A P2P-Based Self-Healing Service for Network Maintenance

Pedro Arthur Pinheiro Rosa Duarte, Jéferson Campos Nobre,
Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil
Email: {paprduarte, jcnobre, granville, liane}@inf.ufrgs.br

***Abstract*—The introduction of self-* properties over distributed network management infrastructures has been proving to be a feasible approach for the new demands of modern network management. Among the properties of the self-* management vision, self-healing figures as key property in improving the dependability of the managed infrastructures. An interesting possibility to materialize self-* support - and self-healing support as well - in network management is through the employment of peer-to-peer (P2P) management overlays. Considering this scenario, we introduce in this paper a self-healing service provided by a prototype P2P-Based Network Management (P2PBNM) system. Such a service is expected to be contracted by human administrators interested in monitoring and recovering their IT infrastructures. In addition, an experimental evaluation of the self-healing service is performed considering a case study where a Host-based Intrusion Detection System (HIDS) needs to be constantly observed and eventually healed to keep the underlying communication network protected.**

I. Introduction

In recent years, communication networks have increased in size, complexity, and heterogeneity. Moreover, the requirements of provided services have become more diverse and resource demanding [1]. In such scenario, where computer networks turns to be sophisticated and complex, solutions to manage the underlying communication infrastructure and help network human administrators in their daily tasks are crucial.

The introduction of distributed technologies in network management (*e.g.*, Management by Delegation - MbD) has lead to improvements over the traditional centralized management approach, for example, when communication networks grow in size. However, the complexity of modern networks demands management features that are not present, at least not explicitly, in usual distributed network management technologies [2].

A recent alternative that complements traditional distributed management technologies consists in the employment of peer-to-peer (P2P) management overlays [3]. Such overlays merge characteristics of P2P and network management systems, better enabling, for example, collaborative management [4], more robust connectivity among management entities, and improved load balance of management tasks at management peers [5].

P2P-based network management also enables embedding self-* properties from the Autonomic Computing (AC) paradigm into the management overlay [6] [7]. The employment of self-* properties for network management, commonly referred as Autonomic Network Management (ANM), increases the efficiency of network human administrators through the facilitated automation of management tasks; such an automation reduces the number of manual interventions during the execution of management tasks, thus freeing network administrators to deal with the high-level management goals [8].

Self-healing is one of the key self-* properties, which aims at automating the failure detection and handling, improving the dependability of the communication network infrastructure [9] [10]. In addition, self-healing mechanisms show a great potential to reduce the *Total Cost of Ownership* (TCO) of the infrastructure, which is important because 50% of this cost, according to Fox *et al.* [11], is spent on fault prevention, diagnosis, and repair.

Although there are interesting investigations on self-healing mechanisms, a *de facto* solution employing self-healing in P2P-based network management systems is missing, remaining as an open research problem requiring further investigation. For example, issues encompassing scalability, heterogeneity, and distribution of the self-healing mechanisms in P2P management overlays still need deeper observation from the network management research community.

In this paper, we introduce a distributed self-healing mechanism that runs over a P2P-based network management system. Our mechanism uses monitoring and healing workplans to deal with the native heterogeneity of networks. These plans are descriptions that capture the knowledge of systems administrators on how the managed devices/systems shall be monitored and healed. In order to sustain its design principles, we experimentally evaluate our proposal using it as a self-healing support for a Distributed Intrusion Detection System.

The main contributions of this paper are the following. We propose a *self-healing P2P mechanism that abstracts the monitoring and healing* of managed elements. Furthermore, the proposed mechanism is *totally decoupled from managed elements.* Finally, we also present *workplans*, descriptions written in a high-level language that aims to gather the knowledge of the administrators on how network devices and systems are maintained.

978-1-4244-9221-3/11/$26.00 ©2011 IEEE

The remainder of this paper is organized as follows. In Section II, we present the state-of-the-art on P2P-based network management systems, and self-healing approaches. In Section III our proposal and associated concepts are described. A system prototype, implemented to help in our research, along with its application programming interface (API) used by developers to code P2P-based network management software, are presented in Section IV. In sections V and VI, we present a case study and a experimental evaluation of our proposal. Conclusions and future work are provided in Section VII.

II. Related Work

In this section we first discuss about self-healing approaches to then review the state-of-the-art on P2P-based Network Management.

A. Self-healing Approaches

PANACEA is a framework for the development of self-healing enabled systems [12]. The approach of this framework is based on the design-time system instrumentation through the use of code annotations. These annotations serve later as an interface for agents to monitor, manage, configure, and heal systems at runtime. The PANACEA framework present improved monitoring performance when compared to other approaches, such as the Glassbox Inspector framework [13]. Despite this improvement, preexisting applications need to be redesigned or instrumented with self-healing enabling components to take advantage of PANACEA framework features. Moreover, the distributed interaction among healing agents must be hard coded into the application and tightly coupled with the service for which it is being designed.

Another approach for the development of self-healing mechanisms is to keep different system models running in parallel [14]. Their objective is to feed monitoring components that evaluate the system at runtime and trigger corrective actions as needed. For its application in network systems, models are used as nodes of an *interaction graph* that, through its arcs, represent the collaboration of the components of the system. The constrains, bounds, and corrective steps for problems of the system are denoted as annotations in the interaction graph. The main advantage of the models approach is to decouple the self-healing mechanisms and the application, enabling to change the healthily concepts of the system as its running context change. However, this approach is suitable only for systems that provide a runtime configuration and data collecting interface. Moreover, generating consistent models is not trivial for complex networked and distributed systems.

Although practical results have not been shown yet, a promising approach is to apply model-driven aspect-oriented techniques for designing self-healing systems [15] [16]. In this approach, a concise analysis framework is adopted to model both failures detection and mitigation strategies. These models are lately used to feed a specialized, aspect-oriented code generator to provide self-healing capabilities to the modeled system though the explicit code instrumentation of entry points (*pointcuts*) provided by the models. The use of aspect orientation would permit that the system dynamically change its runtime self-healing behavior, thus providing environmental adaptation capabilities. However, this approach also employs the instrumentation of the system in design-time. Moreover, as PANACEA, this approach does not provide a native communication and coordination mechanism.

Self-healing approaches show desirable characteristics that may be introduced into network management systems, such as improvements in monitoring performance and environmental adaptation capabilities. Besides, these characteristics could be employed in addition to improvements in the infrastructure of these systems, such as the utilization of P2P overlays. In this case, however, the mechanisms that implement self-healing approaches over P2P management overlays must be performed maintaining the scalability and robustness features of these overlays.

B. P2P-based Network Management

P2P-Based Network Management (P2PBNM) extends distributed management models through the composition of characteristics of distributed management and P2P overlays [3]. A P2P overlay uses resources distributed in several peers in order to implement applications such as file sharing and distributed computing. In the P2PBNM model, peers have a double role: in addition to the execution of normal management tasks, they also act as regular peers in the P2P overlay [17]. Thus, many functions required for distributed management are intrinsically provided by P2P overlays, such as load balancing of management tasks and cooperative management [3]. Besides, P2PBNM systems use application layer routing as their main message passing resource, which makes them an appropriate choice for inter-domain management, since application layer routing protocols bypass more easily administrative domains boundaries [18].

The system administrator interacts with the management overlay to retrieve information about the network, or to contract services to be applied on the managed elements. Complementary tools, like topology maps and real-time messaging, may also be provided by the management overlay. Several works present the support of different management features in P2PBNM systems, such as *Policy-based Network Management* (PBNM). In this context, the introduction of self-* features in P2PBNM is investigated by some initiatives. For sake of simplicity, only few initiatives will be cited, as follows.

"Self-Managed Cells" (SMC) [19] are proposed as an architectural pattern for ubiquitous computing applications, aiming at different levels of scale. Each SMC is autonomous and uses policy-based techniques for driving adaptation decisions. Among different cross-SMC interactions, the authors describe P2P interactions. But, concerning to levels of abstraction, it is not clear whether different SMCs could be "peers". A managed device is logically connected with only one SMC, thus, management tasks are not evaluated in a P2P fashion. There are extension for SMCs to enable self-healing features for specific contexts, such as pervasive computing systems [20].

Fallon *et al.* [6] employ a P2P approach to the self-organization of network management topologies, the "Madeira Management System", in order to accomplish specific network management tasks. This approach uses the concept of Adaptive Management Components (AMC), which are containers that run on managed elements. AMCs can manage elements on which they are running and communicate with other AMCs running on other managed elements through P2P communication services. Despite the support for self-configuring and self-optimization, the authors do not mention the possibility of self-healing features.

Panisson *et al.* [5] propose "ManP2P", a P2PBNM system prototype. ManP2P is partially inspired by the Management by Delegation (MbD) model and it is based on a service-oriented approach. This system is designed to support self-* features through the implementation of autonomic modules in peers. These modules are designed to communicate with other components of the P2PBNM system, when necessary. The possibility of implementing self-healing features in ManP2P is discussed, however, only for the management overlay itself [21]. Besides, there is not an actual implementation of self-healing features.

The utilization of P2P overlays as an infrastructure for network management systems could offer improvements in different aspects, such as scalability and reliability. Despite many improvements brought by P2PBNM systems (*e.g.*, load balancing of management tasks), there are still issues to be addressed. For example, the mechanisms used to enable a higher availability of the network management system are not extended to the managed devices/systems [22].

III. Proposal

As earlier stated, P2P-based Network Management (P2PBNM) systems extends distributed network management systems through the composition of characteristics of distributed management models and P2P overlays. This composition can offer improvements in scalability, reliability, operational costs [3], and also permits heterogeneity among its constituent entities.

In a P2PBNM system, peers have to perform management tasks and their related provisioning details (*e.g.*, location of peers in the P2P network). From the user perspective, however, the overlay provisioning details must be transparent, requiring no knowledge about the implementation or architectural organization of nodes in the overlay topology.

Peers perform management tasks through management services in P2PBNM systems. *Management service* is a key concept of the P2PBNM model, and, the present work follows the conception of management service from Panisson *et al.* [5], which is briefly revisited along this section. In this context, the result of these services are the execution of a management task. Each Management Service has a unique service identification and is reached through the overlay communication services.

Management components are responsible for implementing the management services. These components perform their tasks in regardless of the location of peers that provide it. Management components have to advertise the P2P management overlay about the services they implement and also communicate with the other peers that provide the same service in order to perform the management tasks in a coordinated way. Besides, management components are responsible for publishing the description of their associated management services, and replying queries issued by other peers.

Peers are organized into groups according to management services they expose [5]. Thus, peers that offer a specific management service are organized into a group (without human intervention) and peers can participate of several groups accordingly to offered services [23]. Peer groups improve the availability of management services through the replication of management components in different peers. Thus, while there are peers in a group, the management service provided by this group will remain available. Figure 1 presents a general view of P2PBNM.

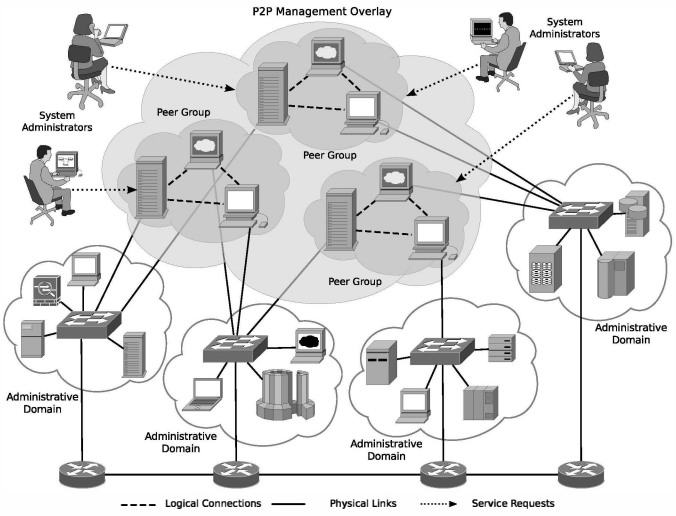

Fig. 1. P2P-based Network Management (P2PBNM) model

The application of P2P technologies improves the dependability of network management system (*e.g.*, through the replication of management components). However, more effort is necessary to improve the dependability of managed services/devices. One approach to this improvement is the utilization of self-* features, specially self-healing.

The present proposal is aimed at develop a self-healing facility on top of a P2PBNM system. This facility is implemented as a management component and divides the self-healing into two different services: *monitoring* and *healing*, both of them provided by peer groups of the P2P management overlay. The monitoring service is responsible to periodically verify the state of the managed device/system and identify anomalies/faults, all of which must be reported to the healing service. The healing service, in its turn, shall stand still waiting for anomalies/faults notifications issued by the monitoring service.

The services are implemented as managements components of the P2P management overlay. When loaded, the components

communicate with the grouping service of the overlay, identify other peers that implement the monitoring and healing service, and synchronize with them.

The binding of the self-healing service and its managed element is done at run time by the system administrator and consists of three major steps, depicted in Figure 2. First, the system administrator issues a *healing service request* to the healing group, which will return a *healing service identifier*. Second, the system administrator issues a *monitoring service request* to the monitoring group, which will return a *monitoring service identifier*. Finally, the system administrator issues a *monitoring-healing binding request* to the monitoring group, which binds a *monitoring service identifier* to a *healing service identifier*. After the binding step, the monitoring service starts to monitor the managed element. The requests are dissected and the interactions among the services are explained in the following subsections.

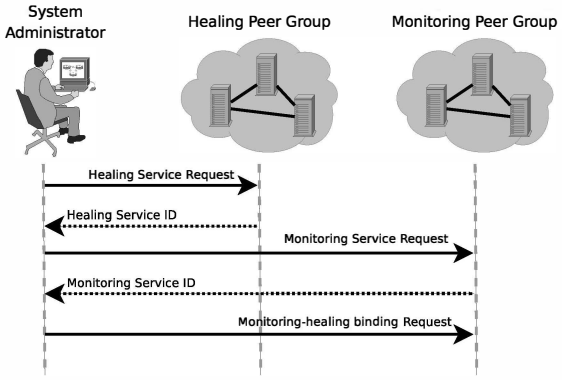

Fig. 2. Self-healing service request.

A. Monitoring Service

A *monitoring service request* is issued by the system administrators to request a monitoring service for a managed element (*e.g.*, a network device). A monitoring service request consists in a tuple $(target,\ workplan,\ period)$. The $target$ attribute of this tuple specifies the target management element of this service request and any other relevant information, like transport layer protocol and service port, or system-specific parameters. The $workplan$ attribute is the management element *monitoring workplan*, a high-level description that defines how the managed element should be monitored and which parameters identify its normal and anomalous state. The extra information passed through the $target$ attribute are used as arguments for the monitoring workplan in order to assist it in dealing with specificities of the managed element and provide more flexibility and reuse possibilities for the monitoring workplan. The $period$ attribute determines how often the target managed element shall be verified.

As response for a monitoring service request, the monitoring service group issues back a *monitoring service reply*. The content of this reply is a *monitoring service identifier* (*MsID*), which globally identify a service request in a monitoring service group. The monitoring workplan received is randomly distributed among $log_2 n$ peers, where n is the number of peers in the monitoring group.

The peers responsible for a monitoring workplan organize themselves in a logical ring and the monitoring workplan is executed every $period$ seconds in a *token-signalized round-robin* fashion. The monitoring peer that hold the token executes the monitoring workplan and i) if the results indicates that the target managed element is healthy, the token is passed for the next peer in the ring flagged as *successful*; ii) if the workplan indicates that the managed element is unhealthy, the token is passed to the next peer in the ring flagged as *unsuccessful*. The message flow during a monitoring is shown in Figure 3 A, for subsequents healthy evaluations, and Figure 3 B, for a unhealthy evaluation.

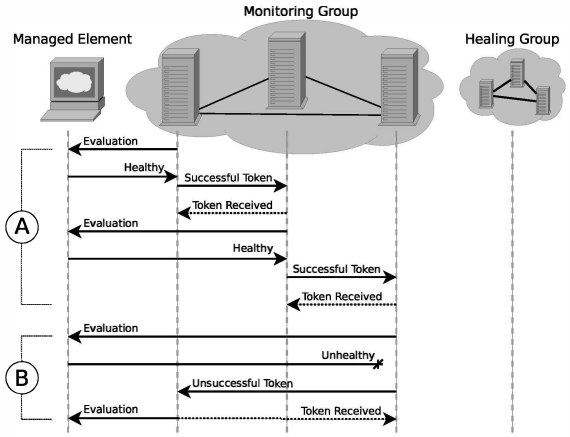

Fig. 3. Monitoring group message flow.

As shown in Figure 3 B, a peer immediately executes the monitoring workplan when it receives a unsuccessful flagged token. This is done to assure that the anomaly is not transient or related to connectivity problems between the managed element and the previous peer in the ring. In the case that the current token holder also detects the anomaly, a notification is raised to the healing service group so the healing process may be initiated. The monitoring group will stop monitoring the ill managed element until the healing group notifies it that the workplan shall be resumed or dropped.

B. Healing Service

A healing service request consists in a tuple $(target,\ workplan)$. As in a monitoring service request, the *target* attribute specifies the management element targeted in the service request and so may also contain information other than its network address. The *workplan* attribute of the tuple is the managed element *healing workplan*, a high-level description that defines how the anomalies shall be treated. The healing group sends a *healing service identifier* as response to a healing service request. This identifier is used to globally identify the healing workplan and to bind it to a monitoring service.

To improve the reliability of the healing service, a healing workplan is replicated among $log_2 n$ peers of the healing

group. Trying to attain more responsiveness and scalability, the peers that replicated the healing workplan also become able to execute it. Moreover, every peer in the group is able to map a healing workplan identifier to its responsible peers. Thus, any peer of the group may act as a notification gateway, upper bounding the healing workplan activation to a maximum of two messages exchanges. A peer that does not hold a specific healing workplan is referred as a *unaware peer.*

C. Healing and Monitoring services interaction

When a unhealthy notification comes from the monitoring group, the healing workplan is executed by one of the peers of the healing group. This is shown in Figure 4.

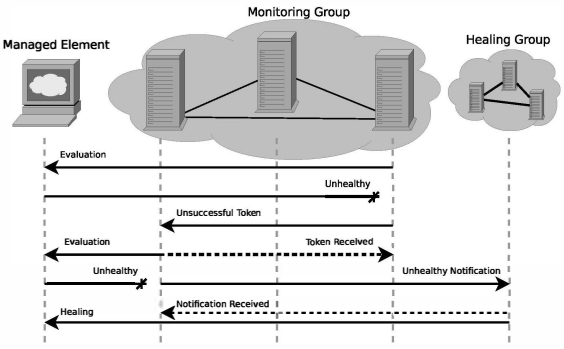

Fig. 4. Anomaly detection and notifications sending.

Since there is no strict rule about which peer of the healing group shall receive a unhealthy notification, may exist cases where a unaware peer of the healing group receives it. In this case, the unaware peer shall assume the responsibility for the notification and forwards it to the right peers.

After executing the healing workplan, the healing group notifies back the monitoring group. This notification shall contain information about the healing success or failure. This message determines if the monitoring group will resume the related monitoring workplan or will drop it. The former is done when the healing workplan succeeds and the system is once more in a healthy state, and the latter is done when the healing workplan can not bring back the managed element from unhealthy.

A special case for a healing notification is when a managed element is replaced by a newly configured one. In order to minimize the time to the self-healing service get reestablished, instead of dropping its monitoring workplan and waiting for a new service request, the monitoring group shall learn about the new element. To accomplish this, the healing group shall notifies the monitoring group as if the healing workplan was successfully executed, and append the information required by the monitoring group about how to monitor this element.

IV. Implementation

Our self-healing mechanism is built on top of a prototype P2P-based network management system. The implementation of this prototype, *ManP2P-ng*, has been done in the *Python* programming language using *Twisted*, an event-driven networking development framework [24] as its underlying network infrastructure. In the following subsections we detail this prototype design and implementation, as so we do for our self-healing mechanism.

A. ManP2P-ng management overlay

ManP2P-ng has embedded mechanisms for the self-organization and maintenance of a flat-topology P2P network, and flood-based resource discovery protocol. Beyond these, ManP2P-ng has not any other functionality, but provides a *Application Programming Interface* that may be used to extend the overlay. Indeed, the group abstraction and the self-healing mechanism are implemented throughout this API.

One of the key features of ManP2P-ng architecture is the notion of peer group, where a set of management components implementing the same management service are grouped together in order to improve service availability. Thus, peers that expose a specific management service are self-organized into a peer group (as described in Section III).

B. Application Programming Interface and Management Components

ManP2P-ng has an API that developers of management components must use in order to integrate their management components in a peer. This API has all the advantages of the Python language, including ease of development and platform independence. Besides, ManP2P-ng uses a standard event-driven networking development framework and exposes all of its features to management component developers.

The use of ManP2P-ng API allows instantiation, initialization, and operation of management components, which implement management services. Thus, a management component is managed through a well defined life cycle. This API also supports the use of P2P services provided by the underlying P2P system or by other management components.

Management components are defined in a *components descriptor.* This descriptor is an XML document that describes the management components to instantiate, the name of each management component, their interdependence, and the Python module to load and use. The descriptor is organized by the "<*management-component*>" XML element, where each element informs the init parameters used during initialization of the management component. This document is published to the P2P management overlay using the P2P Services.

Management components, and their associated management services, may vary from very simple ones (*e.g.*, a protocol gateway to access management devices via SSH or HTTP) to more complex ones (*e.g.*, support for Policy-Based Network Management - PBNM). These services have a unique service identifier (also known as group identifier). Moreover, a management component may extend the API, providing extra features to other components.

All Management Components that join a group have the same operations. These operations form the management services. The management services are then provided by the peers of the peer group. The multiple instances of the same

management component at different peers of a peer group provide fault tolerance features inherent to P2P systems.

C. Monitoring and Healing Workplans

We describe monitoring and healing workplans using *Ponder2* [25]. Ponder2 is a toolkit that supports the specification and enforcement of authorization and obligation policies. In workplans, only obligation policies are used. Obligation policies define actions that should be invoked in response to an event if specific conditions are met, and, are described in the form of Event-Condition-Action (ECA) rules.

Self-healing requires the combination of a monitoring workplan and a healing workplan. The use of workplans means they can be easily changed without recoding components. The Monitoring workplan is the first descritpion used for the self-healing procedure. As management data is required to verify the health of a device or system, the monitoring workplan maps the human knowledge of how to acquire this management data, and which events and system parameters indicates the healthy or unhealthy state of a device or system.

The Listing 1 shows an example of monitoring workplan. On the reception of a message from a device/service monitored, it is checked if the message payload presents a fault indicating content. If this condition is met (fault indicating content), the healing service is informed about this fault.

```
<add name="Monitor">
 <create type="obligation"
  event= "/event/DetectionServiceMessage"
  active="true">
  <arg name="fault"/>
  <condition>
   <eq>!content;<!-- -->fault</eq>
  </condition>
  <action>
   <inform "healingService"/>
  </action>
```

Listing 1. Monitoring workplan example

The Listing 2 shows an example of healing workplan. After fault detection, the healing workplan is used to correct the fault detected. In this example, the "healingActivities" script would be performed.

```
<add name="Healer">
 <create type="obligation"
  event= "/event/MonitoringServiceMessage"
  active="true">
   <arg name="monitor"/>
   <condition>
     <eq>!content;<!-- -->fault</eq>
   </condition>
   <action>
    <execute script="healingActivities"/>
   </action>
```

Listing 2. Healing workplan example

V. Case Study

The case study presented is based on the use of a self-healing mechanism to assist a Host-based Intrusion Detection System (HIDS). An HIDS monitors and analyzes hosts in order to determine whether they are being attacked or compromised. Ideally, an HIDS must employ a correlation engine able to detect patterns in misuse scenarios. Moreover, HIDS also must produce human-readable outputs so system administrator may diagnose threatening events and, when necessary, respond to them. However, IDS can be easily compromised by new or unknown attacks [26], or non-malicious faults.

Though commonly deployed HIDS use a manager-agent approach, HIDS slightly differ in the sense that, while standard approaches are based on data *pushes* from managers into the agents, HIDS uses a *pull* approach, where each intrusion detection agent periodically collects data about the host it runs on and sends this data to the manager. In the context of HIDS, agents are usually referred as *sensors*, and so they will be referred along this case study.

A. Failure Scenarios

From a macro vintage point, there are two scenarios for faults in an HIDS. The first scenario is when a sensor node fails. A well-designed HIDS must continue to function properly without the failing node. Is not essential, but is strongly recommended that administrators get informed about these faults. The second and most critical scenario is when the manager node fails. In this scenario, an HIDS must notify the administrator about the manager fault and graciously halt the sensor nodes to prevent misuses.

Due to the primary role that Information Technology (IT) security plays in nowadays communication networks infrastructures, both scenarios described are undesirable as they partially or completely stop the intrusion detection facility. During the HIDS downtime, in the first case, the host which the sensor fails may be lately compromised and maliciously used without knowledge by the system administrator. In the second scenario, the statements for the first scenario are also true, but in a more comprehensive scale (*e.g.*, an *0-day* worm-like exploit tool would indiscriminately spread itself).

B. Self-healing of HIDS Manager and its Sensors

The Self-healing of HIDS can be performed in different ways, and may involve different failures scenarios, reasons and recovery procedures. In this subsection, we discuss a common procedure to recover sensors and managers nodes.

The fault monitoring is traditionally performed through periodical polling by a network management system. When faults occur, human administrators have to manually perform the healing procedure. The traditional procedure to heal a HIDS manager or its sensors consists in i) remotely access the machine; ii) verify its latest entries in the system log files; iii) determine the cause of the failure or degradation; iv) readjust its parameters or develop a new set-up; and finally, v) put it online. In HIDS manager failure scenario, this is more critical, as some modifications needs to be propagated to all the sensors managed by the faulty manager.

The utilization of a P2P-Based Self-Healing Service brings advantages as the traditional procedures for monitoring and healing HIDS have concerns related to scalability and robustness. First, in a large HIDS system, would be infeasible for

human administrators to deal with a faulty manager that has a high number of sensor nodes associated with them. Second, the decisions related on how to deal with most faults, usually, do not involve complex analysis and action performing, thus, these faults would be easily healed though simple healing workplans. As minor faults are the most frequent, this would greatly reduce the demands for the attention of the system administrators. Finally, the repeatedly execution of the same tasks by human resources is proved as error prone.

VI. Evaluation

In this section, assuming the previous statements about self-healing of HIDS infrastructures, we present experimental measurements to verify the feasibility of our proposal. These experiments aim to measure the *total management traffic* generated throughout the self-healing process and its *average duration time*. The experiments considered two scenarios. In the first scenario, the healing workplan is completely executed by the peer that received a unhealthy notification from the monitoring peer group. The second scenario presents a cooperative healing workplan, where some actions involve the use of services provided by other peers of the management overlay. For simplicity, we refer to the former scenario as *independent healing workplan execution*, and the last as *cooperative healing workplan execution.*

In both experiments and evaluation scenarios, we consider a management overlay composed of 32 peers, evenly divided into monitoring and healing groups. Besides, in order to evaluate the relation of the size of the HIDS infrastructure and the parameters observed, we vary the number of sensor nodes in 1, 2, 4, and 8. For the second scenario, two peers of each group were also used to implement and provide a Remote Procedure Call (RPC) service used in the healing workplan.

In the first experiment, the total amount of management traffic generated during the healing process is measured. The objective of this experiment is to show the impact of the self-healing related traffic in the communication network demands as the number of managed elements grows. Figure 5 shows the results of this experiment.

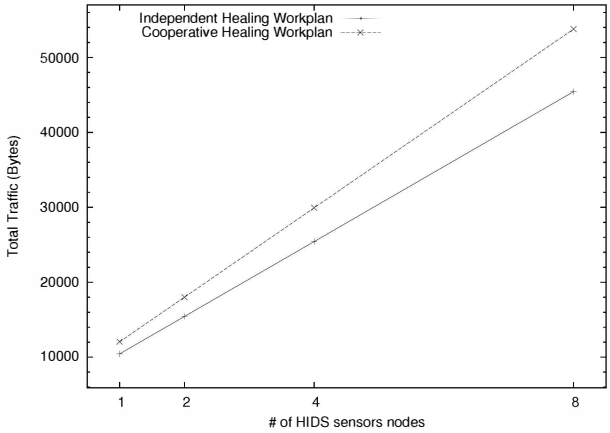

Fig. 5. Total management traffic during the Self-Healing process.

Figure 5 shows that, although essentially the same tasks are performed in both scenarios, as both workplans are functionally the same, the independent workplan execution require less network traffic than the cooperative workplan execution. During the independent workplan execution, messages are exchanged only between the peer who notices the fault (*i.e.*, a peer from the monitoring group) and the peer who performs the workplan. (*i.e.*, a peer from the healing group). On the other hand, when a collaborative execution is in place, in addition to the regular flow, synchronization messages must be exchanged between the peers of the healing group and the peers of the RPC group, so the result of the procedures calls may be collected and analyzed.

In the next experiment, the average duration time of the healing, considering both evaluation scenarios, is measured. The motivation of this experiment is to evaluate the impact of the cooperation among the overlay peers in the time needed to heal the system. The results are shown in Figure 6.

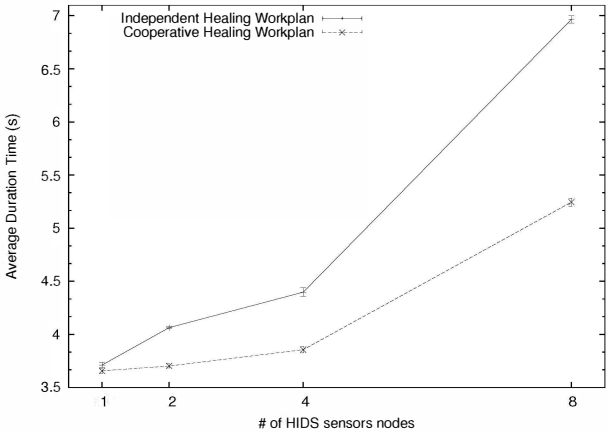

Fig. 6. Average duration time of the Self-Healing Process

Figure 6 shows that the cooperation among the peers of the management overlay beneficially impacts in the self-healing duration time. The independent workplan execution demands resources from a single peer of the healing group. Then, although some tasks run in parallel, the peer responsible for the workplan gets overloaded as the number of HIDS sensor grows. Collaborative workplan execution permits a more fine grained resource usage, allowing peers to share the healing workload. Thus, as the management overlay grows and its peers implements the services necessary to a specific self-healing process, the during time of this process would greatly reduce.

The results shown in Figures 5 and 6 make explicit the *trade-off* between the amount of traffic generated by the execution of a healing workplan, and the total time it takes. While an independent healing workplan execution is less resource demanding, a cooperative healing workplan execution is most suited for time-critical situations, as the workload it imposes might be shared among the peers of the management overlay.

VII. Conclusions and Future Work

The new trends of network management demands requires more sophisticated approaches, and so, is a key research issue in the network management area. Self-* features over distributed network management infrastructures, like P2P overlays, has been proving by the research community as feasible approaches for these demands. This is particularly true when improvements to the dependability of managed devices/systems are taken into consideration. However, distributed network management systems, in general, have only mechanisms to improve the availability of its constituent entities.

In this paper we have defined the basic building blocks required to develop self-healing mechanism over a P2PBNM system. One of the key features of the presented self-healing mechanism is the notion of *monitoring and healing workplans*, descriptions in a high-level language that capture the knowledge of the systems administrators of how the managed devices/systems shall be monitored and healed. We have also presented the design and implementation of *ManP2P-ng*, a prototype P2PBNM system, used as a basis for the instantiation of our self-healing service proposal. This service was implemented as a management component through the *Application Programing Interface* provide by *ManP2P-ng*, and is underlaid in its coordination and communication primitives. We have also presented an experimental evaluation of this proposal. In addition, we have described a case study using the self-healing of a distributed Host-based Intrusion Detection System (HIDS) to show the possibilities of our proposal.

Although the present proposal shows good results in evaluations performed until the present moment, it is necessary to evaluate more complex scenarios, in the number of healed elements and their heterogeneity, the number of peers in the management overlay, the services they expose, and high rates of churn. We are also looking at additional infrastructure settings that could lead to important effects, such as high intermittency in the connection between peers in the management overlay.

References

[1] J. Famaey, S. Latrea, J. Strassner, and F. De Turck, "A hierarchical approach to autonomic network management," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, 19-23 2010, pp. 225 –232.

[2] A. Pras, J. Schoenwaelder, M. Burgess, O. Festor, G. M. Perez, R. Stadler, and B. Stiller, "Key research challenges in network management," *IEEE communications magazine*, vol. 45, no. 10, pp. 104–110, October 2007.

[3] L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchiors, M. J. B. Almeida, and L. M. R. Tarouco, "Managing computer networks using peer-to-peer technologies," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 62–68, 2005.

[4] C. Melchiors, L. Z. Granville, and L. M. R. Tarouco, *Open Information Management: Applications of Interconnectivity and Collaboration*. Information Science Reference, 2009, ch. P2P-Based Management of Collaboration Communication Infrastructures.

[5] A. Panisson, D. M. da Rosa, C. Melchiors, L. Z. Granville, Maria, and Liane, "Designing the Architecture of P2P-Based Network Management Systems," in *ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications*. IEEE Computer Society, 2006, pp. 69–75.

[6] L. Fallon, D. Parker, M. Zach, M. Leitner, and S. Collins, "Self-forming Network Management Topologies in the Madeira Management System," *Lecture Notes in Computer Science*, vol. 4543, p. 61, 2007.

[7] A. Konstantinou and Y. Yemini, "A2A: An Architecture for Autonomic Management Coordination," in *Integrated Management of Systems, Services, Processes and People in It: 20th Ifip/Ieee International Workshop on Distributed Systems: Operations and Management, Dsom 2009, Venice, Italy, October 27-28, 2009, Proceedings*. Springer-Verlag New York Inc, 2009, p. 85.

[8] B. Jennings, S. Van Der Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, W. Donnelly, and J. Strassner, "Towards autonomic management of communications networks," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 112–121, 2007.

[9] D. Ghosh, R. Sharman, H. Raghav Rao, and S. Upadhyaya, "Self-healing systems - survey and synthesis," *Decis. Support Syst.*, vol. 42, no. 4, pp. 2164–2185, 2007.

[10] R. Sterritt, "Autonomic networks: engineering the self-healing property," *Eng. Appl. Artif. Intell.*, vol. 17, no. 7, pp. 727–739, 2004.

[11] A. Fox, E. Kiciman, and D. Patterson, "Combining statistical monitoring and predictable recovery for self-management," in *WOSS '04: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*. New York, NY, USA: ACM, 2004, pp. 49–53.

[12] D. Breitgand, M. Goldstein, E. Henis, O. Shehory, and Y. Weinsberg, "Panacea towards a self-healing development framework," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, may 2007, pp. 169 –178.

[13] Glassbox Documentation, "http://www.glassbox.com/."

[14] D. Garlan and B. Schmerl, "Model-based adaptation for self-healing systems," in *WOSS '02: Proceedings of the first workshop on Self-healing systems*. New York, NY, USA: ACM, 2002, pp. 27–32.

[15] Y. Liu, J. Zhang, and J. Strassner, "Model-driven adaptive self-healing for autonomic computing," in *MACE '08: Proceedings of the 3rd IEEE international workshop on Modelling Autonomic Communications Environments*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 62–73.

[16] Y. Liu, J. Zhang, M. Jiang, D. Raymer, and J. Strassner, "A model-based approach to adding autonomic capabilities to network fault management system," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, apr. 2008, pp. 859 –862.

[17] A. Binzenhofer, K. Tutschku, B. auf dem Grabem, M. Fiedler, and P. Carlsson, "A p2p-based framework for distributed network management," in *Proceedings. Wireless Systems and Network Architectures in Next Generation Internet*, ser. Lecture Notes in Computer Science, vol. 3883. Heidelberg, Springer-Berlim, 2006, pp. 198–210.

[18] A. Fiorese, P. Simões, and F. Boavida, "A P2P-Based Approach to Cross-Domain Network and Service Management," in *Proceedings of the 3rd International Conference on Autonomous Infrastructure, Management and Security: Scalability of Networks and Services*. Springer-Verlag, 2009, p. 182.

[19] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, L. Keoh, and A. Schaeffer-Filho, "Amuse: autonomic management of ubiquitous systems for e-health," *Journal of Concurrency and Computation: Practice and Experience, John Wiley*, 2007.

[20] S. M. Bourdenas T. and E. Lupu, *Architecting Dependable Systems (To be published)*. Springer LNCS, 2010, ch. Self-healing for Pervasive Computing Systems.

[21] C. C. Marquezan, C. R. P. dos Santos, J. C. Nobre, M. J. B. Almeida, L. M. R. Tarouco, and L. Z. Granville, "Self-managed services over a p2p-based network management overlay," in *Proceedings. 2nd Latin American Autonomic Computing Symposium (LAACS 2007)*, 2007.

[22] C. C. Marquezan, L. Z. Granville, G. Nunzi, and M. Brunner, "Distributed autonomic resource management for network virtualization," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, apr. 2010, pp. 463 –470.

[23] J. C. Nobre and L. Z. Granville, "Consistency maintenance of policy states in decentralized autonomic network management," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, apr. 2010, pp. 519 –526.

[24] A. Fettig, *Twisted network programming essentials*. O'Reilly Media, Inc., 2005.

[25] Ponder2 Documentation, "http://www.ponder2.net/."

[26] C. J. Fung, Q. Zhu, R. Boutaba, and T. Basar, "Bayesian decision aggregation in collaborative intrusion detection networks," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, apr. 2010, pp. 349 –356.