that combines multiple resource status in an estimative within the $[0, 1]$ interval. The third mechanism is the adaptive punishment mechanism. Based on the system load and use profiles, the mechanism uses a fuzzy controller that outputs a level of punishment, proportional to the system overall state. For instance, if the system presents a low load, a medium violation (for example, overcome in 20% the SLA) generates a small punishment (reduce in 2% the resource utilization of the machine which violates the SLA). On the other hand, if the system is overloaded, without available resources, even a small violation can be punished severely.

2.2.1 Generating Router Profiles

The use profile of each virtual router represents the resource consumption pattern of each virtual router. The profiles can be used to detect rule violation, to estimate the use of resources, and to predict future needs. Profiles are generated through the capture of memory, processor and network utilization over time, to store the recent past and the long past of those variables. There are two sliding windows with distinct sizes to store those two time series. The generation of profiles based on probabilistic density functions is used in Sandpiper [1]. Recent past is used to check SLAs whilst the long past is used to predict future behaviors and estimate possible future demands. Behaviors are analyzed by probability functions.



(a) Probability Density Function (PDF). (b) Cumulative Distribution Function (CDF).

Figure 2.2: processor utilization profiles of a virtual router running RIPv2.

A Probability Density Function (PDF) of the processor utilization by a given virtual router executing RIPv2 is shown in Figure 2.2(a). It can be seen that the exchange of control and data messages in this specific router has generated a processor usage of approximately 0.7% of CPU in 70% of the measurements from the long past window , which consists in the last two hundred measurements in this scenario. Due to this PDF, we can conclude that it is acceptable to aggregate a group of virtual routers with similar resource patterns and make them share the same physical core, without losing performance. The prototype also provides Cumulative Distribution Functions (CDF) to verify SLAs. Through this perspective, we can think of flexible SLAs. For instance, we can define that a virtual router can use up to 0.7% of a given resource for a maximum of 80% of its execution time. Through the CDF of the short past window, it is possible to determine that the router in Figure 2.2(b) would fulfill the given SLA. It is important to mention that the generated distributions can be applied to any router from any machine. The

given example just demonstrates which kind of distributions can be obtained.

2.2.2 Strategy and Policy Module

The Strategy and Policy Module stores the current acting strategies and maps administrative decisions into actions and strategies. We use fuzzy controllers [9] due to its capability to deal with decision making problems that present uncertainties and qualitative parameters, e.g. the strategies of a network manager or administrator. In fuzzy logic, as mentioned before, a given element belongs to a given set according to its membership level inside the interval $[0, 1]$, where $\mu_A(x) : X \rightarrow [0, 1]$ defines a membership function. We adopt the Mamdani Imply Method, with Zadeh [10] AND and OR operators and the centroid method of defuzzification. The fuzzy controllers have small computational complexity and can parallelize inference procedures enhancing system performance and reducing the response time of the controller.

The Strategy and Policy Module supports different acting strategies. Each strategy is composed of a set of inference rules, a set of membership functions that map input parameters according to the network manager perception (high processor utilization, low memory load, etc.) and a set of membership functions that regulates the output. There are two strategy types: The system load strategies and the punishment strategies. These two approaches and their correlated strategy packets are described in Section 2.2.5. Those strategies formalize a computational behavior that reflects the will and the strategies of the network manager.

2.2.3 Estimating the System Load

The system load is a measurement that determines the load level of the managed resources. Multiple parameters such as processor utilization , memory utilization, network utilization, system overall temperature, and system robustness, which indicates the existence of redundancy mechanisms for disks and power supplies, can be analyzed to characterize the system state. We define the set of membership functions μ_{Proc},μ_{Mem}, μ_{Net}, μ_{Temp}, and μ_{Rob}, which associates each of the resources in fuzzified variables. The combination of the parameters generates an output defined as system load, limited in the interval $[0, 1]$, which defines the load of the system. This value is used as an input parameter of another controller, together with delta, which is the difference between the resources contracted and the resources used, to estimate the punishment grade of routers that violate SLAs. A block diagram that represents the system load controller can be seen in Figure 2.3.

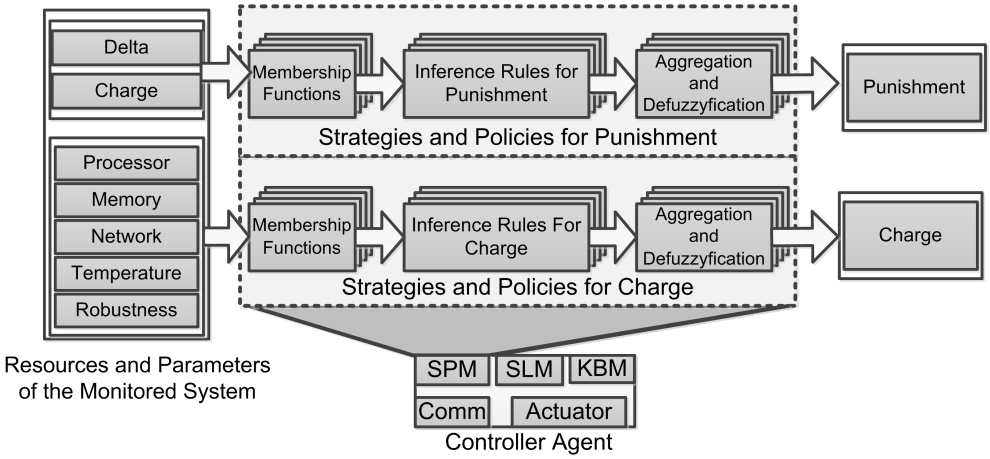
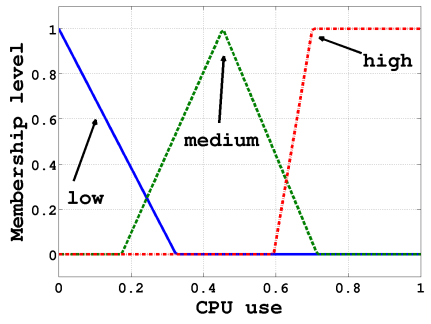
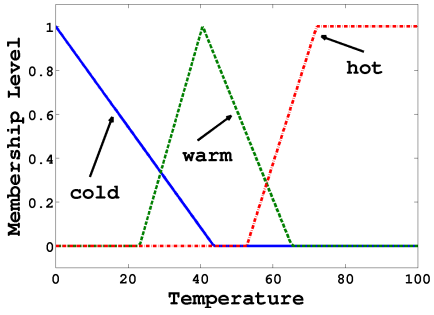


Figure 2.3: Strategy and Policy Module.

The resource use of each virtual router is aggregated to generate the controller input. An example of a possible configuration that evaluates the processor use and system temperature can be seen in Figure 2.4. It is important to remember that the presented curves can be modified according to each manager needs and qualitative thoughts . Low, medium, high, cold, warm, and hot are membership functions. In the given configuration, we have employed three membership functions to map each resource. We can see that the definition of membership functions represents the mapping of the network managers' qualitative decision making. Most of the rules can be defined as triangular or trapezoidal function. For example, if the processor use is high and the overall system temperature is high, then the system overload is also high.



(a) Membership functions for processor utilization.



(b) Membership functions for temperature.

Figure 2.4: Membership functions for processor and temperature.

2.2.4 Strategies Based on Inference Rules

The fuzzy controller strategies are based on the default fuzzy inference rules of the fuzzy system. Those rules follows the IF $\rightarrow$ THEN pattern which represents the current action strategy scheme. The set of rules that defines a strategy is named a strategy package. An example of strategy package that calculates the punishment level according to the difference between the contracted SLA and the current resource use, denoted by delta, and the system load can be seen in Table 2.1.

Strategy Packet
If **Delta** (low) and **Load** (low) Then **Punishment** (low)
If **Delta** (average) and **Load** (low) Then **Punishment** (low)
If **Delta** (high) and **Load** (low) Then **Punishment** (average)
If **Delta** (low) and **Load** (high) Then **Punishment** (average)
If **Delta** (average) and **Load** (high) Then **Punishment** (high)
If **Delta** (high) and **Load** (high) Then **Punishment** (high)

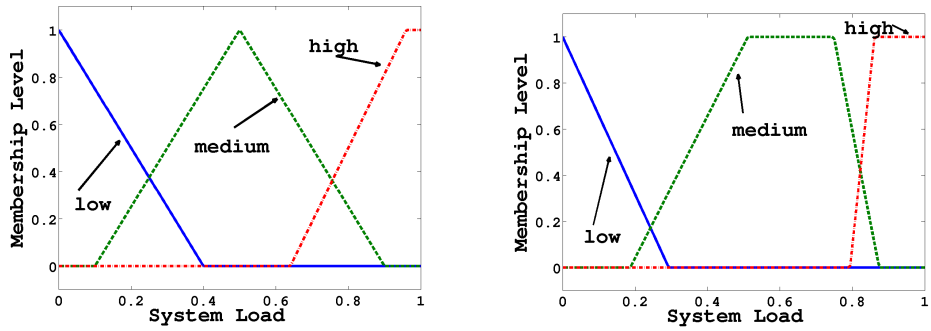
Table 2.1: Example of a piece of a strategy packet.

The presented strategy packet corresponds to a network manager policy that establishes that even huge SLA violations is not punished severely, when the system is lightly loaded, because the system has plenty of resources and at this moment, giving additional resources to the violating router do not disturb the others. On the other hand, when the system is overloaded, the network manager is more rigorous and even light violations are punished severely. These rules work together with the membership functions that can be also developed by the network manager. It is possible to easily insert new rules and strategies and the controller agent must export the strategy packets to the daemon that must use the defined strategy. We can also establish different strategies for each resource under control, enhancing the flexibility of the controller.

2.2.5 Load Policies

After the application of the inference rules, fuzzy values are generated to represent the level of pertinence of each inference rule and then those values are mapped into a single controller output, which is a value in the interval $[0, 1]$ which represents the current system load. In Figure 2.5, we can see two possible load policies, a conservative and an aggressive one. Depending on

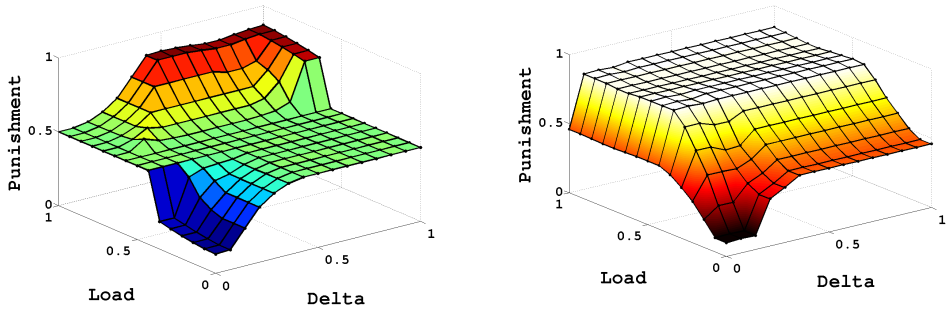
the network manager profile, we can dynamically change the current domain policies.



(a) Conservative policy. (b) Aggressive Policy.

Figure 2.5: System charge policies.

Given the configurations and membership functions, it is possible to verify the relationship among the punishment level, the system load and delta, i.e., the level of SLA violation. This relationship can be viewed in Figure 2.6. In those surfaces, we can see that a given policy defines how the punishment level variation. Besides, the combination of different inference rules and membership functions generate different surfaces. In the conservative policy, we can see that punishments are severe only when delta is high and the system is overloaded (Figure 2.6(a)) whilst in the aggressive policy, small positive variations in system load and in delta generate high levels of punishment (Figure 2.6(b)).



(a) Decision surface for a conservative policy. (b) Decision surface for an aggressive policy.

Figure 2.6: Decision surfaces to different management strategies.

2.2.6 Controlling the System Overload and SLAs

The developed system is capable of generating use profiles, evaluating if profiles correspond to established SLAs, generating estimative of the system load and punishing virtual routers that violate the proposed rules. In the implementation of the prototype, the daemon that executes within each domain performs the parameter gathering at each given sampling interval, which can be defined by the network manager. The parameters are used to generate temporal series that represents the variation in the use of each resource and the statistics and distributions that allow the verification of profiles and the compliance with SLAs. All this information is sent to the responsible controller agent. The daemon verifies if each virtual router profile corresponds to the negotiated SLAs. Moreover, it aggregates the resources used by each router to estimate the total load of the physical system. If a virtual router violates the contracts, the system generates a value that represents delta. The system then uses this delta value and the system load to decide which is the appropriate punishment level to be applied on the router. In the Xen architecture, we use the cap control parameter. The cap regulates the number of CPU cycles each virtual element can use. Thus, varying the cap value, we can control the use of processor resources of virtual routers. Another control tool that can be used is the Traffic Control (TC), which allows the queue control, permitting the management of the throughput of each virtual router, if they violate SLAs.

2.3 Results

The developed control system is efficient and flexible. We have already demonstrated system outputs, such as the tested strategy packets, the decision making surfaces generated by the system and the use profiles of multiple parameters from virtual routers and physical servers. We have developed a set of experiments that prove the efficiency and the low overhead induced by the proposed system. Tests were performed in a physical machine with a core i7 860 processor with 4 real cores and 8GB DDR3 RAM. The machine was configured with hypervisor Xen 4.0. The virtual routers were instantiated with 128MB RAM memory and access to one virtual core. The virtual routers and the control domain execute Debian Lenny Linux with 2.6.32-5-amd64 kernel.

The design of the control system minimizes the processing overhead of the monitoring and control daemon. To evaluate the processor overhead of the daemon, we have instantiated some virtual routers and measured the average

processor utilization of the control domain according to the number of virtual routers monitored. The results of Figure 2.7 show the processor utilization in the control domain according to the number of monitored virtual routers. In this configuration, measurements and decisions were gathered and evaluated at each second. The points in the graph represent the average processor utilization for each configuration, with a confidence interval of 95%. We can see that the relation between processor utilization and number of monitored routers is approximately linear and even in situations where the daemon manages eight routers at the same time, the overhead load is acceptable and reaches up to 40% of processor utilization of a single core.

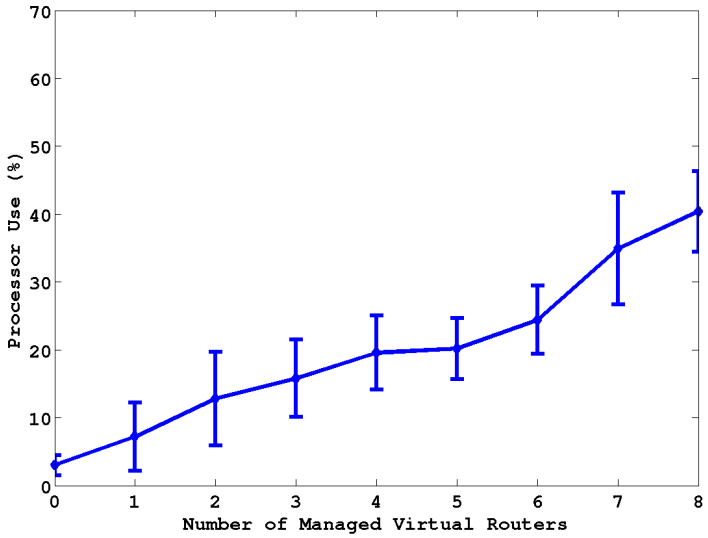


Figure 2.7: Processor utilization in control domain when varying the number of managed virtual routers.

The system presents a good performance considering that it is monitoring multiple variables from multiple routers at the same time and that the system is managing the SLAs of each router. We can estimate that a control domain with the same configuration as the one used in this experiment can manage up to 20 virtual routers at the same time, by dedicating only one single core to this task.

The second experiment evaluates the controller efficiency and the effects of the punishment mechanism. We selected one of the virtual routers. The SLA of this router defines that the router can use up to 85% of processor of a single core to execute packet forwarding. Next, we create a packet flow that is forwarded by the router. When the flow is forwarded, the router violates

the SLA and the control system regulates processor utilization through the cap mechanism. In the experiment, the punishment system is enabled when the router is already violating the SLA. It is important to remember that the sampling interval can be regulated by the network manager.

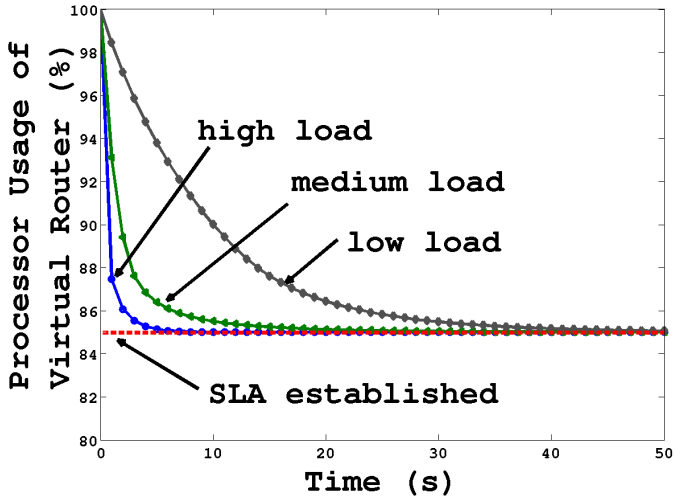


Figure 2.8: System stability under different system loads. Faster convergence to the contracted SLA due to more rigorous punishment when the system load is high.

In the proposed scenario, we defined three background environments. In the first one, there is one monitored router and a virtual router that does not use resources, so the system keeps a low load. In the second environment, there is the monitored router and five more virtual routers that are consuming a moderate amount of resources. In this scenario, the load was medium. The third case has one the monitored virtual router and seven extra virtual routers, both using almost all of the available resources. In this case the system load is characterized as high. The characterization of the scenario comprises the low, medium, and high load outputs. Those values were obtained considering a set of membership functions and inference rules that were on the system. In each configuration, the system generated a different load output, depending on the background environment. We selected three specific environments from all the environments tested, to demonstrate the different behaviors of the system for each possible system load output. Results shown in Figure 2.8 demonstrate that the system converges to ensure the router SLA. Depending on the load level of the system, the punishment level varies. We can see that in the low load environment, the system takes up to 40 seconds to reach the SLA. There is plenty of resources and so the

SLA violation does not harm other routers. When we use the controller in the medium load scenario, the punishment level is intensified and in less than 15 seconds the abuse is contained. In the high load scenario the punishment is severe and the system limits resources within 5 seconds. We can see that the proposed controller is efficient and fulfills the established requirements, acting conservatively way in low load scenarios and acting aggressively in critical situations.

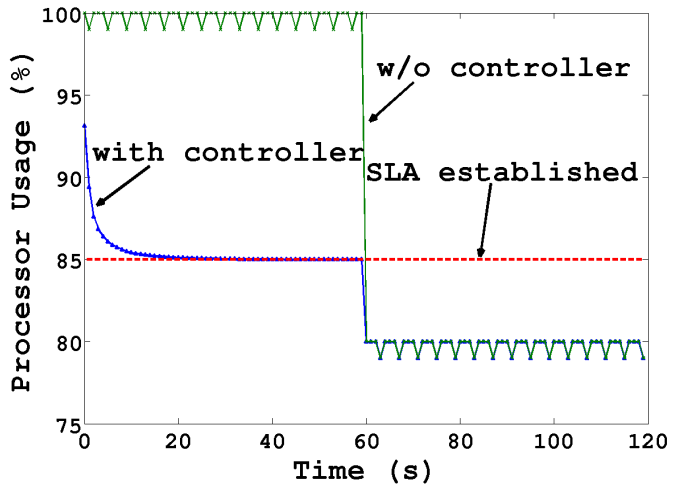


Figure 2.9: processor utilization of a virtual router that violates a SLA for a given period, with medium system load.

The third experiment evaluates the controller efficciency for transient misbehavior, when a given virtual router violates the SLAs for a given period of time and then it starts to respect the contracts. The virtual router forwards packets at a high packet rate and uses 100% of its processor. After 60 seconds, the router forwards packets at a lower rate and consumes up to 80% of its processor, respecting the SLA. In this result, the system load is controlled and is classified as medium. Figure 2.9 shows that without the controller the virtual router can use whatever it wants to, possibly harming other routers. When the controller is active, the virtual router suffers a gradual caps reduction until the machine starts respecting the SLA.

2.4 Dynamic SLA Controller Conclusions

We have developed an efficient fuzzy controller for SLA control of virtualized network environments, where isolation prerogatives represents a huge

challenge. The results show that the developed system is efficient and it is compatible with existent resource control techniques. The network managers can easily insert rules that reflect their particular qualitative decision making strategies. The results obtained show that the system can control the SLAs in an efficient manner by punishing routers that violate rules according to the system load and the level of violation. In the experiments, the system has successfully limited the SLAs adaptively. When there is plenty of resources, the system applies light punishments and at critical moments the system applies severe punishments. The results also show that the controller adequates the resource use in less than five seconds. In a low load environment the system converges within 40 seconds. Besides, the monitoring and management generate only a small overhead in the control domain (5% of a single processor for each virtual router managed). Later, the system will aggregate decision algorithms based on the migration of virtual routers to ensure proper resource allocation with broader possibilities.

Chapter 3

Mechanism to predict the need for update of local information

This chapter describes ADAGA, an Anomaly Detection for Autonomous manaGement of virtuAl networks system, which provides mechanisms for collecting and analyzing data in virtual network environments. Through ADAGA, monitoring managers observe the monitored systems in the network, such as servers and routers, physical or virtual. The main objective of the proposed system is to send alarms to report possible anomalies at the network equipment. This study characterizes the anomalous behavior as short-term changes in the observations that are inconsistent with past measures.

ADAGA uses time series to predict the actual value considering the past of the series and to compare with the new observation. ADAGA considers that all time series initiat at zero. For that, the initial value of each series is decremented of all its observations values, what allows zero error during the initialization and no influence of the initial condition on future predictions. The correct predictor initialization is an important configuration, because it impacts the performance of the whole system [11].

The predictor analysis is based on the false positives and false negatives values when predictor parameters are varied. This report analyzes the behavior of the monitored systems, describes anomalous situations, and shows the impact on the emission of alarms when an anomalous situation happens. The experiments are performed on a real router and the anomalies were artificially generated to simulate an overload in the router. The results show that ADAGA system detects anomalies presenting average false positives and negatives rates.

This chapter is organized as follows. Section 3.1 discusses the works related with the autonomous management and anomaly detection. Section 3.2

describes the developed system and its modules. Section 3.3 presents the test bed scenario of the experiments with the implemented prototype and the results obtained. Finally, Section 3.4 concludes this work and presents directions for future work.

3.1 Related Work

Anomaly detection-based virtual network monitoring and management is not a topic well exploited. Anomaly detection techniques are commonly used in security such as intrusion detection systems [12] and have also been used in autonomous systems for network management, which are triggered by the detection of an anomaly and the emission of an alarm. Anomalies can be classified in three types [13]: anomalies in network operation, which consist of network equipment failures or configuration changes; anomalies caused by a flash crowd, which usually happens when a particular location information is requested by many users at the same time, such as the distribution of a new version of an operating system or a viral video; and anomalies caused by network abuse, such as denial of service attacks and port scanning. The proposed system considers all of these anomalies because they are significant for the users' satisfaction of autonomous virtual networks management.

Brutlag [11] uses time series and predictive mechanisms for anomaly detection in computer networks. The author focuses on the analysis of network traffic anomalies from a router to generate alarms. Results for single parameter configuration of the predictor are presented with initialization at zero, without any data pre-processing in series. In ADAGA, we pre-process the received data in order to obtain zero error at start up. Moreover, the key difference between ADAGA and Brutlag's system is the multidimensional analysis, where the calculations of time series are applied to different network characteristics such as memory and processor utilization, network traffic, and system load. In virtualized scenarios, because of low network isolation not only the network traffic impacts the network performance, but also processor and memory utilization. Thus, all of these ADAGA monitored metrics are important in virtualized scenarios [14]. Besides, the network management operations such as virtual machine migrations, impact the network equipment operation [6].

Lucena and Moura analyze network traffic focusing on packet-flow based observations [15]. The authors define flows by a 4-tuple (source IP address, destination IP address, source transport port, and destination transport port). The flow approach defines various types of anomalies, such as Denial of Service (DoS), configuration failures, and flash crowds. In the ADAGA

system, the flows are not grouped in the conventional way. The system brings together the packages by services, because our purpose is to manage the network from the point of view of their function, considering that the pluralistic network approach considers one virtual networks per service on the Future Internet [16]. Therefore, in the ADAGA system, the groups of packages are made taking into account protocol number and destination transport port, because they are packet characteristics which define the flow service.

Several works related to anomaly detection present observation intervals of the order of units or tens of minutes [11, 15, 17, 18], which reduce processing and storage required. Measuring ranges of this magnitude, however, do not allow a quick reaction to important anomalies that happen in a short period of time. The ADAGA system offers observations spaced approximately by 15 seconds, and, therefore, allows quick detection and reaction to anomalies. The experiments with the prototype present satisfactory results, since the process of collection and analysis is accomplished in real time and the analysis of 40 different characteristics takes 10^{-4} seconds with a Intel Core i7 950 processor.

Besides the sampling interval, another important characteristic of intrusion detection systems is the packet sampling rate. Systems that require per-packet processing have high processing and storage loads if they evaluate all packets. For this reason, several works perform packet sampling [18, 19, 20]. Nevertheless, packet sampling introduces distortions, noise, and smoothing on the observations [21]. Recent proposals address this problem through tagging potentially anomalous packets for further analysis in other network equipment [22] or through filters more efficient than random sampling [23]. Because the ADAGA system does not process packet by packet, but uses the statistics collected, it does not perform sampling and all packets are considered. So ADAGA does not receive these disruptions.

Following the chain of autonomous management processes described by Dobson et al. [24], i.e., collection, analysis, decision, and acting, the anomaly detection requires discovering the root cause of the anomaly. The root cause is obtained from the most recent observations of the network [17, 25]. ADAGA does not address discovering the root cause of the anomalies, but sends the latest observations with all collected data to be processed in other root-cause discovering mechanisms.

3.2 ADAGA System

The ADAGA (Anomaly Detection for Autonomous manaGement of virtuAl networks) system provides mechanisms for collecting and analyzing data

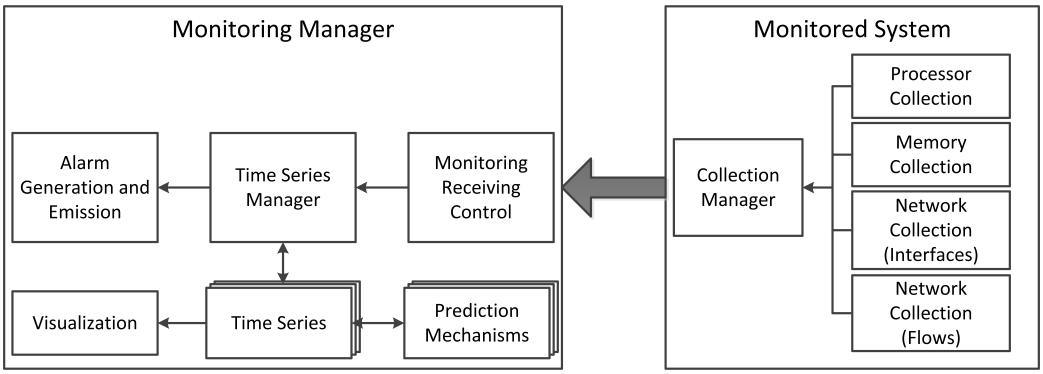


Figure 3.1: ADAGA system architecture.

in a virtual network environment and gives support to the autonomous management of virtualized networks. The proposed system aims at detecting anomalies on virtual networks and also activating anomaly fixing mechanisms. Figure 3.1 presents the architecture of ADAGA, as well as the modules interconnection and the communication among the monitored systems and manager.

An ontology to represent and organize the knowledge of the network at any given moment is defined in Subsection 3.2.1. The network administrator selects the relevant characteristics that must be monitored . Each of the characteristics defined for monitoring is organized into a time series, as defined in Subsection 3.2.2. All the defined time series have an associated prediction mechanism, as described on Subsection 3.2.3, which is responsible for measuring the deviation of the current measure in this characteristic past, generating alarms described on Subsection 3.2.4. The virtualization module is responsible for generating graphics in the time series evolution and of the predictor error, as the graphics presented on the Subsection 3.3.1.

3.2.1 Data Collection and Representation

The ADAGA system performs the data collection from network equipment through remote data requests on the *eXtensible Markup Language* (XML) format. The monitoring manager enquires monitored systems that can be either physical or virtual elements in the network. Each monitored system executes the monitoring agent that, upon receiving the requests, invokes several specific agents to obtain specific data such as processor utilization, memory utilization, and network state from the monitored system, as well as the Figure 3.1 presents.

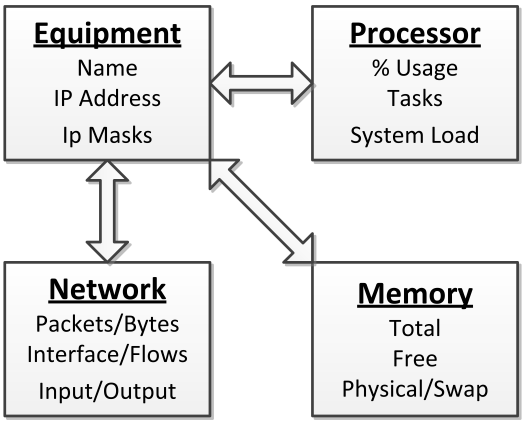


Figure 3.2: Representation structure of network components in ADAGA system.

The observation metric is fundamental in a network monitored scenario [26]. Our prototype collects observations through available tools of Linux operating system, which is the base-system used in ADAGA, and of Xen [27], which is the chosen virtualization platform. Figure 3.2 presents a simplified representation structure for the network elements in ADAGA.

The network elements modeling considers multi-core processors, RAM mechanisms, i.e., physic and virtual memory, and several network interfaces. The network equipment model includes identification and location data of equipment and connection with different processors, memory and network interface models. Each characteristic represented at the model participates in the anomaly detection as distinct time series. In the ADAGA system, the calculations described on Subsection 3.2.3 are independently applied for each model characteristics.

We claim that an effective anomaly detection system must be able to detect an anomaly by only analyzing the data statistics instead of the real collected data. This feature enables the proposed system to have a more effective processing and storage, in addition to avoiding problems with privacy of the data packets.

3.2.2 Time Series

According to Brockwell and Davis [28], time series are a set of observations s_t, each one of them performed in a specific instant $t \in T$, where T is the finite group of the measurement times. The difference between a time series and a common group of values is that, on the time series, the order among

the observations is important. There are two types of time series considering the observation process. In discrete time series, the observations have defined times and are performed in specifics moments of time. On continues time series, the observation is performed continuously during an interval of time. In this work, discrete time series are used.

We define the set T by the sequence of observation times. We also consider that sampling interval between the observations follows the Poisson distribution with center in 15 seconds. According to Paxson [29], a fixed sampling interval may cause distortions in the observations because it can be synchronized with an unpredictable event and it is unable to correctly observe periodic behaviors on the network. These problems reduce detection accuracy or hide the anomalies.

In time series, at any given specific time t, it is possible to predict the next value s_{t+1} of the series based on the history of the series (s_1, s_2, ..., s_t). Based on this prediction and on the real value observed at the time $t+1$, it is possible to define the predictor error as

$$\epsilon_t = |\hat{s}_{t+1} - s_{t+1}| \,. \tag{3.1}$$

If these errors are bigger than the tolerance defined to the prediction, ADAGA triggers the alert generation module, as described on the Subsection 3.2.4.

In ADAGA, the time series management module is responsible for feeding and controling the time series on-the-fly with the received observations. This module defines the monitored characteristics of each router, the predictor parameters of each characteristic and the insertion of new values on the characteristics time series.

3.2.3 Prediction Mechanisms

Prediction mechanisms determine the next value of each time series. Two prediction mechanisms are considered: the simple and well used predictor called *Exponential Smoothing* and *Holt–Winters Seasonal*, which considers the trend and the seasonal components of time series.

Exponential Smoothing

The prediction mechanism *Exponential Smoothing* is a simple algorithm to calculate the next value in a time series, which is based on the moving average of the history of the series [11]. In order to calculate each next value $\hat{s}_{t+1}$, the current measured value s_t and the prediction calculated for the

current value $\hat{s}_t$ are

$$\hat{s}_{t+1} = \alpha s_t + (1 - \alpha)\, \hat{s}_t, \tag{3.2}$$

where $\alpha \in [0, 1]$ and $\hat{s}_1 = s_1$.

The α parameter is the weight of the current value in relation with the history of the series. Therefore, the bigger the α value, the smaller the influence of the past of the series on the calculation of the predicted value. According to the analysis of Lucena and Moura [15], on the computer networks scenario, appropriated values for the α parameter must be lower than 0.1.

Due to the recursions performed as from $t \geq 2$, the influence of the past of the series is reduced throughout the time, following the expression given by

$$\hat{s}_t = \left[\sum_{j=0}^{t-2} \alpha(1 - \alpha)^j s_{t-j} \right] + (1 - \alpha)^{t-1} s_1. \tag{3.3}$$

Therefore, Figure 3.3 shows that the influence of each past observation on the prediction calculation declines exponentially, except by the first value. Thus, the initial condition strongly impacts the results, as it is observed on Figure 3.3(a). Hence, to reduce the impact of the initial condition on the prediction of values, this work applies a conversion on the values of the series such that

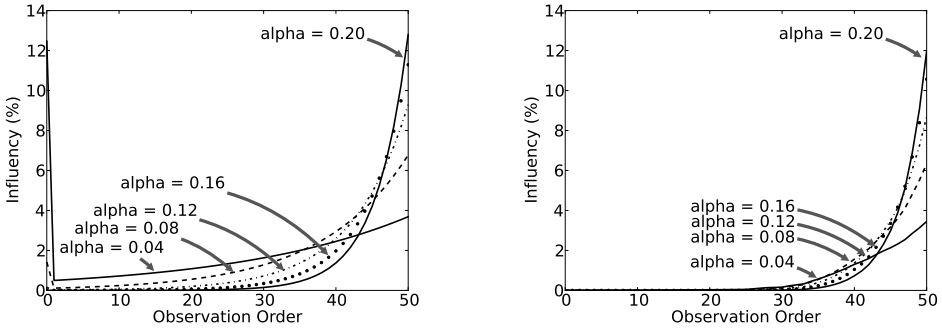
$$\forall s_t, s_t = s_t - s_1. \tag{3.4}$$

Therefore, the initial configuration will be $\hat{s}_1 = 0$ without prediction error, as presented on Figure 3.3(b).

Holt–Winters Seasonal

The *Exponential Smoothing* mechanism is not suitable for time series presenting periodical behaviors, called seasonality, because it assumes linear serie values and approximates the next value with the moving average of the series history. The *Holt–Winters Seasonal* mechanism is a predictor well adapted to seasonal time series. Brutlag [11] defines a model of seasonality for the behavior of computer network, which consists of a greater activity during the morning than during the night.

The *Holt–Winters Seasonal* decomposes the time series in tendency, seasonality and noise. Each one of these components is handled as a variation of the *Exponential Smoothing* method. There are two methods for aggregating these components into a predicted value: additive and multiplicative [30]. These components are aggregated with the additive method when the statistic variation of the period does not depend on the series. In case it depends,



(a) Collected data.

(b) Collect data with the proposed transformation applied.

Figure 3.3: Influence of the past observations in the prediction of the $t = 50$ observation for the transmitted packet time series.

the components are multiplied. Lucena and Moura [15] claim that, computer networks, the additive composing of the components presents better results and, thus, the next value prediction is

$$\hat{s}_{t+1} = R_t + T_t + P_{t+1-m}, \tag{3.5}$$

where T_t represents the tendency of the time series, P_{t+1-m} the periodic component, where m is the seasonality period, and R_T the series aggregated noise. Then, the prediction equations of each one of these values are given by

$$R_t = \alpha\,(s_t - P_{t-m}) + (1-\alpha)\,(R_{t-1} + T_{t-1}) \tag{3.6}$$

$$T_t = \beta\,(R_t - R_{t-1}) + (1-\beta)\,T_{t-1} \tag{3.7}$$

$$P_t = \gamma\,(s_t - R_t) + (1-\gamma)\,P_{t-m}. \tag{3.8}$$

where α, β, and $\gamma \in [0, 1]$ represent the smoothing constants for each component of the predicted value. Similarly to the *Exponential Smoothing* mechanism, these parameters represent the weigth of the past series on the predicted value calculation. The bigger the constant values, the smaller the influence of the past component on the prediction.

3.2.4 Alarm Generation

Alarm generation occurs when the predicted error is bigger than the calculated accepted error. The accepted error is recalculated for each new collected

observation and defined by

$$\epsilon_t = \delta\Psi_t, \tag{3.9}$$

where δ is an amplification constant of the error acceptance and Ψ_t is defined according to the predictor mechanism. Brutlag [11] claims that optimum values of δ belong to the interval $[2, 3]$. ADAGA uses $\delta = 2$, because we propose a sensitive system and $\delta = 2$ generates a smaller acceptance error.

For the *Exponential Smoothing* mechanism, Ψ_t is the standard deviation of the values considering the observation window and the next value prediction. For the *Holt-Winters Seasonal* mechanism Ψ_t is

$$\Psi_t^{HOLT} = \gamma(|s_t - \hat{s}_t|) + (1 - \gamma)(\Psi_{t-m}^{HOLT}). \tag{3.10}$$

The ADAGA system proposes an alarm control with the cumulative emission of alarms. Then, not all generated alarms are emitted to remove punctual alarms of the system. The implementation of this method in ADAGA uses a hysteresis in order to define the emissions of generated alarms. If the system detects an anomaly during the observations, this one is not emitted immediately. Only after the accumulation of η generated alarms, one alarm starts to be emitted. The value of η is defined by the network administrator. If the system stops detecting anomalies before η generated alarms, the anomaly is not reported and the alarm accumulation counter is set to zero.

An alarm emission means a report sending to the decision system and acting on the network. This report consists of a group of the latest 15 observations of all characteristics of the network element which generated the alarm, the information of predictor value, the real observed value, and the characteristic that generated the alarm.

3.3 Evaluation

The ADAGA system was evaluated to determine its capacity to detect anomalies. The evaluation considers false positives and false negatives rates. False positives mean that alarms were wrongly emitted because there was no anomalies and false negatives are characterized by observations at moments with anomalies that do not generate alarms, including, therefore, the alarms accumulated during the period of alarm generation hysteresis.

A developed prototype analyzes the ADAGA system. We observe the testbed scenario performed to this report in the Figure 3.4. The monitoring manager executes the analysis mechanism of ADAGA, which consists of the management of the time series, predictions and alarms generation. The monitored system has mechanisms to collect information about its executions. The information collection is periodic and the collecting interval

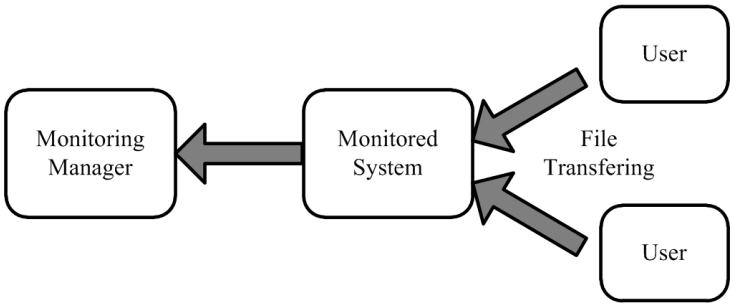


Figure 3.4: Testbed scenario for analysis of false positive and false negatives in ADAGA.

follows the Poisson distribution with center in 15 seconds [29]. On the performed tests, the monitored system is a wireless network router used in a real network.

After, approximately, two days of monitoring, two nodes started two consecutive uploads of a 15 gigabytes file to the router using the *Secure Shell* (SSH) protocol. These uploads aim at turning the router inaccessible for remote access. The objective is to cause an anomaly that must be detected by ADAGA and, as a consequence, send the alarms. Besides, we evaluate the impact of each characteristic on the detection of this anomaly. As we expect, time series of network characteristics, such as network traffic in the reception interface and the flow at TCP port 22, detect the anomaly while other characteristics, e.g., number of running processes and memory utilization, do not. We observe variables like the system load, that presents a particular behavior i.e, a lot of peaks, detect the anomaly. The system load is influenced by the file transfer because the performed upload induces network and disk I/O load in the router.

3.3.1 Results

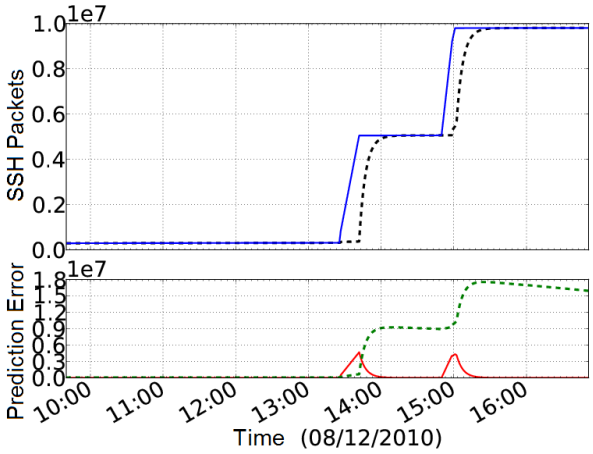
The series evolution of observations of the TCP packet receiving in SSH service are presented in Figure 3.5. Subfigure 3.5(a) shows the results for the *Exponential Smoothing* mechanism and Subfigure 3.5(b), for the *Holt-Winters Seasonal* mechanism. At the top of the graphs, there are the real observations, in solid line, and the mechanism predicted values, in dashed line. From this graph, we observe that the *Exponential Smoothing* mechanism predicted value easily follows the real value evolution, which makes anomaly detection more difficult, even for anomalies of this magnitude. Differently, the *Holt-Winters Seasonal* mechanism has higher difficulty in follow-

ing the observed values, showing better anomaly detection efficiency. When a predictor is able to follow abrupt changes, it will have low efficiency for detecting anomalies, which are abrupt changes. The bottom part of the graphs presents the values directly connected with the anomaly detection. The solid line represents the prediction error of the mechanism, obtained according to Equation 3.1 and the dashed line shows the accepted prediction error that is calculated by Equation 3.9. For the best view in the report, these graphs exhibit only briefly near the anomaly generated.

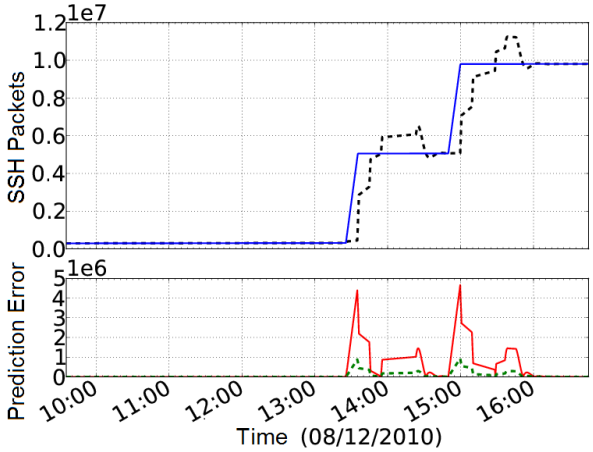
In Figure 3.6, we observe the historical evolution of two characteristics monitored by ADAGA, i.e., the average load of the monitored system in the five minutes before each collection of data, in Subfigure 3.6(a), and the percentage of the processor utilization, on Subfigure 3.6(b). From the graphics analysis, we observe the characteristics, such as the system load, that suffers influence of network traffic anomalies. This influence confirms the importance of the multidimensional monitoring of the network equipment. Anomaly signals in several characteristics are useful to trace the cause of anomalies, functionality which is not available at the ADAGA yet, but is an object of study in future works. Characteristics such as the processor utilization are not influenced by the generated anomaly as shown on the Figure 3.6(b).

From analysis of graphs on Figure 3.5, we wait that the rates of false positives and false negatives of the *Holt-Winters Seasonal* mechanism would be better than the rates of the *Exponential Smoothing* mechanism. Indeed, by the graphs on Figure 3.7, we notice the improvement on the rates of false positives and false negatives. In Figure 3.7, we observe the percentage of false positives and false negatives of several parameter configurations of the prediction mechanisms as we change the η parameter, which represents the quantity of alarms accumulated before the report emission. Each one of the presented curves represents a configuration of parameters of the predictive mechanisms. In the case of *Exponential Smoothing*, the evaluated values of α were 0.05, 0.10, and 0.15. The same values were applied to the parameters α, β, and γ of the *Holt-Winters Seasonal* mechanism and it resulting 27 curves, that are represented on Subfigures 3.7(c) and 3.7(d). All graphs of Figure 3.7 are considered from an observation window of 2000 samples. The tests performed for others sizes of observation window presents similar results in the induced anomaly detection and are not in this report for the sake of brevity.

In the analysis undertaken in this report, the false positive rate is calculated over all the collected observations, trying to get the amount of alarms that were emitted in moments without anomalies. The false negative rate was obtained considering the alarms that were not emitted during moments



(a) *Exponential Smoothing* mechanism.

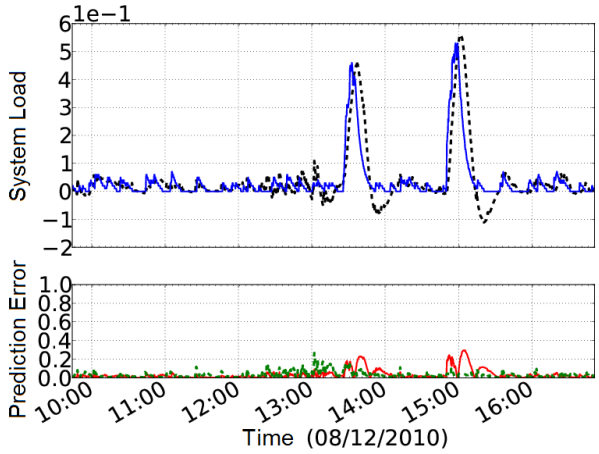


(b) *Holt-Winters Seasonal* mechanism.

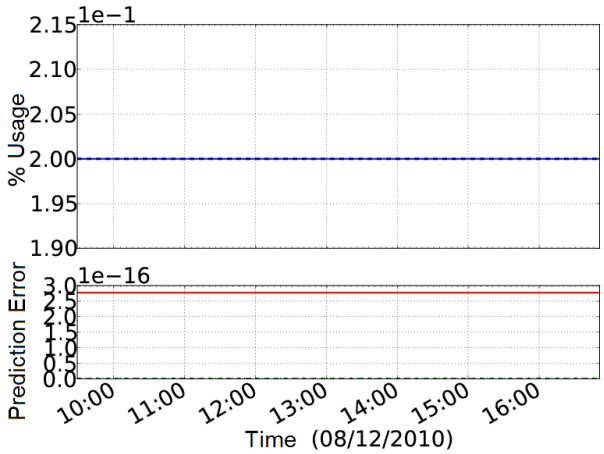
Figure 3.5: Temporal evolution of TCP packet receiving in SSH service for each mechanism. Observation window of 2000 samples and α, β, and γ are equal to 0.1.

with anomalies over all collected observations in this moment.

Based on the adopted definition, we attest the low effectiveness of the *Exponential Smoothing* mechanism, because, although it has low false positives rates (Figure 3.7(a)), it is not able to detect the generated anomaly. Nevertheless, all the parameters configurations evaluated on the *Holt-Winters*



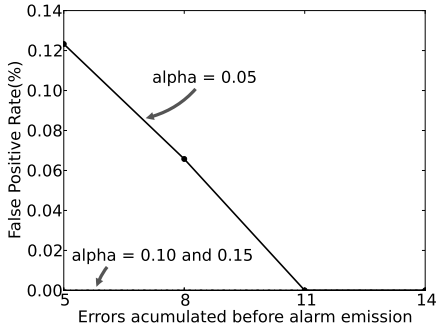
(a) Average load of the system in five minutes before each observation.



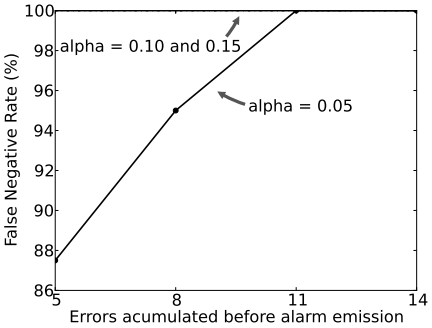
(b) Percentage of processor utilization on the router.

Figure 3.6: Temporal evolution of two characteristics not directly related to network traffic for the *Holt-Winters Seasonal* mechanism. Observation window of 2000 samples and α, β, and γ are equal to 0.1.

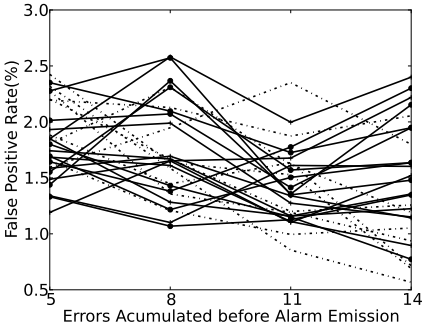
Seasonal mechanism presented a good effectiveness, with false positive rates lower than 2.5%, and false negative rates appropriated with the defined accumulation of alarms. On the interval where the anomalies happen we collected 40 samples. Therefore, a false negative rate equal to 2.5% of the errors accumulation value means that for all observations where there was anomalies,



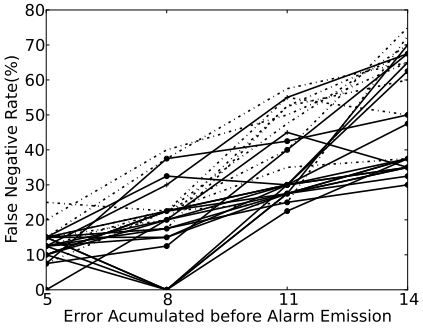
(a) False positives of *Exponential Smoothing* mechanism.



(b) False negatives of *Exponential Smoothing* mechanism.



(c) False positives of *Holt-Winters Seasonal* mechanism.



(d) False negatives of *Holt-Winters Seasonal* mechanism.

Figure 3.7: Analysis for the TCP packet receiving in SSH service characteristics of each mechanism, when α, β, and γ parameters change. Observation window size is 2000 samples.

the system could detect the anomalies with the *Holt-Winters Seasonal* mechanism. In fact, that is the case, because, for instance, for the value $\eta = 5$, we have the expected false negatives rate about of 12.5%, value that concentrates a big part of the analyzed curves.

In order to have a more detailed analysis, we observe the false positive rates in Table 3.1 and the false negative rates in Table 3.2 for all the parameter configurations evaluated for the *Holt-Winters Seasonal* mechanism. The rows are the false positive and negative values for each value of α and the columns presents the values of false positives and negatives for each value of γ, grouped by the values of β. From these tables, it is possible to conclude that there is a big similarity of the results for the different configurations. Nevertheless, we observe a minor gain for values of smaller β. As seen on Sub-

β	0.05			0.10			0.15		
$\alpha \setminus \gamma$	0.05	0.10	0.15	0.05	0.10	0.15	0.05	0.10	0.15
0.05	1.69	1.34	2.19	1.64	1.49	1.81	1.44	1.85	1.59
0.10	1.33	1.19	2.20	2.35	1.85	2.28	1.55	1.93	2.42
0.15	1.80	1.59	1.65	2.28	1.74	1.88	2.01	1.68	2.32

Table 3.1: Comparative table of false positive rates for the α, β, and γ configurations presented in the Figure 3.7(c) with $\eta = 5$. Values are expressed in percentage.

β	0.05			0.10			0.15		
$\alpha \setminus \gamma$	0.05	0.10	0.15	0.05	0.10	0.15	0.05	0.10	0.15
0.05	15.0	12.5	7.5	10.0	15.0	12.5	7.5	15.0	10.0
0.10	12.5	15.0	25.0	0	12.5	15.0	15.0	10.0	15.0
0.15	7.5	12.5	15.0	12.5	10.0	15.0	15.0	10.0	20.0

Table 3.2: Comparative table of false negative rates for the α, β, and γ configurations presented in the Figure 3.7(c) with $\eta = 5$. Values are expressed in percentage.

section 3.2.3, the β parameter is related with the series tendency. Therefore, the more the predictor accepts drastic changes on the time series tendency, i.e., β big, the quicker the predictor is able to adapt itself to the anomaly and, as a consequence, the smaller are the rates of success.

3.4 ADAGA Conclusions

In this chapter, we describe the ADAGA, an Anomaly Detection for Autonomous manaGement of virtuAl networks system. Our system aims to collect and analyze data from virtual networks. The major advantages of ADAGA are multidimensional monitoring, lower monitoring interval, which is responsible for lower reaction times, and avoidance of sampling strategies for packet processing. Because the system does not consider the data packets as a whole, but only its statistics and counters, the overhead is minimal. The ADAGA system provides two mechanisms for time series prediction that are explored in the literature. Nevertheless, the results show that the *Exponential Smoothing* mechanism is not suitable for computer network predictions, because these networks presents seasonality that is not accounted by this mechanism. The results also demonstrate that the multidimensional analysis

performed by ADAGA is an important ally to the autonomous management, since several characteristics of the monitored equipment can trigger alarms at the same time, which facilitates to trace the root cause of anomalies, that is object of our future work. The analysis of false positive and false negative results that the mechanism *Holt-Winters Seasonal* has excellent efficacy, because it detects the network anomalies with low false positive rates and with false negative rates expected considering the hysteresis system for alarm emission of ADAGA.

Chapter 4

Conclusion

All the presented results allow users and agents to be aware of the need of updates or virtual elements. The knowledge plane comprises information regarding all environment variables and detects the need to take decisions and pilot the networks. The two approaches presented in this report fulfill those requirements related to the need of updates. The first approach monitors and manages the use profile and SLAs to maintain QoS in networks and to estimate the need of SLA updates to obtain each virtual router demands. By generating the use profiles, it is possible to detect violations and guarantee SLA constraints. The proposed architecture also fills the knowledge plane with the use profiles and their associated time series sliding windows. When there are violations, they are reported to the knowledge plane as well. The second approach monitors several parameters and use advanced algorithms and estimators to predict the future behaviors of machines, to detect possible changes and, then, to trigger alarms to force the system to keep the information refreshed in the knowledge plane. The proposals work together with existent tools developed earlier in the Horizon project and allow an effective manner to proactively monitor routers and detect the need for updates.

Bibliography

[1] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. Networked Systems Design and Implementation*, 2007.

[2] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[3] D. Menasce and M. Bennani, "Autonomic virtualized environments," in *Autonomic and Autonomous Systems, 2006. ICAS'06. 2006 International Conference on*, p. 28, IEEE, 2006.

[4] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "Autonomic resource management in virtualized data centers using fuzzy logic-based approaches," *Cluster Computing*, vol. 11, no. 3, no. 3, pp. 213–227, 2008.

[5] Y. Wang, J. van der Merwe, and J. Rexford, "VROOM: Virtual routers on the move," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, Citeseer, 2007.

[6] P. Pisa, N. Fernandes, H. Carvalho, M. Moreira, M. Campista, L. Costa, and O. Duarte, "OpenFlow and Xen-Based Virtual Network Migration," *Communications: Wireless in Developing Countries and Networks of the Future*, pp. 170–181.

[7] N. C. Fernandes and O. C. M. B. Duarte, "XNetMon: Uma Arquitetura com Segurança para Redes Virtuais," in *Anais do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg'10*, (Fortaleza, CE, Brazil), pp. 339–352, Oct. 2010. (in portuguese).

[8] E. Keller, R. Lee, and J. Rexford, "Accountability in hosted virtual networks," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 29–36, ACM, 2009.

[9] V. Kecman, *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. The MIT press, 2001.

[10] L. Zadeh, "Fuzzy sets*," *Information and control*, vol. 8, no. 3, no. 3, pp. 338–353, 1965.

[11] J. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proceedings of the 14th USENIX conference on System administration*, 2000.

[12] A. Patcha and J. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, no. 12, pp. 3448–3470, 2007.

[13] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 69–73, ACM, 2001.

[14] N. Fernandes, M. Moreira, I. Moraes, L. Ferraz, R. Couto, H. Carvalho, M. Campista, L. Costa, and O. Duarte, "Virtual networks: Isolation, performance, and trends," *Annals of Telecommunications*, pp. 1–17, 2010.

[15] S. de Lucena and A. de Moura, "Análise dos Estimadores EWMA e Holt-Winters para Detecção de Anomalias em Tráfego IP a partir de Medidas de Entropia," *CSBC2009*, 2009. (in portuguese).

[16] M. Moreira, N. Fernandes, L. Costa, and O. Duarte, "Internet do futuro: Um novo horizonte," *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2009*, pp. 1–59, 2009. (in portuguese).

[17] F. Silveira and C. Diot, "URCA: Pulling out anomalies by their root causes," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[18] F. Silveira, C. Diot, N. Taft, and R. Govindan, "ASTUTE: Detecting a different class of traffic anomalies," in *Proceedings of the ACM SIGCOMM 2010 Conference*, (Nova Deli, India), ACM, September 2010.

[19] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, p. 31, USENIX Association, 2005.

[20] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 71–82, ACM, 2002.

[21] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 159–164, ACM, 2006.

[22] S. Ali, I. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S. Khayam, and F. Mirza, "On mitigating sampling-induced accuracy loss in traffic anomaly detection systems," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, no. 3, pp. 4–16, 2010.

[23] D. Brauckhoff, K. Salamatian, and M. May, "A signal processing view on packet sampling and anomaly detection," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, IEEE, 2010.

[24] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, no. 2, pp. 223–259, 2006.

[25] I. Paredes-Oliva, X. Dimitropoulos, M. Molina, P. Barlet-Ros, and D. Brauckhoff, "Automating root-cause analysis of network anomalies using frequent itemset mining," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, no. 4, pp. 467–468, 2010.

[26] A. Ziviani and O. Duarte, "Metrologia na Internet," *Minicursos do XXIII Simpósio Brasileiro de Redes de Computadores, SBRC*, pp. 285–329, 2005. (in portuguese).

[27] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 164–177, ACM, 2003.

[28] P. Brockwell and R. Davis, *Introduction to time series and forecasting*. Springer Verlag, 2002.

[29] V. Paxson, "On calibrating measurements of packet transit times," in *SIGMETRICS/PERFORMANCE: Joint International Conference on Measurement and Modeling of Computer Systems*, (Madison, WI), 1998.

[30] A. Koehler, R. Snyder, and J. Ord, "Forecasting Models and Prediction Intervals for the Multiplicative Holt-Winters Method," *Monash Econometrics and Business Statistics Working Papers*, 1999.