A Modular Agent Architecture for Optimising Hypothesis confirmation cost in Network Diagnosis

Álvaro Carrera and Carlos A. Iglesias

Universidad Politécnica de Madrid,
Madrid, Spain
a.carrera@dit.upm.es, cif@dit.upm.es

Abstract. This article proposes a MAS architecture for network diagnosis under uncertainty. Network diagnosis is divided into two inference processes: hypothesis generation and hypothesis confirmation. The first process is distributed among several agents based on a Multiply Sectioned Bayesian Network (MSBN), while the second one is carried out by agents using semantic reasoning. A diagnosis ontology has been defined in order to combine both inference processes.
To drive the deliberation process, dynamic data about the influence of observable variables (data) are taken during diagnosis process. In order to achieve quick and reliable diagnoses, this influence is used to choose the best action to perform. This approach has been evaluated in a P2P video streaming scenario. Computational and time improvements are highlight as conclusions.

Keywords: agent, Bayesian, ontology, diagnosis, network

1 Introduction

The complexity of telecommunication networks has increased the demand for network and service management systems. Nowadays, network fault management requires high skilled engineers, which are not able to cope with the increasing heterogeneity and complexity of the network. The probability of occurrence of faults in large telecommunication networks grows as they become widespread, complex and heterogeneous [3]. Thus, the role of automatic diagnosis modules is getting more attention, in order to cover the detection, isolation and recovery of faults. Another important aspect to point out is the need for dealing with uncertainty during the diagnosis task, since many corroboration tasks cannot be carried out because of different reasons, such as the cost itself of the action or that the action requires to access the subscriber equipment and could cause him any trouble.
The main focus of this paper is to present a Multi-Agent System (MAS) architecture that combines two reasoning processes: semantic reasoning

and Bayesian reasoning. This approach proposes to use Bayesian inference to handle uncertainty inherent in any diagnosis process and semantic inference to discriminate which action is the best one to perform depending on the available data.
The reminder of this article is structured as follows. Firstly, Sect. 2 proposes an agent architecture for reasoning during both phases of a diagnosis: hypothesis generation and hypothesis confirmation. These two phases can be deployed in different agents. Sect. 3 introduces a discussion comparing autonomy with resource consumption. Sect. 4 shows the knowledge model used in this work. Sect. 5 explains the case study where our approach is applied. Sect. 6 shows the evaluation and presents the results of comparison with other approaches. Finally, Sect. 7 draws out the main conclusions about the application of this approach and, besides, a brief description of future possible improvements.

2 Agent Architecture

This section proposes an agent architecture for diagnosis tasks which combines two reasoning processes. It consists of the following modules shown in Fig. 1.

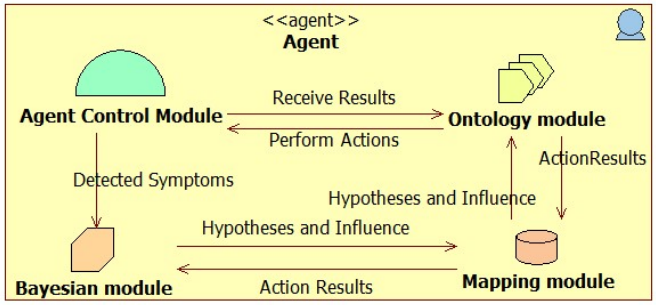

Fig. 1. Agent Architecture

Bayesian Module is a Bayesian reasoning inference engine that processes environment data (test results, symptoms, etc.) to infer possible root causes of the symptoms with an associated confidence about these beliefs. The outcomes of this module are hypotheses with its respective confidences and strength of influence among nodes [7].
We propose to use dynamic data from Bayesian networks. Each node of a Bayesian network has a concrete influence over its neighbours [7] in each moment of the diagnosis procedure. This influence between nodes is the strength of the dependencies between nodes that is quantified via conditional probability tables (CPT). This influence changes dynamically

depending on the evidences of the network (in other words, depending on the available information about the environment).
To obtain these data, CDF [8] distance is used. This method is suitable when there are ordinal nodes, because it represents the shift of probability according to the cumulative probability functions of the two distributions.

Ontology-based Reasoning Module has the aim of deliberating which action should be performed out of the information from the Bayesian module. Bayesian module generates an ordered list of possible actions to confirm the diagnosis hypothesis. This module filter this action based on the action preconditions. After executing the actions, the result is feedback to the Bayesian module. The reasoning process which includes rules for multimedia failure diagnosis are expressed with SWRL [9] and OWL [13].

Agent Control Module follows an extended BDI agent architecture where beliefs are distributed across the two above mentioned inference modules. When the agent receives new symptoms, a diagnosis plan is launched, combining the previous modules.

Mapping module translates information between *Bayesian module* and *Ontology module*. It performs the mapping process to create ontology individuals and extract information from ontology concepts to probabilistic data that can be input in the Bayesian module. To perform this task, we use PR-OWL [4] ontology that supports a way to add probabilistic information to others concepts defined using OWL.

3 Discussion: Agent Types

We have to remark that it is not necessary that all agents have all modules. Some functionalities can be distributed across several agents in order to obtain more scalability, remote access to restricted data, less computational requirements, etc.
At this point of the explanation and to clarify the multi-agent system proposed, three types of agents can be discriminated:

- *Fully Autonomous Agent* which has all modules presented before. It is able to evaluate the environment, reason (in a distributed way) under uncertainty, perform actions, etc. It can work autonomously, but it has better performance working together with other agents.
- *Semi-Autonomous Agent* which has Agent Control Module and Ontology based Reasoning Module. It cannot deal with uncertainty, but it is able to interact with its environment. To reason with uncertainty, it has to interact with an *Fully Autonomous Agent*.
- *Dependent Agent* which has only the Agent Control Module. It is able only to perform prefixed request actions. For example, the execution of one test or one monitoring action.

The usage of multi-agent technology for diagnosis tasks in telecommunication networks brings a range of benefits. For example, agents can be deployed in remote nodes, work when they are isolated or even create other agents dynamically when its functionality is required. These features are highly recommended for systems that work in complex environments. Our proposal consists of defining a flexible agent architecture which integrates the previously identified modules. These functionalities can be distributed at design time or even run time by the agents themselves (creating agents on demand), depending on non functional requirements (time to repair) or functional requirements (distribution requirements because of actions on remote equipments).

4 Knowledge-level model of the diagnosis task for telecommunication networks

Following the knowledge-level analysis of the diagnosis task by Benjamins [2], diagnosis can be decomposed into three subtasks: (i) *symptom detection*, finding out whether complaints are indeed symptoms, (ii) *hypothesis generation*, generate possible causes based on the symptoms, and (iii) *hypothesis discrimination*, discriminating between the hypotheses based on additional observations.
In this article, we focus on the last two tasks, hypothesis generation and discrimination, as well as in the repair task, as illustrated in Fig. 2.

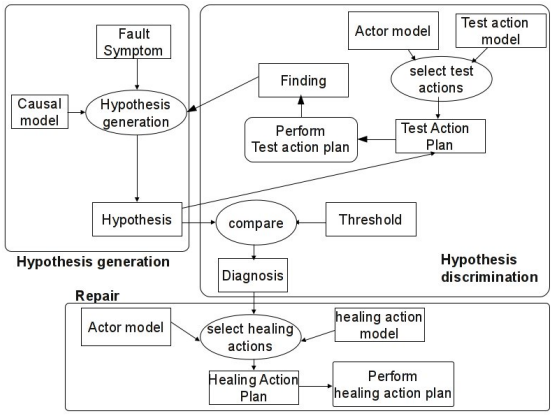

Fig. 2. Diagnosis inference structure. Legend: box (concept), oval (inference), rounded corner box (task)

The first process, *hypothesis generation*, consists of generating hypothesis from the notified fault based on a causal model. Since this process needs to handle uncertainty, a Bayesian network has been selected for expressing the causal model. Moreover, given that this Bayesian network

could not scale well with the size and heterogeneity of telecommunication networks, our architecture proposes the usage of Multiply Sectioned Bayesian Network [12], which allows to distribute this reasoning process in a multi-agent system.
The two other processes, *hypothesis discrimination* and *repair*, follow a similar pattern. The first one obtains a test action plan to confirm the generated hypotheses. This process contains a list of ordered actions to be executed based on the expected benefits of the tests. These benefits change dynamically with the influence between variables inside causal model [7]. This influence between variables is the strength of the dependencies between variables that is quantified via conditional probability tables (CPT). In this way, the system can perform more efficient hypothesis discrimination (see section 6).
Finally, the repair process obtains a healing action plan to repair the confirmed diagnosis. In order to reason under uncertainty, we propose to use an ontology based reasoning process, combining a diagnosis ontology expressed in OWL [13] with rules expressed in SWRL [9].
Nevertheless, a technique to communicate both reasoning processes is needed, in order to be able to provide feedback and integrate learning mechanisms of the confidence of they generated hypotheses based on the results of the tests.
Given the heterogeneity and complexity of systems in a telecommunication networks, this section defines a meta-model for diagnosis for facilitating information communication between the agents in the global network fault diagnosis task.

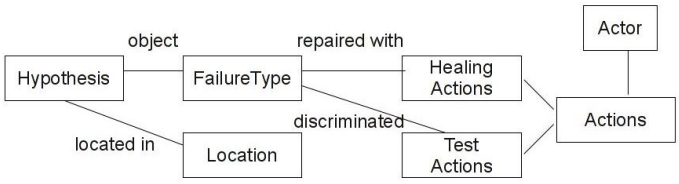

Fig. 3. Meta-model for diagnosis

This model (Fig. 3) shows that *hypotheses* are generated according to *failure classes*. These hypotheses identify a suspected *component* as the *location* of the failure. In this way, the model represents what is happening and where is happening. Depending on the hypothesis class, different *actions* can be carried out for corroborating the hypothesis (*test actions*) or repairing the component (*healing actions*). All *actions* have *conditions* (preconditions and postconditions) that allow somebody to evaluate its eligibility.
Another important concept is that *actions* are executed by *actors*. *Actors* can be *human* (manual actions) or *agents* (automatic actions).
Furthermore, the model also includes the concept of *diagnosis*. A *diagnosis* has its set of hypotheses, its set of performed tests and its set of

performed healing actions. This concept is useful for self-learning processes, and learning from past experiences, such as which healing actions repaired a certain failure.

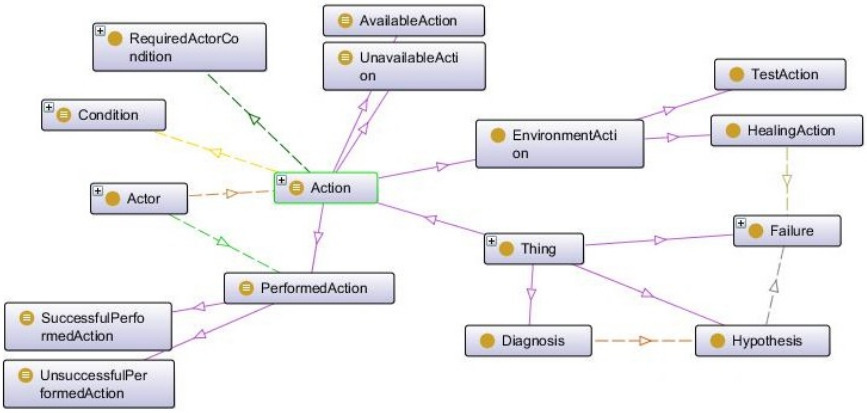

Fig. 4. Action meta-model

This model has been formalised as a domain ontology for reasoning on diagnostic tasks. This generic model can be extended for a particular domain, including the specific taxonomy of *faults*, *actions*, *actors* and *hypotheses*.
Actions can be classified according to the disjoint classes *Available/Unavailable*, when all preconditions are satisfied or not; and, if an *available action* has been performed, it is classified according to the disjoint classes *Successful Performed/Unsuccessful Performed*, when all postconditions are satisfied or not, as shown in Fig. 4.
The conditions of *an action* are modelled with the class *Condition*. Fig. 5 shows two generic conditions: *Required Data* that specifies a required parameter and *Required Actor* that specifies an actor to perform the action. The second one is common in all actions, because all actions need to be executed by someone. But these two actions are generic actions; all conditions that particularise specific restrictions should be added depending on the domain. *Conditions* can be classified according to the disjoint classes *Satisfied/Unsatisfied*. To check all conditions, we use SWRL. In section 5, the use of *conditions* are shown.

5 Case study

This section shows the case study used in this work. First of all, the scenario used to evaluate the model is presented in Sect. 5.1. The diagnosis process is shown in Sect. 5.2. And, finally, the system behaviour is explained step by step including SWRL rules examples in Sect. 5.2.

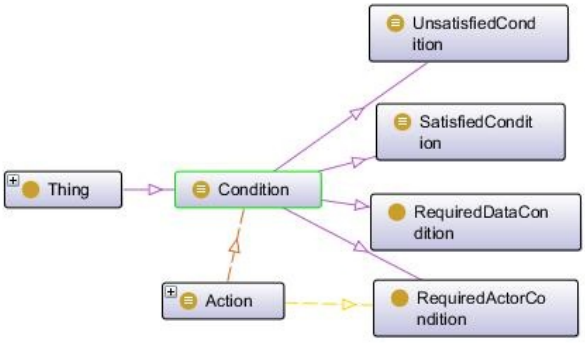

Fig. 5. Condition meta-model

5.1 Scenario

To properly frame this study, a P2P streaming scenario (see Fig. 6) was chosen. In this scenario, there are a multimedia provider user and a multimedia consumer user. Many faults may occur both in connection and in services. The system is designed to provide, to an end-user or an operator, the result of the diagnosis made upon receipt of a notification of a symptom of failure. The result is expressed in percentages representing the certainty of the occurrence of a given hypothesis.
The scenario network topology is as follows:

- Multimedia Provider Home Area Network that feeds the multimedia content.
- Multimedia Consumer Home Area Network that consumes the streaming service.
- ISP intranet that belongs to the service provider.
- Access network that provides access to home users.

Sharing of multimedia contents between two home users is addressed. These contents are stored in a video server inside of the Multimedia Provider HAN (Home Area Network) and are remotely accessed from the Multimedia Consumer HAN. Multimedia contents are transmitted in real time using RTSP (Real Time Streaming Protocol) for session establishment and RTP (Real-time Transport Protocol) for content delivery.

5.2 Streaming Diagnosis Case

In order to simplify the explanation of the proposed approach, only a simple case is exposed in the following paragraphs. First of all, the deployment of our multi-agent system is presented, and then, an overview of the Bayesian Network as well as the design principles are discussed. And finally, the key processes of the case are highlighted.

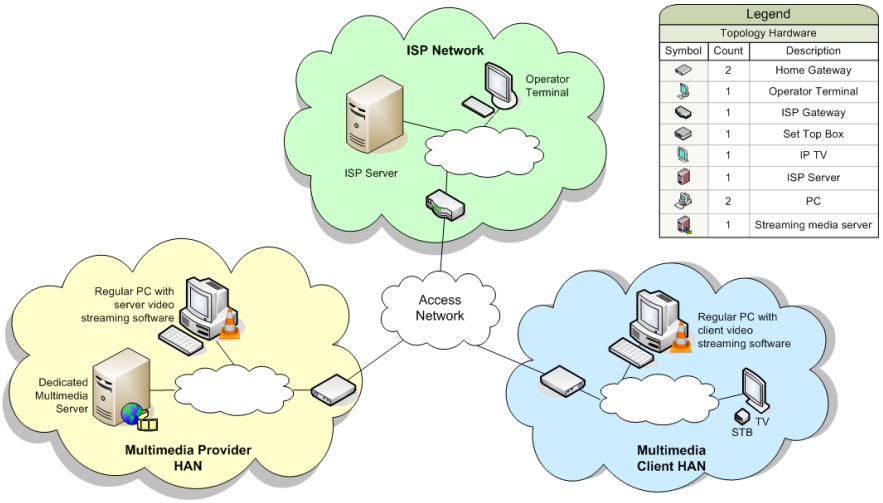

Fig. 6. Scenario topology

Agents have been deployed according to geographic distribution. Thus, one *Semi-Autonomous Agent* has been deployed into the multimedia client PC. This agent has monitoring capabilities. One *Fully Autonomous Agent* and one *Dependent Agent* have been deployed into Consumer Home Gateway. These agents have diagnosis and test capabilities. Two other agents like these (one *Fully Autonomous Agent* and one *Dependent Agent*) have been placed into ISP network. And finally, one agent of each type is deployed into Multimedia Provider HAN like into Multimedia Consumer HAN. In Multimedia Provider HAN, there are two agents into Home Gateway one *Fully Autonomous Agent* and one *Dependent Agent* and another one into Streaming Server (one *Semi-Autonomous Agent*). Each one of these agents know which actions are able to perform by itself and publish a list of these actions to allow other agents to request them.

Each *Fully Autonomous Agent* has its own piece of the whole MSBN, its own subnetwork. These Bayesian Networks have been modelled following the BN3M model [8]. In this model, three types of variables are distinguished: *context*, *fault* and *evidence*. *Context* variables model the environment, in this case, these variables are used to model information about the network in which each agent resides. *Fault* and *Evidence* variables are used to model the possible failures through hypotheses and observations. Fig. 8 shows a simplified subnet used by one *Fully Autonomous Agent* in Multimedia Provider HAN. To easy viewing of the subnet, *Context* nodes have been omitted. Only, *Fault* and *Evidence* nodes are shown.

Diagnosis Process In the following paragraphs, a diagnosis scenario is described based on the previously described configuration. Further-

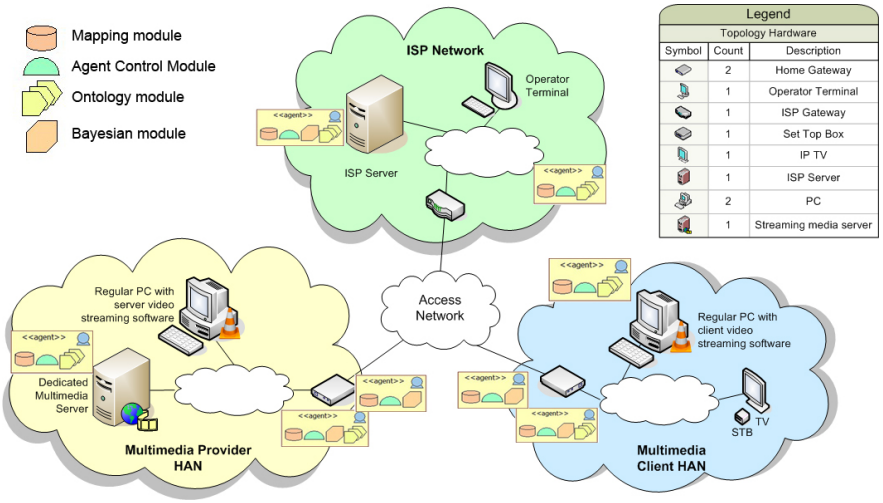

Fig. 7. Agent deployment

more, two simple examples of semantic reasoning used during the deliberation process are presented.

First, a streaming session is detected by the *Semi-Autonomous Agent* that resides inside multimedia client PC. This agent performs a monitoring action to know the quality of the session. If there is a quality degradation, a symptom is generated. However, this agent has not enough information to process this symptom, and it needs to cooperate with a *Fully Autonomous Agent* (in this case, the *Fully Autonomous Agent* that resides in the Multimedia Consumer Home Gateway).

This agent is able to process symptoms performing Bayesian inference in a distributed way (using MSBN approach). In other words, this agent shares information with others *Fully Autonomous Agent* that are able to reasoning with high level data. At this point, all *Fully Autonomous Agents* are working together and in parallel. Each one takes its own decisions using all available knowledge (shared knowledge and its own private knowledge).

But, once a symptom has been processed, which action is the best one to perform? Depending on the state of the environment and the knowledge base of the agent, one action could change its influence in the diagnosis process. To deal with this issue, we use the CDF method (see Section 2). With this method, all possible actions are ordered by relevance to reach a reliable confidence in the diagnosis process. The first one whose preconditions are fulfilled is selected and executed. (Note: This deliberation process is briefly described later in this article showing some used SWRL rules.)

Finally, when an hypothesis has a confidence higher than a threshold, the diagnosis finishes and a healing action is searched to fix the problem. Agents have been deployed in different devices to perform distribute diagnosis and the proposed meta-model has been applied (see Fig. 7). Some

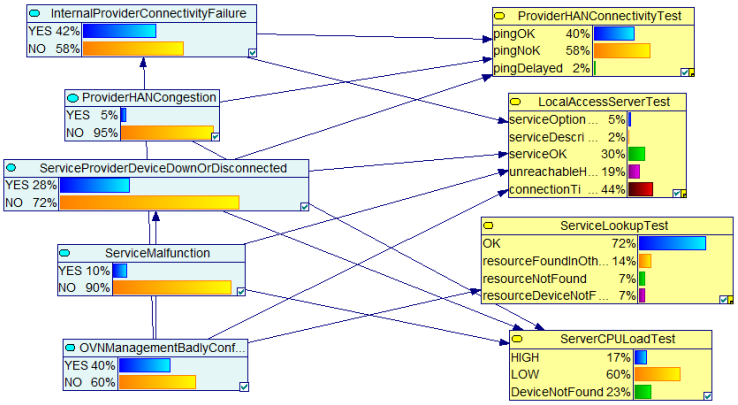

Fig. 8. Subnet of global Bayesian network used in the case study

agents have all modules proposed in section 2, these agents can work isolated without problems and offer more functionality to our diagnosis system. But, there are some devices such *dedicated multimedia server* that have low performance computational resources. On one hand, to solve this problem, we deploy modules in several agents to reduce required resources. But, on the other hand, an agent without Bayesian module cannot reach any diagnosis, because it only can perform actions, it does not perform inference. This inference process is performed in other agent deployed in a different device.

Semantic reasoning This section shows the behaviour of the proposed architecture in the case study scenario during one simple diagnosis. In sake of brevity, we only present two simple examples of deliberation process with SWRL rules: connectivity tests and CPU overload problem. As it is shown in section 4, the proposed ontology models general concepts like actor, action, condition, etc. But this generic terms have to be specialised for specific domains (in our case study, a streaming scenario). The first step to use the presented meta-model is introducing actors and actions (the most important concepts in our meta-model). Then, we should fill the relations between both entities: "actors can perform actions". The second step is introducing conditions about actions. Preconditions to know when an action is available and postconditions to know when a performed action was successfully performed or failed during its execution.
The following subsections show how the meta-model is used to choose an action during a diagnosis.

Case 1: Choose the best test to perform The generic meta-model is extended for this scenario with a specific ontology for diagnosis in multimedia. Generic *Test Action* class is specialised according to the test that can be carried out, such as *Connectivity Test* and new subclass of connec-

tivity test: *ClientToRouterConnectivityTestAction*. There is a *MultimediaClientDiagnosisAgent* actor that has *canPerform* property with range *ClientToRouterConnectivityAction*. Each of these test actions is specified with pre-conditions, according to the previously presented diagnosis ontology. For example, the action X has the preconditions Y and W. Below, several rules of the diagnosis module for selection test actions are presented.
Rule 1 presents a generic rule that determines if there is an actor capable of executing an concrete action. This condition is a*Required Actor Condition* and it is generic for all actions. If it is not satisfied, an agent can try to create and/or deploy new agents that are able to perform this action. The rule to know if this type of condition is satisfied is shown below.

Rule 1. Actor(?actor),
RequiredActorCondition(?condition),
hasPrecondition(?action, ?condition),
canPerform(?actor, ?action)
→ satisfied(?condition, true)

An exemplification of the use of Rule 1 is shown below:

Algorithm 1 Examplification of Rule 1

1: $Actor(?actor)$, {Return 3 actors: PingAgent, ServerAgent and StreamingAgent.}
2:
3: $RequiredActorCondition(?condition)$, {Return 10 contitions (each action has a RequiredActorCondition condition).}
4:
5: $hasPrecondition(?action, ?condition)$, {This query is used to obtain the action that has the condition (10 actions).}
6:
7: {Let's suppose that only one of the ten actions can be performed by the agents, for example, PingAgent can perform PingTest}
8:
9: $canPerform(?actor, ?action)$ {Return PingAgent actor and PingTest action.}
10:
11: $\rightarrow satisfied(?condition, true)$ {Then, RequiredActorCondition condition is satisfied, because there is at least one actor that can perform the action.}

Rule 2 presents a generic rule that determines if there is known parameter to execute an action. This condition is a *Required Data Condition*. This type of condition has a data property that represents which data is needed: *required Variable Type*. In this example, the value of this property is a string with content *Router LAN IP*.The required action to know this variable is specified as other action which has its own preconditions and can be performed by an actor. The rule to know if this type of condition is satisfied is shown in rule 2.

Rule 2. RequiredDataCondition(?condition),
Variable(?variable),
requiredVariableType(?condition, ?typecond),
variableType(?variable, ?typecond)
→ satisfied(?condition, true)

An exemplification of the use of Rule 2 is shown below:

Algorithm 2 Examplification of Rule 2

1: $RequiredDataCondition(?condition)$, {Return 3 contitions.}
2:
3: $Variable(?variable)$, {Return 3 variables (data about the environment: Router LAN IP, Streaming Server IP and Streaming Client.).}
4:
5: $requiredVariableType(?condition, ?typecond)$, {This query is used to obtain which variable is required to execute the action. In this example, Router LAN IP.}
6:
7: $variableType(?variable, ?typecond)$ {Return the variable which has *Router LAN IP* as type and *192.168.0.1* as value.}
8:
9: $\rightarrow satisfied(?condition, true)$ {Then, RequiredDataCondition condition is satisfied, because there is a variable with the required type.}

Finally, we have obtained an available action that can be performed by an actor. In this example, a test action to know how is the connectivity between the client PC and its router.
After this process, an agent has several actions that are available and is able to perform. Then, we need to know which is the best action to perform. To choose one, each test action has an expected benefit (data property). The value of this property is extracted from the Bayesian network (see section 2), depending on the influence of an action in a hypothesis.

6 Evaluation

The benefits of the proposed meta-model have been evaluated comparing this approach with previous works [10, 6]. In this paper, we compare the performance of the system using deliberation driven by "cost" or by "influence".
In previous works, test actions were classified by estimated cost. This cost combined time cost and computational cost and is estimated a priori by human experts. Then, all test actions are executed always in the same order. And sometimes, unneeded actions are executed.
The evaluation has been carried out based on a benchmark for a real diagnosis scenario of the R&D project Magneto. With data stored in database with old diagnoses and the same Bayesian networks have been

used in both cases. The volume of this data is around 500 diagnoses. We have clustered diagnoses in 13 diagnosis cases to simplify comparison and shown results.
As it is shown in Fig. 9, the number of performed tests has been reduced. Taking data from data base mentioned above, the average of performed test with deliberation driven by cost is 5.23 tests (with standard deviation 3.11). Using deliberation driven by influence, this number is reduced to 2.76 (with standard deviation 1.42); in other words, the number of performed tests has been reduced in 47.05%.
With deliberation driven by influence, there are two diagnosis cases that performs one test more than following the previous approach (driven by cost). The reason of this behaviour is that these are connectivity failures inside user HAN. These failures are very uncommon; for this reason, these hypotheses have, a priori, a little confidence and other hypotheses have to be confirmed or refused first.

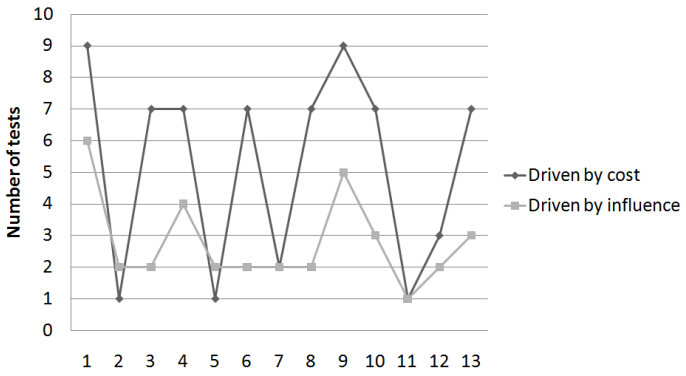

Fig. 9. Comparison: previous work vs proposed approach

Table 1 shows the evaluation results in several columns. MTTD [5] (Mean Time to Diagnose) usually is the average number of minutes until the root cause of the failure is correctly diagnosed; but, in this table, we show this time rounded in seconds. Other relevant times, like MTTR [5] (Mean Time to Respond) or MTTF [5] (Mean Time to Fix), are not covered in this study.
The column named "Result" represents if deliberation driven by influence improves driven by cost one or not in a specific diagnosis case.
The average of MTTD in previous approach is 25.47 seconds (with standard deviation 15.33), in proposed approach is 12.01 seconds (with standard deviation 7.12). Time improvement is 52.87%.

7 Conclusions and future work

We have presented a MAS that uses a meta-model ontology to diagnosis with Bayesian reasoning using OWL and SWRL to choose actions to

Table 1. MTTD and number of test comparison

Diagnosis case	**MTTD**		**Number of tests**		**Result**
	Cost	Influence	Cost	Influence	
Case 1	41	29	9	6	$\triangle$
Case 2	5	7	1	2	$\triangledown$
Case 3	36	8	7	2	$\triangle$
Case 4	39	15	7	4	$\triangle$
Case 5	4	8	1	2	$\triangledown$
Case 6	36	8	7	2	$\triangle$
Case 7	9	9	2	2	$\sim$
Case 8	34	8	7	2	$\triangle$
Case 9	40	24	9	5	$\triangle$
Case 10	33	13	7	3	$\triangle$
Case 11	5	5	1	1	$\sim$
Case 12	13	8	3	2	$\triangle$
Case 13	36	14	7	3	$\triangle$

perform. We focused on the deliberation process, leaving outside other research scope issues. Our proposal of decision support improves previous approaches [10, 6] both in time and in computational cost. Furthermore, the proposed modular architecture presents the capability to add or remove some modules of the architecture to reduce the resources required by agents, because there are important computational restrictions in devices like routers, TVs or STBs. And, one of the main advantages of use Bayesian Networks and semantic reasoning is the possibility of deploy new knowledge base or new reasoning rules on the fly, without restarting or deploying new agents.

As future work, we will study in depth the application of Multiply Sectioned Bayesian Networks (MSBN) [12, 11] to distribute the Bayesian inference engine that offers support to handle uncertainty and to maintain coherence and consistency in a distributed reasoning process. Applying MSBN approach, we can have a distributed inference engine that does local information processing, partial intermediate information exchange, inference global consistency and self-organization due to partial damage.

Acknowledgement

This research has been partly funded by the Spanish Ministry of Science and Innovation through the projects Ingenio Consolider2010 AT Agreement Technologies (CSD2007-0022), Cenit THOFU (CEN-20101019) and T2C2 (TIN2008-06739-C04-03/TSI) as well as the Spanish Ministry of Industry, Tourism and Trade through the project RESULTA (TSI-020301-2009-31).

The authors would also like to thank to J. García-Algarra, J. González-Ordás, P. Arozarena and R. Toribio for their support and collaboration in this research through the *Magneto R&D project* [1].

References

1. Arozarena, P., Toribio, R., Kielthy, J., Quinn, K., Zach, M.: Probabilistic Fault Diagnosis in the MAGNETO Autonomic Control Loop. In: 4th International Conference on Autonomous Infrastructure, Management and Security. pp. 102–105. Springer (Jun 2010)
2. Benjamins, R.: Problem-solving methods for diagnosis and their role. International Journal of Expert Systems: Research and Applications 8(2), 93–120 (1995)
3. Berthet, G., Fischer, N.: A unified theory of fault diagnosis and distributed fault management in communication networks. In: Proceedings of IEEE. AFRICON '96. pp. 776–781. IEEE (1995)
4. Costa, P.C.G., Laskey, K.B.: PR-OWL: A framework for probabilistic ontologies. In: Proceeding of the 2006 conference on Formal Ontology in Information Systems Proceedings of the Fourth International Conference FOIS 2006. pp. 237–249. IOS Press (2006), http://portal.acm.org/citation.cfm?id=1566079.1566107
5. FitzGerald, J., Dennis, A.: Business Data Communications and Networking. John Wiley and Sons (2008)
6. García-Algarra, F.J., Arozarena-Llopis, P., García-Gómez, S., Carrera-Barroso, A.: A lightweight approach to distributed network diagnosis under uncertainty. In: INCOS '09: Proceedings of the 2009 International Conference on Intelligent Networking and Collaborative Systems. pp. 74–80. IEEE Computer Society, Washington, DC, USA (2009)
7. Kjaerulff, U.B., Madsen, A.L.: Bayesian Networks and Influence Diagrams. Information Science and Statistics, Springer New York (2008)
8. Kraaijeveld, P., Druzdzel, M., Onisko, A., Wasyluk, H.: Genierate: An interactive generator of diagnostic bayesian network models. In: Proc. 16th Int. Workshop Principles Diagnosis. pp. 175–180. Citeseer (2005)
9. O'Connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., Musen, M.: Supporting rule system interoperability on the semantic web with SWRL. The Semantic Web - ISWC 2005 pp. 974–986 (2005)
10. Sedano-Frade, A., González-Ordás, J., Arozarena-Llopis, P., García-Gómez, S., Carrera-Barroso, A.: Distributed Bayesian Diagnosis for Telecommunication Networks, Advances in Intelligent and Soft Computing, vol. 70. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
11. Xiang, Y.: Belief updating in multiply sectioned Bayesian networks without repeated local propagations. International Journal of Approximate Reasoning 23(1), 1–21 (Jan 2000)
12. Xiang, Y., Poole, D., Beddoes, M.P.: MULTIPLY SECTIONED BAYESIAN NETWORKS AND JUNCTION FORESTS FOR LARGE KNOWLEDGE-BASED SYSTEMS. Computational Intelligence 9(2), 171–220 (May 1993)
13. Xiao Hang Wang, D.Q.Z.: Ontology based context modeling and reasoning using OWL. IEEE (Mar 2004)