SINMS: A Slow Intelligence Network Manager based on SNMP Protocol

Francesco Colace[1], Shi-Kuo Chang[2] and Massimo De Santo[1]

[1]Department of Information and Electrical Engineering, University of Salerno, Italy
[2]Department of Computer Science, University of Pittsburgh, USA
{fcolace, desanto}@unisa.it, chang@cs.pitt.edu

Abstract:

Networks and distributed computing systems are becoming increasingly important and at the same time, more and more critical to the world of Information Technology. This rash spread, however, resulted in increased difficulty in configuring and managing networks. In particular the tasks of configuration management for IP network devices are becoming more and more difficult and the Simple Network Management Protocol is not able to manage very complex scenarios. Over the past years much efforts has been given to improve the lack of Simple Network Management Protocol and a promising approach involves the use of Ontology. Ontology is a very promising technology and can be effectively used in intelligent configuration management scope. This paper introduces a novel approach to the network management based on the use of the Slow Intelligence System methodologies and ontology. Slow Intelligence Systems is a general-purpose systems characterized by being able to improve performance over time through a process involving enumeration, propagation, adaptation, elimination and concentration. A Slow Intelligence System continuously learns, searches for new solutions and propagates and shares its experience with other peers. So the proposed approach aims to develop a system able to acquire, according to an SNMP standard, information from the various hosts that are in the managed networks and apply solutions in order to solve problems. To check the feasibility of this model first experimental results in a possible scenario are showed.

1. Introduction

Networks and distributed computing systems are becoming increasingly important and at the same time, more and more critical to the world of Information Technology. This rash spread, however, resulted in increased difficulty in configuring and managing networks. In fact there is an emergence of diverse network devices and it has become greatly difficult to configure those multifarious network devices with a manual work. The concept of network management is quite articulated. It involves activities such as the identification and management of various networks elements (hosts, gateways, routers, proxy ...), monitoring its performance, inventory of hardware and software available on a network and much more. So given this scenario, efficient and intelligent configuration management techniques are urgently needed to configure these devices with automation or semi-automation [1]. A solution for this problem can be the adoption of the Simple Network Management Protocol (SNMP). The SNMP manages network hosts such as workstations or servers, routers, bridges and hubs to a central computer that runs the network management software. SNMP performs management services through a distributed architecture of systems and management agents. Since network management is critical for both the control for managing resources, SNMP can be used to:

- Configure remote devices: the management system can send configuration information to each host on the network.
- Monitor network performance: you can track the processing speed, network speed and outcome data transmissions.
- Detect network failures or unauthorized access: the management system can detect the occurrence of certain events on specific network devices. Activating an alarm, the device management system sends a message indicating the event occurred. Common types of alarms are activated and start down a device when an error connecting to a router and in case of unauthorized access.
- Check the network: the management system can monitor both the entire network to identify users or groups that access, use and specific types of devices and network services.

Since its introduction in the late 1980s the SNMP showed good performance in monitoring for fault and performance, but it is very hard to use in managing large networks. In fact SNMP structure of management information (SMI) or Next Generation Structure of Management Information (SMIng) is insufficient to represent hierarchical network configuration data. SNMP, besides, supports with difficulties several high-level management operations required by network configuration tasks. Another problem is that SNMP is based on UDP and so bulk and reliable configuration data transfers are difficult to ensure. On the other hand as previously said the network management is a hot topic and there is a real interest in the development of an effective methodology. In literature ontology is considered a good way for supporting the network management and many papers deal with ontology based

methodologies for network management. In particular they propose ontology as a layer able to improve the interoperability among devices and operators. In this sense [2] proposes an ontology driven approach to the semantic interoperability problem in the management of enterprise service. Another interesting approach is in [3], which proposes an improvement of the current network management methods with the application of formal ontologies techniques. In particular it introduces a management information meta-model integrating all the information that currently belongs to different management model used to interoperate with the managed resource. Another advantage related to this approach is the ability to include basic semantic behavior for a manager to monitor and control these resources. Given this scenario this paper introduces a novel approach to the network management based on the use of the Slow Intelligence System methodologies [4] and ontology. The proposed approach aims to develop a system able to acquire, according to an SNMP standard, information from the various hosts that are in the managed networks and apply solutions in order to solve problems. In particular the proposed system can handle multiple networks and adopt solutions that have proved successful in some other context. By the use of ontologies the system will be able to choose the right action to take when some hosts send SNMP alerts. The use of the Slow Intelligence System approach will allow the system to automatically infer the actions to take. This paper is organized as follows. The next section introduces the slow intelligence systems approach. The second section describes the ontology while the third section describes the proposed system. The last section introduces the first experimental results.

1. What is a Slow Intelligence Systems

We will first introduce the concept of Slow Intelligence and present a general framework for designing and specifying Slow Intelligence Systems (SIS). We view Slow Intelligence Systems as general-purpose systems characterized by being able to improve performance over time through a process involving enumeration, propagation, adaptation, elimination and concentration [4]. A Slow Intelligence System continuously learns, searches for new solutions and propagates and shares its experience with other peers. A Slow Intelligence System differs from expert systems in that the learning is implicit and not always obvious. A Slow Intelligence System seems to be a slow learner because it analyzes the environmental changes and carefully and gradually absorbs that into its knowledge base while maintaining synergy with the environment. A slow intelligence system is a system that (i) solves problems by trying different solutions, (ii) is context-aware to adapt to different situations and to propagate knowledge, and (iii) may not perform well in the short run but continuously learns to improve its performance over time [4]. Slow Intelligence Systems typically exhibit the following characteristics:

Enumeration: In problem solving, different solutions are enumerated until the appropriate solution or solutions can be found.

Propagation: The system is aware of its environment and constantly exchanges information with the environment. Through this constant information exchange, one SIS may propagate information to other (logically or physically adjacent) SISs.

Adaptation: Solutions are enumerated and adapted to the environment. Sometimes adapted solutions are mutations that transcend enumerated solutions of the past.

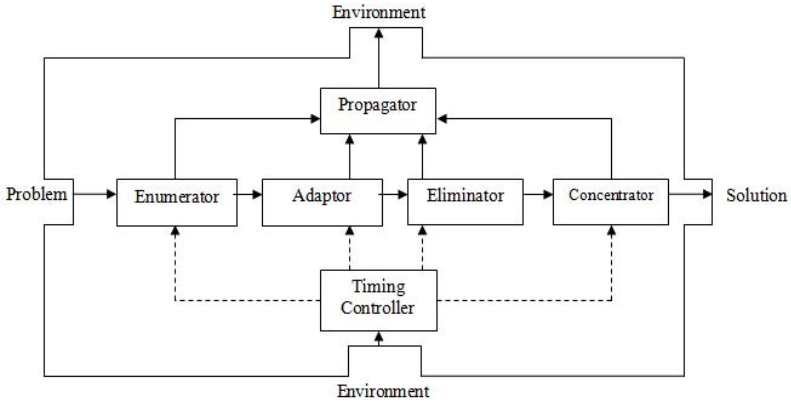

Figure 1 - The basic building block (BBB)

Elimination: Unsuitable solutions are eliminated, so that only suitable solutions are further considered.

Concentration: Among the suitable solutions left, resources are further concentrated to only one (or at most a few) of the suitable solutions.

The sixth one, on the other hand, is rather unique for SIS:

Slow decision cycle(s) to complement quick decision cycle(s): SIS possesses at least two decision cycles. The first one, defined as the quick decision cycle, provides an instantaneous response to the environment. The second one, defined as the slow decision cycle, tries to follow the gradual changes in the environment and analyze the information acquired by experts and past experiences. The two decision cycles enable the SIS to both cope with the environment and meet long-term goals. Sophisticated SIS may possess multiple slow decision cycles and multiple quick decision cycles. Most importantly, actions of slow decision cycle(s) may override actions of quick decision cycle(s), resulting in poorer performance in the short run but better performance in the long run. Now we can consider the structure of SIS by the introduction of the

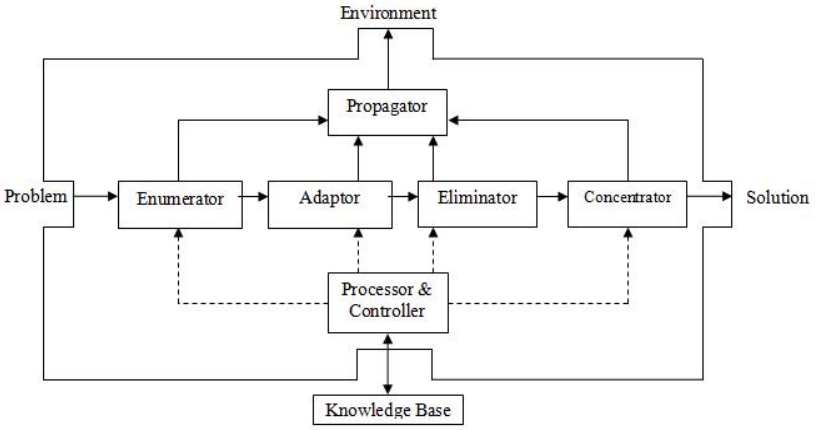

Figure 2 - The advanced building block (ABB)

basic building block and advanced building block. Figure 1 illustrates the Basic Building Block (BBB).

Problem and solution are both functions of time, thus we can represent the time function for problem as $x(t)_{problem}$, and the time function for solution as $y(t)_{solution}$. The timing controller is also a time function timing-control(t). For the two-decision-cycle SIS, the basic building block BBB can be expressed as follows:

```
if timing-control(t) == 'slow'
then /* timing-control(t) is 'slow' */
       y(t)solution = gconcentrate (geliminate (gadapt
(genumerate(x(t)problem))))
else  /* timing-control(t) is not 'slow' */
       y(t)solution = fconcentrate (feliminate (fadapt
(fenumerate(x(t)problem))))
```

where $g_{enumerate}$, g_{adapt}, $g_{eliminate}$, and $g_{concentrate}$ are the *transform functions* for enumeration, adaptation, elimination and concentration respectively during slow decision cycles, and $f_{enumerate}$, f_{adapt}, $f_{eliminate}$, and $f_{concentrate}$ are the transform functions for enumeration, adaptation, elimination and concentration respectively during quick decision cycles. Section 5 presents an example where some of these transform functions may be omitted during quick decision cycles.

An Advanced Building Block can be a stand-alone system as shown in Figure 2. The major difference between an ABB and a BBB is the inclusion of a knowledge base, further improving the SIS's problem solving abilities.

Since the transform functions of the ABB are influenced by the knowledge base, they are knowledge transforms. When the knowledge base is ontology, the transforms are ontological transforms. A particular ontological transform emphasized in our research is the ontological filter. Ontological filters can be used both as the Eliminator and as the Concentrator. The Propagator can also use ontological filtering to selectively send messages to other SISs. It is useful to underline how a slow intelligent system works at its best when works on lightweight plus ontology. This kind of ontology will be explained in details in the next section.

2. Why a Slow Intelligence System needs Ontology?

The definition of ontology is still a challenging task [5]. The term 'ontology' has its origin in the Greek word 'ontos', which means 'being'. So in this sense ontology could be defined as a branch of philosophy dealing with the order and structure of reality. In the 1970s ontology came to be of interest in the computer science field. In particular the artificial intelligence community started to use the concept in order to create a domain of knowledge and establish formal relationships among the items of knowledge in that domain for performing some processes of automated reasoning, especially as a means for establishing explicit formal vocabulary to be shared among applications. The term 'ontology' was first used in the computer science field by Gruber who used the term to refer to an explicit specification of a conceptualization [6]. The use of this term is rapidly growing due to the significant role it plays in information systems, semantic web and knowledge-based systems, where the term 'ontology' refers to "the representation of meaning of terms in vocabularies and the relationships between those terms" [7]. Also this kind of definition is still satisfactory for each field where ontology can be applied and so perhaps a good practical definition would be this: "an ontology is a method of representing items of knowledge (ideas, facts, things) in a way that defines the relationships and classification of concepts within a specified domain of knowledge" [5]. Following this point of view, ontologies are "content theories", since their principal contribution lies in identifying specific classes of objects and the relations that exist in some knowledge domains [8]. Ontologies can be classified into lightweight and heavyweight ontologies [9]. Lightweight ontologies include concepts, concept taxonomies, simple relationships between concepts (such as

specialization "is_a") and properties that describes concepts. Heavyweight ontologies add axioms and constraints to lightweight ontologies. Axioms and constraints clarify the intended meaning of the terms gathered in the ontology. Commonly ontology is defined as $O = \{C, A, H, R_T, R\}$ where:

- C is the concept set. $c \in C$ expresses one concept and in each ontology there is ever a root concept marked as "Thing". In particular for each $c \in C$ there exist a descendant nodes set (C_{DN}) containing all its under layer concepts and an ancestry nodes set (C_{AN}) containing all upper layer concepts
- A is the concept attributes set. For $c \in C$ its attributes set is expressed as $A_C = \{a_1, \ldots, a_n\}$ where n expresses the number of attributes related to *c*
- H expresses the concept hierarchy set. The formalism (c_i,c_j) means that c_i is the sub-concept of c_j. In other words this set contains the *is_a* relations among the classes.
- R_T is the set of semantic relations type. $R_T = R_{TD} \bigcup R_{TU}$. R_{TD} means the set of predefined relation (*same_as, disjoint_with, equivalent*) while R_{TU} means the set of user defined relation type. The formalism (c_i,c_j, r) with $r \in R_T$ means that between c_i and c_j there is the r relation. The set $RelR_T(c_i,c_j)$ contains the relation r between c_i and c_j
- R is the set of non-hierarchical relations. The formalism (c_i,c_j, r) with $r \in R$ means that between c_i and c_j there is the r relation. The set $Rel(c_i,c_j)$ contains the relation r between c_i and c_j

In the case of a Lightweight Ontology there are no R_T or R sets, therefore, the Lightweight Ontology is defined as: $O_L = \{C, A, H\}$. Therefore, the lightweight ontology is a very simple and basic representation of a knowledge domain and contains only a very general statement of a problem. The lightweight ontology can be viewed as a sort of taxonomy of the scenario and a first approach to the resolution of various users' queries. Obviously, it could be complicated by the insertion of new non-hierarchical relationships that enrich its semantic expressivity. In this case, a more mathematically rigorous representation is needed to provide greater confidence that the real meaning behind terms coming from different systems is the same. Heavyweight ontologies are extensively axiomatized and thus represent ontological commitment explicitly. Axioms help to exclude terminological and conceptual ambiguities due to unintended interpretations. Heavyweight ontologies can have a lightweight version. Many domain ontologies are heavyweight because they support heavy reasoning. Therefore, the heavyweight ontologies add axioms, well-formed formulas in a formal language, and constraints to lightweight ontologies in order to clarify the intended meaning of the terms gathered on the ontology. Therefore, a heavyweight ontology is defined by $O_H = \{C, A, H, R_T, R, A_X\}$ where A_X indicates the axioms that are in the ontology. Lightweight and heavyweight ontologies can not be considered as good solution for a Slow Intelligent System.

Lightweight ontologies propose a very simple view of the domain and do not allow a detailed description of the interactions among all the components in the domain. Using a lightweight ontology, users or systems can only share a very small and simple description of a domain that might not be useful for the resolution of a problem. On the other hand, using a heavyweight ontology, users can create and share very complex domains. However, it is very easy to have problems defining axioms as well as managing the ontology. To address these problems, a solution could be the introduction of a lightweight plus ontology defined as $O_{L+} = \{C, A, H, R_T, R\}$. By introducing non-hierarchical relations, a lightweight plus ontology is more complex and semantically richer than the lightweight ontology, but it is not complex as a heavyweight ontology because there are no axioms to consider. The starting point of this approach is the idea that it is reasonable to think that between the lightweight and the heavyweight ontology there are a series of ontologies that can be defined as lightweight-plus ontologies.

Some user requests can be supported by the use of one of these lightweight plus ontologies. Although the use of heavyweight ontologies guarantees the full execution of a task or a request, it does not assure the optimal solution. Each of these ontologies enriches its semantic level by the introduction of new classes, relations, functions, formal axioms and instances. These enrichments allow a better configuration of the system and the satisfaction of user requests. The lightweight plus ontology will be the starting point for the definition of a management network system.

3. A Slow Intelligence Network Manager based on SNMP Protocol

As previously said the aim of this paper is the introduction of a LAN-based management system based on SNMP protocol and the Slow Intelligence approach. The starting point is the general schema depicted in Figure 3. In this way, we have M different LANs to which may belong to N different types of hosts that have to be managed. Each of these LANs is dynamic and therefore allows the introduction of new hosts and the disappearance of some of them. The local servers are in principle able to solve the main problems in the LAN management, but thanks to the dynamism of the LANS may be faced with unexpected situations. The environmental conditions in which the LAN operates can influence the performance of various hosts and must be taken into account. In this scenario a fundamental role is played by ontologies.

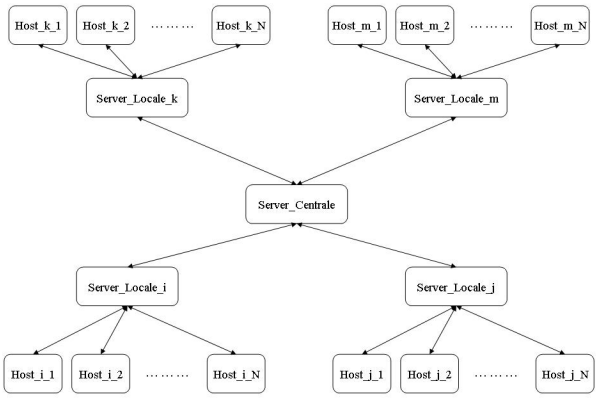

Figure 3: Network Management: A general Scenario

In particular it is necessary to introduce and define the following ontologies:

- O_{SNMP} = {C_{SNMP}, A_{SNMP}, H_{SNMP}, R_{TSNMP}, R_{SNMP}}. This ontology aims to define the entire structure of SNMP protocol by analyzing the various messages and the relations between them
- O_{Fault} = {C_{Fault}, A_{Fault}, H_{Fault}, R_{TFault}, R_{Fault}}. This ontology describes each kind of possible errors that can occur within a LAN
- O_{Cause} = {C_{Cause}, A_{Cause}, H_{Cause}, R_{TCause}, R_{Cause}}. This ontology defines the causes of the faults that may occur in a LAN
- $O_{Solution}$ = {$C_{Solution}$, $A_{Solution}$, $H_{Solution}$, $R_{TSolution}$, $R_{Solution}$}. This ontology defines the solutions that can be taken to recover from fault situations which occurred within a LAN
- O_{Action} = {C_{Action}, A_{Action}, H_{Action}, $R_{TAction}$, R_{Action}}. This ontology aims to identify the actions to be taken in order to recover from fault situations
- $O_{Component}$ = {$C_{Component}$, $A_{Component}$, $H_{Component}$, $R_{hComponent}$, R_{Action} }. This ontology describes the components that may be present within a LAN
- $O_{Environment}$ = {$C_{Environment}$, $A_{Environment}$, $H_{Environment}$, $R_{hEnvironment}$, $R_{Environment}$}. This ontology describes the environment where the LAN works

In order to allow the communication among the various hosts and servers that are in the various LAN the following messages have to be introduced:

M_{CSl}(SNMP, ID_Components) = this is the SNMP message that the client sends to the local server when an error has occurred. The ID_Componente used to identify the type of component that launched the message.

M_{SlC}({Action}) = this message, sent by local server, contains the actions that the client have to implement for the resolution of the highlighted fault.

The local server has to implement the following functions:

O'_{Fault} = f(M_{CSl}(SNMP), O'_{SNMP}) = this function aims to build the ontology of faults from the analysis of received SNMP messages and SNMP ontology within the local server. It is important to underline how the SNMP ontology on the local server is only a part of that present in the central server and is built from time to time following the faults that occur within the LAN.

O'_{Cause} = g(M_{CSl}(SNMP), O'_{SNMP}) = this function aims to obtain the ontology of the causes that generated the received SNMP messages.

$O'_{Solution}$ = h(O'_{Fault}, O'_{Cause}) = this function calculates the ontology of possible solutions that the local server can find for the solution of the fault situation

O'_{Action} = k($O'_{Solutiom}$) = this function calculates the ontology of possible solutions that the system can identify error to resolve the situation highlighted by the SNMP

These functions can be considered as the enumeration phase of the Slow Intelligent.

After the determination of these functions the system can adopt the Action to apply in the LAN by the use of the following function: {Action} = t(O'_{Action}, $O'_{Component}$, $O'_{Environment}$) = this is the set of actions that the client, or the host involved in the fault, must implement in order to solve the problem identified by the SNMP message. In practice, this involves defining, from ontologies of actions and components, the instances of actions to implement to resolve the faults that occurred.

This function implements the Adaptation, Elimination and Concentration phases of a Slow Intelligence System.

All these operations are carried out by involving the local server and hosts on the managed LAN. It is obviously the local server can not always perform operations that are asked, because it does not know the full SNMP ontology. In fact the managed LAN can change: for example new components can be added. So new messages, functions and actions have to be expected among local servers and central server. The messages are so defined

M_{CSlj}(SNMP, ID_Component) = this message contains the SNMP signal, sent by a host, that the local server is unable to manage and that it sends to the central network. The central server sends this message to the other local servers local in order to obtain information on the management of the SNMP signal.

M_{SljC}(O'_{SNMP-i}, $O'_{Cause-i}$, $O'_{Solution-i}$, $O'_{Action-i}$, {$Action_i$}) = this message contains the information obtained from local

servers about the SNMP signal management. downstream of question to which they have undergone. This message can be empty when no local servers ever managed in the past this kind of SNMP signal. Related to these messages there are the following functions:

$O'_{SNMP\text{-}i} = F(M_{SliSlj}(SNMP), {}_{SNMP\text{-}j})$ = this function expresses the subset of the SNMP ontology built in the local server j needed by the local server i.

$O'_{Cause\text{-}i} = G(O'_{SNMP\text{-}i})$ = this function expresses the ontology representing the causes of the fault. This ontology is built in the j-th local server and can be empty when this server never faced this problem.

$O'_{Solution\text{-}i} = H(O'_{Cause\text{-}i})$ = this function gives the ontology of the solutions that can be adopted in order to solve the fault related to the SNMP signal. This ontology is built in the j-th local server and can be empty when this server never faced this problem.

$O'_{Action\text{-}i} = K(O'_{Solution\text{-}i})$ = this function gives the ontology representing the actions that can be adopted for the solutions of the faults related to the SNMP signal. This ontology is built in the j-th local server and can be empty when this server never faced this problem.

The central server collects all the ontologies, obtained in the various local servers and previously described, and selects one of them according to an analysis based on ontology similarity. After this phase the central server can determine the action that have to be applied in the i-th LAN in order to solve the fault. So these actions can be sent to the i-th local server. In this way the following function can be introduced: $\{Action_i\} = T(O'_{Action\text{-}i}, O'_{Component\text{-}j})$ = this function calculates the set of actions that the client must adopt in order to solve the problem identified by the SNMP signal. The set of possible actions can of course be zero. In this case the support of an expert is needed. The previous messages and functions implement the propagation phase of the slow intelligence system approach.

The operational workflow is the following:

- Step 1: a SNMP messages generated by the Client as a result of a fault and sent to the local server
- Step 2: The local server receives the SNMP message and tries to identify the problem through analysis of various ontologies.
- Step 3 If the local server can identify the problem it generates the solutions and the actions that the various hosts in the LAN have to be apply.
- Step 3.1 The hosts get the actions and put them into practice
- Step 4 If the local server does not identify the problem sends the report to the central server.
- Step 5 The central server sends to all local servers received the message
- Step 5.1 Other local servers after receiving the message attempts to determine the possible actions and then send everything to the central server.
- Step 6 If the central server has received the possible actions by local servers then sends them to the local server that has requested it. If no action is received, however, the central server, based on the received message and its general ontologies determines the actions to be sent to the local server.
- Step 7, the local server send the actions to the various hosts that are in the LAN
- Step 7.1 The hosts get the actions and put them into practice

4. Experimental results

In order to test the performance of the proposed system an experimental campaign has been designed. First of all the working scenario has been settled (figure 4).

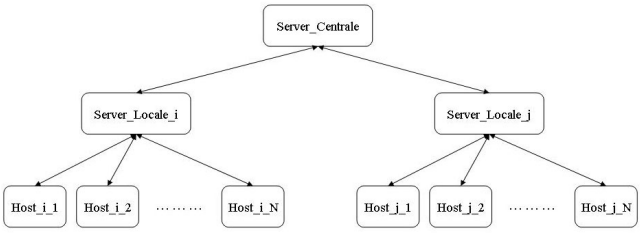

Figure 4: the operative scenario

In this scenario the central controller has to manage two different LANs. The first one is composed by a Cisco switch and 30 personal computers equipped with Microsoft as Operative System and Microsoft Office as applicative software. The second LAN is composed by a Nortel switch, 30 personal computers equipped with various operative systems (Microsoft Windows XP, Linux Red Hat and Apple SnowLeopard) and a HP network printer. Each local server has SNMP ontology able to cover the 80% of the SNMP messages that the hosts in the LAN can launch. The experimental phase aimed to evaluate the following system's parameters:

- The system's ability to identify the correct management actions to apply in the LAN after a SNMP signal. This parameter, named CA, is so defined:

$$CA = \frac{\#Correct\_Action}{\#Correct\_Action + \#Wrong\_Action}$$

- The system's ability to select in a LAN a viable solution that was previously adopted in a similar case in another LAN. This parameter, named IS, is so defined:

$$IS = \frac{\#Correct\_Inferred\_Action}{\#Correct\_Inferred\_Action + \#Wrong\_Inferred\_Action}$$

- The system's ability to manage the introduction of a new component in a LAN. In particular the system has to recognize components that were previously managed in other LANs. This parameter, named KC, is so defined:

$$KC = \frac{\#Correct\_Action\_NC}{\#Correct\_Action\_NC + \#Wrong\_Action\_NC}$$

The previous indexes were calculated in the following way:

- The CA index: this index was calculated after 10, 20, 30, 40 and 50 SNMP signals. In this case there was not variations in the LANs
- The IS index: this index was calculated forcing some SNMP events in the LAN not expected in its SNMP reference ontology. This index was evaluated after 10, 20, 30, 40, 50 SNMP signal not expected.
- The KC index was estimated after the introduction of new components in a LAN. In particular for five times a component belonging to a LAN has been shifted in the other LAN and the index was evaluated after 10, 20, 30, 40, 50 SNMP signal launched from the host.

In the next table the obtained results are showed:

Index	10	20	30	40	50
CA	90,00%	95,00%	93,33%	92,50%	92,00%
IS	50,00%	60,00%	66,67%	70,00%	74,00%
KC*	60,00%	70,00%	76,67%	80,00%	82,00%

Table 1: Obtained Results. The KC has to be considered as average value

The indexes show the good performances of the system. In particular the CA index, that expresses the ability of the system in the recognition of the correct actions in the LAN after a SNMP signal, is very good. The IS index witnesses how the system uses at the best the SIS approach. In fact the system improves its performances sharing knowledge among the various local servers. At the beginning the index is very low but it increases after few iterations. In addition, the KC index shows a similar trend.

Conclusions

In this paper a novel method for network management has been introduced. This method is based on, SNMP; Ontology and Slow Intelligence System approach. It has been tested in an operative scenario and the first experimental seems to be good. The future works aim to improve the system by the use of new and effective methodologies for the ontology management and the use of other network management approaches.

References

[1] Hui Xu, Debao Xiao, "A Common Ontology-based Intelligent Configuration Management Model for IP Network Devices", Proceedings of the First International Conference on Innovative Computing, Information and Control

[2] A. K. Yiu Wong, P. Ray, N. Parameswaran, J. Strassner, "Ontology Mapping for the Interoperability Problem in Network Management", IEEE Journal on Selected Areas in Communication, Vol. 23, No. 10, 2005

[3] Jorge E. López de Vergara, Antonio Guerrero, Víctor A. Villagrá, Julio Berrocal, "Ontology-Based Network Management: Study Cases and Lessons Learned", J. Network Syst. Manage. 17(3): 234-254, 2009

[4] Shi-Kuo Chang, "A General Framework for Slow Intelligence Systems", International Journal of Software Engineering and Knowledge Engineering, Volume 20, Number 1, February 2010, pp. 1-15.

[5] Jepsen, T., Just What Is an Ontology, Anyway?, IT Professional, vol. 11, no. 5, pp. 22-27, Sep./Oct. 2009.

[6] Gruber, T.R, Translation approach to portable ontology specification, Knowledge Acquisition, vol. 5, pp. 199-220, 1993.

[7]"OWL Web Ontology Overview", W3C Recommendation, 10 february 2004, http://www.w3.org/TR/2004/REC-owl-features-20040210/

[8] Maedche A., Staab S., Ontology Learning for the Semantic Web, IEEE Intelligent Systems, vol. 16 no. 2, Mar/Apr 2001, Page(s): 72-79.

[9] Corcho, O., A Layered Declarative Approach to Ontology Translation with Knowledge Preservation, Volume 116 Frontiers in Artificial Intelligence and Applications, 2005.