Distributed Ontology-Based Monitoring on the IBBT WiLab.t Infrastructure

Stijn Verstichel, Eli De Poorter, Tim De Pauw, Pieter Becue, Bruno Volckaert, Filip De Turck, Ingrid Moerman, and Piet Demeester

Dept. of Information Technology - IBBT, Ghent University, Ghent, Belgium
G. Crommenlaan 8/201, 9050 Ghent, Belgium
Tel.: +32 9 331 4981; Fax: +32 9 331 4899
`<name>.<surname>@intec.UGent.be`
`http://www.ibcn.intec.UGent.be/`

Abstract. Testbeds as a means to evaluate protocol and software development are gaining importance, not least because of the oftentimes unpredictable influence of environmental behaviour. IBBT, the Interdisciplinary Institute for Broadband Technology, recognizes the importance of such testbeds and has therefore invested in WiLab.t, a wireless sensor and mesh testbed. It contains over 200 wireless and programmable nodes. The monitoring and management of such a testbed is very important so as to guarantee a proper functioning and stable environment to be used by researchers. This is however not a trivial task, even more so when in the future, the testbed is expanded with new devices and as such becomes a heterogeneous environment. Therefore, we have developed an ontology-based monitoring approach, which allows hiding the heterogeneity from the monitoring application and enables to process the data in a formal manner. Additionally, it allows adaptation according to characteristics of the local deployment, without the need to re-engineer the entire monitoring application every time alterations are made to the testbed.

Keywords: wsn, wmn, ontology, semantics, monitoring, reasoning.

1 Introduction

Whereas network simulators can be used to evaluate the performance of applications in general and for wireless sensor networks in particular, these do not have the capabilities to simulate all effects of real-life deployments, leading to a considerable discrepancy between experiments and simulations. Therefore, both wireless and other testbeds are a valuable tool for evaluating the performance of such applications.

IBBT, the Interdisciplinary Institute for Broadband Technology has recognized the importance of such testbeds. IBBT is an independent research institute founded by the Flemish government to stimulate ICT innovation. The IBBT team offers companies and organizations active support in research and development. It brings together companies, authorities, and non-profit organizations to join forces on research projects. Both technical and non-technical issues are addressed within each of these projects. Because of its belief in testbeds to enhance research and development, it has therefore invested heavily in the installation of a number of testbeds, to support a variety of research topics.

To enable intelligent monitoring and management of this testbed, an ontology-based monitoring framework has been developed. This framework has been deployed on the WiLab.t wireless sensor and mesh network testbed, but because of the adoption of a generic ontology approach, the framework could be used on other testbeds as well. Apart from offering monitoring information, the ontology-based approach also allows classification and inference to be performed on the monitored information. As such, only pre-processed and important information is exposed to the administrative team, avoiding the need to constantly manually analyze the data being produced by the monitoring application.

Although the monitoring use-case presented in this paper is very important on itself, it has supported and demonstrated the research and development of an ontology-based collaboration and data-aggregation platform, which can also be used in a wider context. Originally, the development of such a distributed ontology-based collaboration platform in a constrained environment was the main research task, but the monitoring use-case has proven very important and worthwhile to further exploit.

The remainder of this paper is structured as follows. The next section introduces related work on similar research topics. Section 3 describes in more detail the nodes and topology of the WiLab.t testbed. In Section 4 the complete architecture is introduced, starting from the software components on the sensor devices, through the mesh and back-end modules, concluding with the monitoring application. The platform has been thoroughly evaluated and the results are presented in Section 7. Finally, in Section 8 we present our main conclusions and introduce aspects for future research.

2 Related Work

The first generation of experimental set-ups' main purpose was the evaluation of nature monitoring applications. These set-ups did not have any advanced benchmarking facilities or have any flexibility regarding the reconfiguration of test set-ups [1]. To increase the reuse of existing testbed set-ups, newly developed testbeds offer more advanced management functions, such as automatic code deployment and scheduling mechanisms. These testbeds are deployed in a wide range of scenarios, from city monitoring [2] to office monitoring [3,4]. The number of active nodes in a single testbed ranges from a few nodes to more than 150 nodes. As of recently, efforts are being made to merge the different testing facilities into a single, world-wide testing environment [5].

However, many innovations are still missing in regards to the flexibility of wireless sensor testbeds. *(i)* Even though energy efficiency is very important for wireless sensor networks, very few testbeds have fine-grained and detailed power management and measurement capabilities. Similarly, the ability to emulate different battery types is still missing in most current testbeds. *(ii)* The topology of current deployments is fixed, resulting in inaccurate results when testing different sensor deployment scenarios. *(iii)* Accurate timestamp logging and time synchronization is frequently missing from the management platform. *(iv)* Sensor testbeds consist of at most a few hundred nodes, which is far less than the thousands of nodes which are foreseen in the vision of "the future internet of things" for office buildings. *(v)* Finally, the types of nodes deployed in a single testbed are very similar, whereas future sensor networks are foreseen to contain heterogeneous nodes with very diverse capabilities.

In this paper, we propose an ontology-based approach towards the monitoring of the WiLab.t testbed. A brief, but all-embracing definition of an ontology, can be found in [6]: "An ontology is a specification of a conceptualisation in the context of knowledge sharing." Accordingly, an ontology describes in a formal manner the concepts and relationships, existing in a particular system and using a machine-processable common vocabulary within a computerised system.

OWL, a modelling language for ontologies, consists of three sublanguages, each of them varying in their trade-off between expressiveness and inferential complexity. They are, in order of increasing expressiveness: *(i)* OWL Lite: supports classification hierarchies and simple constraint features, *(ii)* OWL DL: OWL Description Logics, a subset providing great expressiveness without losing computational completeness and decidability and *(iii)* OWL Full: supports maximum expressiveness and syntactic freedom, however without computational guarantees.

Using one of the three sublanguage flavours of OWL, one can easily adapt to the required expressiveness. Arguably the most interesting sublanguage for many application domains is OWL DL, balancing great expressiveness with inferential efficiency. The efficiency is guaranteed by the underlying Description Logics. Due to its foundation in Description Logics, OWL DL is also very flexible and computationally complete.

A number of initiatives were investigated previously to incorporate web semantic technology in wireless sensor environments. [7] presents a proposal that combines the benefits of autonomic and semantic sensor networks to build a semantic middleware for autonomic wireless sensor networks. Ontology-based data provisioning mechanisms for wireless sensor networks, in order to deal with varying applications, are presented in [8], while [9] defines a set of ontologies and accompanying architecture for knowledge sharing.

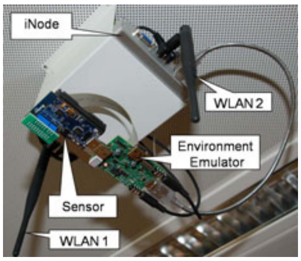
To conclude, even as there are currently a wide variety of testbeds available, many of these could be improved by providing more flexibility in regards to the configuration, power management, scale and topology of the testbeds. Also the adoption of ontologies and more specifically distributed reasoning mechanisms within the specific nature of wireless sensor and mesh networks to support reasoning in constrained environments is an additional feature presented in this research.

3 WiLab.t Infrastructure

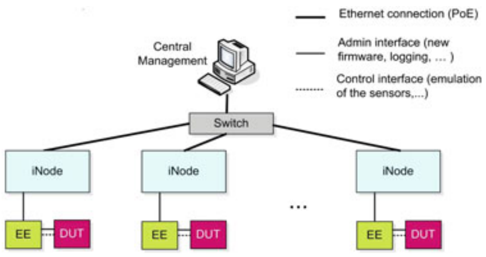
The WiLab.t test infrastructure is located at the IBBT office building of Ghent University, Belgium. This testbed consists of 200 TMoteSky [10] sensor nodes, spread over 3 floors. Wireless communication between the different floors is possible through 2 air shafts, in which several sensor nodes are installed.

Each sensor node is connected with an ‘Environment Emulator’ (EE) (see Figure 1(b)). This device can be used to control the physical properties of connected sensor nodes, *i.e.* sensor values can be emulated, the battery voltage can be regulated, general purpose pins can be connected with the sensor node and energy harvesting or battery models can be programmed. Finally, the EE also enables very accurate power measurements, in the order of microsecond intervals.

Finally, for ease of programming and debugging as well as to experiment with software in constrained environments, the wireless sensor nodes are connected with intermediate nodes, called iNodes, which are Alix 3C3 devices [11] running Linux Voyage [12]. Management of the testbed is performed using a modified version of the motelab [4] management software. This software is expanded with additional features such as a visualizer tool, a tool for graphical analysis of measured data and a customizable SQL database. A picture of a node in the WiLab.t testbed can be seen in Figure 1(a).



(a) A tesbed node consisting out of an Alix device, a TMoteSky sensor board and an Environment Emulator


Ethernet connection (PoE)
Admin interface (new firmware, logging, ...)
Control interface (emulation of the sensors,...)
Central Management
Switch
iNode
iNode
iNode
...
EE
DUT
EE
DUT
EE
DUT

(b) Component break-down diagram of the testbed nodes

Fig. 1. The nodes in the WiLab.t testbed infrastructure

4 Monitoring Architecture

This section introduces the architecture of the software components developed to monitor the behaviour of the WiLab.t infrastructure. The TinyOS NesC components to be run on the TMoteSky devices are described. Additionally, the more heavy-weight reasoning components, deployed on the iNodes, are presented. These components reason on the raw data being produced by the sensors. Finally, the overall workflow to trigger a monitoring task from the client, through the back-end and iNodes, finishing at the TMoteSky sensors is detailed.

Our general ontology-based monitoring approach has already been thoroughly evaluated in a back-end heavy-duty environment. This was published in [13]. Here, the ontology processing modules and mechanisms for query partitioning and execution have been detailed. However, because of the constrained environments taken into account in this scenario, we have had to define a number of additional modules and enhance certain mechanisms, in order to facilitate the deployment of the platform in this constrained environment. The extended platform architecture is given in Figure 2.

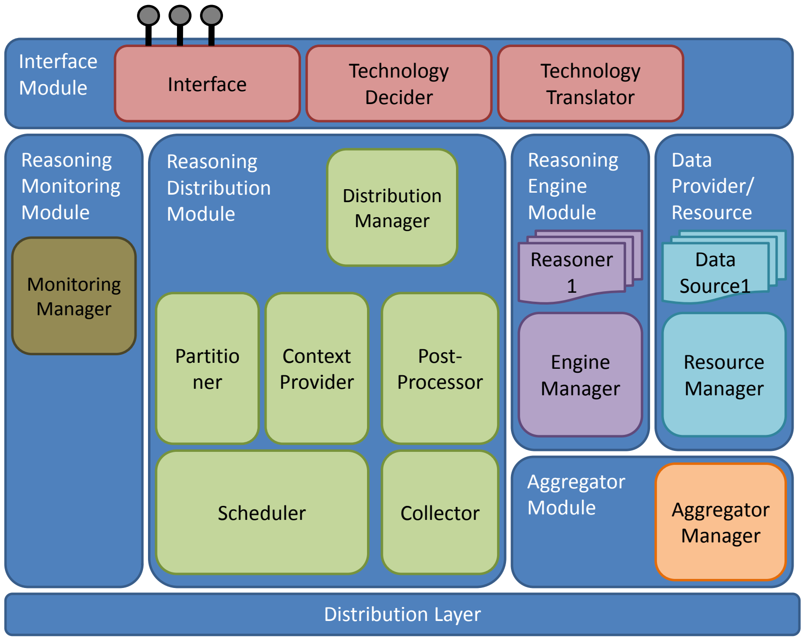


Fig. 2. Extended ontology-based agent collaboration platform

The main building blocks of the architecture are the *Reasoning Distribution Module* and *Reasoning Engine Module*. Additionally, to improve the transparency for the outside world, an extra indirection layer has been included at the interface level, namely the *Interface Module*. This layer is introduced to facilitate multiple reasoning technologies without the need for the clients to be aware of this. As such, only generic interface operations should be defined, avoiding the usage of reasoning technology dependent query and invocation mechanisms, *e.g.* SPARQL [14]. Additionally, to decouple the reasoning from the data storage, which was not the case in the original architecture presented in [13], two extra modules are introduced, namely the *Data Provider/Resource Module* and the *Aggregator Module*. The *Data Provider/Resource Module* will collect the data from the resources on which it has been deployed or for which it is responsible and feed it to the *Aggregator Module*. Upon request of the *Reasoning Engine Module*, the *Aggregator Module* will feed this collected data to the reasoning process. This way of working allows including sensor devices in the workflow and thus facilitates the monitoring of the sensor network, by means of sensor information generated from the *Data Provider/Resource Modules* deployed on the sensor devices.

4.1 The Data Components

Each node that contains an *Aggregator Module* needs to collect the appropriate data. To this end, each sensor node regularly gathers *system statistics*, such as internal voltage, node ID, total queue occupation, *network statistics*, such as number of bytes and packets sent, number of bytes and packets received, number of packet drops, number of failed transmissions and number of faulty packets received and *measurement data*, *e.g.* external temperature, internal temperature and humidity.

The information is sent at-runtime and wirelessly over an IEEE 802.15.4 wireless interface to the sensor node connected to the Alix board responsible for the reasoning on the data from that sensor node. This particular node is called the aggregator. It forwards the incoming data immediately to the back-end database. Since we want to include information about the networking layer in the reasoning, we need to set up a wireless sensor cloud with sufficient network traffic. Therefore we do not use every local iNode with its WiFi connection to transmit its information, but use the senor nodes for this matter. However, due to the large scale of typical sensor networks, measured information cannot be sent directly to the aggregator node. Instead, sensor nodes use a multi-hop approach whereby measured information is sent over intermediate nodes to reach the *Aggregator Module*. Each *Aggregator Module* contains a software component responsible for notifying nearby sensor nodes. Therefore, a software component is installed that regularly broadcasts “sink” notification messages [15]. Each sensor node that receives a sink message checks if the hop count is lower than any previously received sink message. If it is, the notification is further forwarded by the node, and the address of the neighbour from which the sink message was received is used as the default next hop address when forwarding measured data. This way, each sensor node sends its information from neighbour to neighbour until the information reaches the nearest aggregator node. Thus, to collect the appropriate data, the following software components are installed on each sensor node:

SensorMeasurement. This component regularly gathers system, network and measures sensor data from the appropriate data components. All information is encapsulated in a packet and is sent to the *DataDistribution* component.

DataDistribution. This component is responsible for selecting the next hop neighbour to which packets are forwarded. As part of this component, the sink with lowest distance is selected, and unreliable routes are regularly purged.

4.2 The Reasoning Components

As indicated earlier in Section 3, every node in the WiLab.t infrastructure, consists out of two devices. One of the devices is a TMoteSky [10] sensor, the other is an Alix [11] board. This allows for a hybrid approach concerning the deployment of software components. More specifically for components having different functionalities, requiring different specifications. The software components on the sensors producing the data in a non-intrusive manner, were detailed in Subsection 4.1. However, as detailed in Section 1, an intelligent monitoring framework, based on distributed formal first-order logic reasoning using ontologies, is pursued. This goal resulted in a number of reasoning components being developed and deployed in the WiLab.t testbed. The reasoning modules impose more stringent requirements on the hardware running these components. Using the iNodes, reasoning components can be deployed to process the data in a more intelligent way. These reasoning modules, using standard reasoning software, such as Pellet [17], analyze the data using the ontology model to draw conclusions about the status of the nodes in the testbed.

4.3 Ontology Used for WiLab.t Monitoring

Starting from the Sensor Node Ontology [16], which describes various states of a sensor node depending upon states of its constituent modules, additions and enhancements were modelled to take the specific situation of the WiLab.t into account. An important addition to the ontology is the location information. In this way, we can model the physical location of the nodes in the ontology. Additionally, a further component breakdown of the sensor nodes was modelled. To facilitate this, the concepts *SensorBoard* and *SensorPart* have been introduced. The general goal of the ontology is to classify the sensor nodes based on the values of the monitored metrics. Therefore, we used a typical observation pattern.

The two most important concepts in the ontology in terms of reasoning are *Fault* and *Solution*. A *Fault* subconcept is defined based on a logical statement

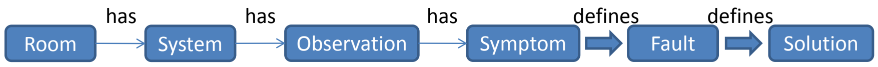


Fig. 3. Main property chain

Table 1. Additional properties and their characteristics in the WiLab.t Monitoring Ontology

Object Property Name	Characteristics	Domain	Range	Inverse Property
hasObservation		System	Observation	isObservationOf
hasSolution		Fault	Solution	isSolutionForFault
hasSensorPart		System	System	isSensorPartOf
hasSymptom		Observation	Symptom	isSymptomOf
hasNextObservation	Functional	Observation	Observation	hasPrevObservation
requiresAction		Solution	Action	isActionFor
hasFault		Symptom	Fault	isFaultOf
hasSystem	Inv. Functional	Room	System	isLocatedIn
hasRoom	Inv. Functional	Floor	Room	isOn

mainly combining *Symptoms*. In turn a *Solution* is defined mainly using a combination of *Faults*. Example definitions of two *Fault* concepts can be found below.

Detected TMoteSkyFault definition

```
[hasSensorPart some
  (hasObservation some
    (hasSymptom some HumidityZeroSymptom)) and
 hasSensorPart some
  (hasObservation some
    (hasSymptom some LightIntensityZeroSymptom)) and
 hasSensorPart some
  (hasObservation some
    (hasSymptom some TemperatureZeroSymptom))] or
[hasObservation some
  (hasSymptom some MissingReportsSymptom)]
```

This definition specifies that a TMoteSky sensor node which has a sensor part that outputs at the same time zero as value for temperature, humidity and light intensity or which does not produce anything is to be classified as faulty.

Incipient HVACFault definition

```
hasSystem some
  (hasSensorPart some
    (hasObservation some
          ((hasSymptom some
            (TemperatureBelow15Symptom or
TemperatureAbove25Symptom))
and
(hasSymptom some
  TemperatureNotZeroSymptom))))
```

In this definition, a room which has a system - *i.e.* a TMoteSky - which in its turn has a sensor part that outputs a non-zero temperature value below 15 °C or above 25 °C, probably has a faulty HVAC system.

4.4 The Coordinating Back-End Component

In the context of the WiLab.t infrastructure monitoring application, two typical queries are used. The first type queries for the inferred *Fault* individuals, while the second type triggers the reasoner to infer the possible *Solution* individuals for a given *nodeID*. The information contained in an ontology is modelled in a triple type format. This means that a subject is linked to an object by means of a predicate. This mechanism is used by the SPARQL query language to specify queries. By inserting unbound variables in the triple patterns, the reasoner will search the model and its data to find the individuals satisfying the triple. For more information we refer to [14].

***Incipient fault* query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX wsn: <http://users.atlantis.ugent.be/svrstich/deus/wsn#>
SELECT ?x0 WHERE {
    ?x0 rdf:type wsn:IncipientFault .
    ?x0 rdf:type wsn:System
}
```

This first query searches for all individuals which belong at the same time to the *IncipientFault* and *System* concept. As described in the previous section, the *IncipientFault* concept is modelled by means of a description logic statement. As such the reasoner will check at-runtime which of the *System* individuals, i.e. the sensors, satisfies this logic statement and will only return those sensors that do match this description.

***HVAC* query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX wsn: <http://users.atlantis.ugent.be/svrstich/deus/wsn#>
SELECT ?x0 WHERE {
    ?x0 rdf:type wsn:PossibleHeatingFault .
}
```

Dually to the previous query, this query triggers the reasoner to search through the entire set of data to find those objects which satisfy the logic description of a *PossibleHeatingFault*. Its definition is presented in Section 4.3.

***Solution* query**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX wsn: <http://users.atlantis.ugent.be/svrstich/deus/wsn#>
SELECT ?x0 ?x1 WHERE {
(1) ?x0 rdf:type wsn:DetectedFault .
(2) ?x0 wsn:hasID \"24\"^^xsd:integer .
(3) ?x0 rdf:type ?x1 .
(4) ?x1 rdfs:subClassOf wsn:Solution
}
```

The line indicated with (1) in this last query can be replaced with an appropriate *Fault* concept from the ontology. The *nodeID* mentioned as object in pattern (2) might obviously be replaced with the id of the node concerned in the query.

Additionally, for that sensor node, the reasoner is asked to infer which other types can be linked to the individual representing the faulty sensor node. This is defined by line (3). However, by also including line (4) in the query, we limit the search for other types, to those types that are modelled in the ontology as a subclass of *Solution*. After all, the goal of this query is to find the solution for a given node with a certain fault.

5 Deployment Overview

This section demonstrates that the architecture defined in the previous sections can be deployed on the different nodes of heterogeneous networks, such as the WiLab.t infrastructure, facilitating a distributed monitoring platform which can be tuned to the needs of each individual deployment. Although only a single type of sensors is currently deployed on the testbed, future extensions with different types of hardware are planned. After the integration of this new hardware, the addition of new ontology models and data providers will suffice to include them in the monitoring workflow. After all, the specific definitions of the concepts against which the observations are checked to realise the correct *Fault* and *Solution* classification, can be changed independently from the end monitoring application.

An example of this claim is the detection of faulty HVAC (Heating, Ventilation and AirConditioning). In a normal office environment, an upper threshold of 25 °C can already indicate an HVAC problem, while in a lab environment this threshold could easily be 35 °C or 40 °C. To support this kind of adaptation, the ontology T-Box which is deployed on the reasoning agent can be altered according to the needs of this particular situation. The other parts of the platform do not need to know about this, because the communication will only involve the request for rooms in which the HVAC system might be corrupted. The reasoner will use the locally deployed definition to check the local data. Figure 4 presents this deployment in a graphical manner.

The monitoring of the sensor network within the WiLab.t infrastructure has been defined in an office environment. In this setting, every office has a number of deployed sensors, working together in a wireless sensor network. Each of the offices is networked together by means of a light-weight dedicated access point, establishing a mesh network for communication between the offices. This mesh network in its turn is supported by a back-end network to facilitate more services, *e.g.* an uplink to the internet. Starting from the data generated at the source of a sensor node, we deployed the *Data Provider/Resources Modules* on the TMoteSky sensor nodes in the sensor network. As described, these modules provide the data to be included in the reasoner for the monitoring process. It makes this information available to the *Aggregator Module*, which stores it in a MySQL database. We envisage deploying such a database either in the backend network, or ideally on a mesh node. In this situation, the mesh node can handle locally the data coming from the sensor nodes attached to it. Therefore we included the *Aggregator Module* on the mesh node as well. Additionally, the

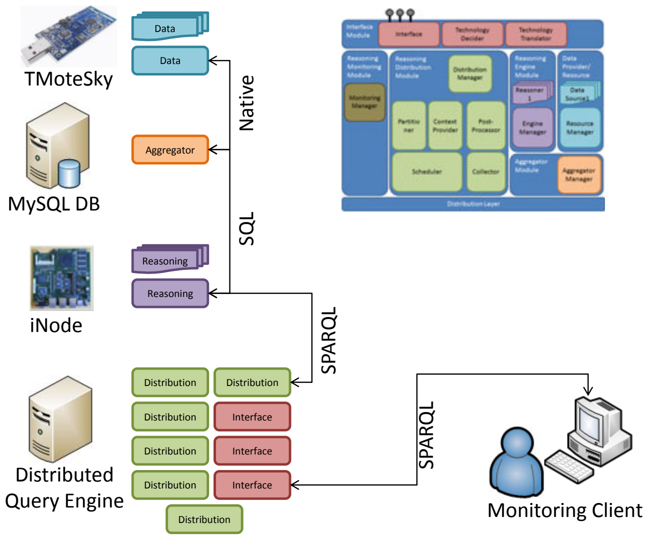


Fig. 4. Deployment view of the platform modules on the devices of a DEUS network

Reasoning Engine Module was deployed on the same mesh node. After all, since there is a mesh node for every office in which a number of sensors have been deployed, this mesh node is the ideal place to locally process and reason on the information coming from the locally deployed sensor nodes. As indicated, the description of the local situation in the ontology T-Box, the model, included in the reasoning process on this mesh node, can be tuned to the particular needs. Using D2R [18], an automatic conversion between raw data in the MySQL database and the ontology A-Box, the data, is supported. An example mapping for a certain *Symptom* concept, namely the observation that no information is being generated by the nodes, can be found below. By using a left-join SQL statement, even the absence of observations can be represented in specific A-Box individuals. For a detailed description of the constructs used in this mapping language, we refer to [18].

```
map:TMoteNoObservationSymptoms a d2rq:ClassMap;
    d2rq:uriPattern "NoObservationSymptoms/@@coordinates.id@@";
    d2rq:class vocab:MissingReportsSymptom;
    d2rq:classDefinitionLabel "MissingReportsSymptom";
    d2rq:condition "coordinates.id NOT IN
        (SELECT DISTINCT sensorinfo.moteid FROM sensorinfo)";
```

All other remaining modules are deployed on a back-end server, which can be used by monitoring clients to trigger the monitoring process.

The adoption of ontologies has not only been driven by the logical formalisms defining the constructs in such models, but also by their ability to be used for communication. After all, an ontology can easily by serialized in a number of specific and well-defined languages, such as RDF/XML, n-triple, *etc.* Their communication nature has always been considered as one of the main strongholds of ontologies. Not surprisingly so, given their origin in the Semantic Web. This means that not only the data is serialized and/or transmitted, but also the meaning of this data. This turns it into machine-processable information. The architecture described earlier takes full advantage of this.

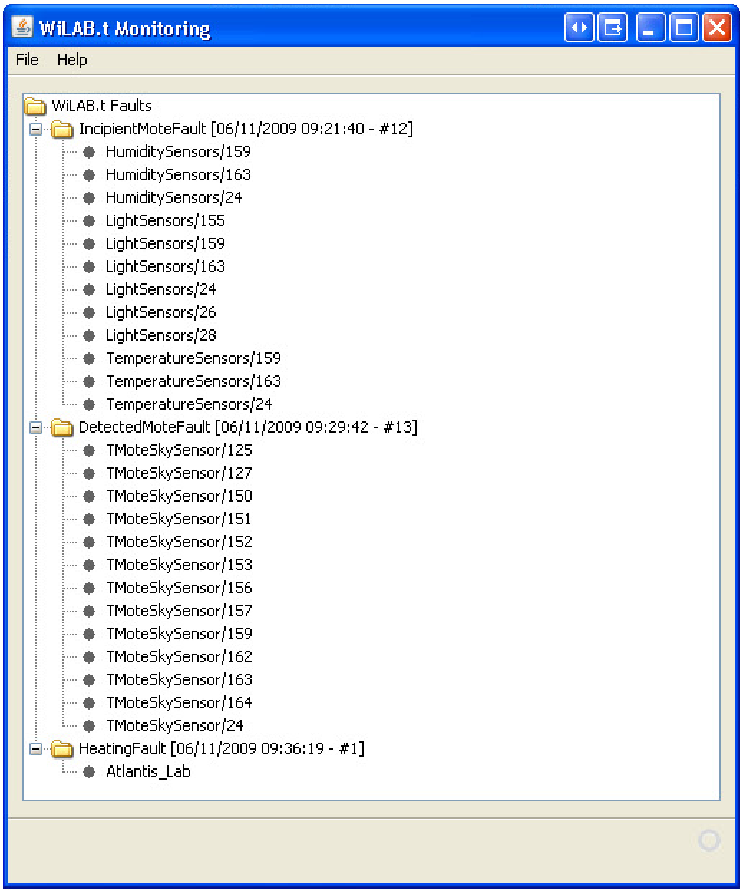
All high-level communication is performed using the SPARQL [14] query language. As such, only certain parts and certain views of the ontology are transferred. Additionally, only one interface operation is necessary, no matter what the content of the ontology T-Boxes is. The argument of this operation is a SPARQL query, and according to the T-Box of the ontology deployed, other SPARQL queries can be used. However, by making use of logically defined concepts, even these queries themselves won't have to change all that drastically. Of course, if a complete new ontology is used in a given deployment, serving a completely different use case, this argument does not hold. But for similar use cases, a specific alteration of the definition of the logically defined concept should be sufficient to handle the different scenarios. Additionally, by introducing the *Interface Module*, even the usage of SPARQL as implementation within the reasoning platform is transparent to the monitoring client. The interfaces and protocols used in the low-level part of the platform, namely the sensors, are specifically implemented to be used on those devices and for the information to be exchanged.

6 The Front-End Monitoring Application

In addition to the monitoring framework as detailed in the previous sections, a front-end application to visualise the reasoned and inferred information was developed as well. The goal of this application is to demonstrate the transparency of the approach, as well as its ability to be used as monitoring application as such. The *Data Provider/Resource modules* are only deployed and actively collecting information about the environment when no other jobs are scheduled on the WiLab.t testbed. This ensures that the monitoring framework does not interfere with the experiments scheduled by other users. A screenshot of this monitoring application can be seen in Figure 5.

Three queries were predefined in this application, one for *Incipient Faults*, *Detected Faults* and *HVAC Faults*. The exact implementation of the queries has been detailed in Section 4.4.

The complete workflow results in a list of nodes with their IDs in the case of node classification, and a list of room numbers in the case of Heating, Ventilation and AirConditioning monitoring. These logical IDs are captured in a tree-based view, which the administrative staff can expand to check the results of the reasoned monitoring process. This process is initiated iteratively by the monitoring application. This results in continuous monitoring and reasoning on the information available in the MySQL database. However, by doing this on



```
WiLAB.t Faults
  IncipientMoteFault [06/11/2009 09:21:40 - #12]
    HumiditySensors/159
    HumiditySensors/163
    HumiditySensors/24
    LightSensors/155
    LightSensors/159
    LightSensors/163
    LightSensors/24
    LightSensors/26
    LightSensors/28
    TemperatureSensors/159
    TemperatureSensors/163
    TemperatureSensors/24
  DetectedMoteFault [06/11/2009 09:29:42 - #13]
    TMoteSkySensor/125
    TMoteSkySensor/127
    TMoteSkySensor/150
    TMoteSkySensor/151
    TMoteSkySensor/152
    TMoteSkySensor/153
    TMoteSkySensor/156
    TMoteSkySensor/157
    TMoteSkySensor/159
    TMoteSkySensor/162
    TMoteSkySensor/163
    TMoteSkySensor/164
    TMoteSkySensor/24
  HeatingFault [06/11/2009 09:36:19 - #1]
    Atlantis_Lab
```

Fig. 5. WiLab.t ontology-based monitoring application front-end

historical data, superseded information is sometimes taken into account even after the fault has been rectified. This is a result of the design of the database and the *Data Provider/Resource Modules*' implementation. We plan to enhance this mechanism into an online deployment in future versions of the monitoring framework, where the information is to be fetched at-runtime during the reasoning process, thus also eliminating the need for maintaining a MySQL database with all raw data.

7 Performance Evaluation

Having presented the developed architecture and components supporting an ontology-based monitoring framework using distributed reasoning mechanisms for the IBBT WiLab.t infrastructure in the previous sections, this section details the evaluation of the platform. Apart from checking whether the conclusions drawn automatically by the distributed reasoning process are really correct, we evaluated the overall round trip time starting from the triggering of the reasoning process until the reception of the conclusions.

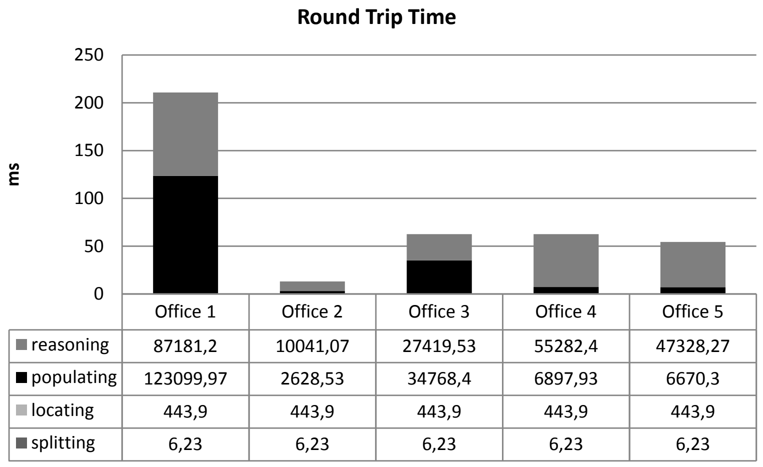
We constructed a WiLab.t test setup, using 5 offices. Each of the offices has on average 6 nodes, producing in total between 500 and 700 measurement reports per office. The scenario measured is as follows:

1. The request is initiated in the back-end, by the monitoring application,
2. The query is analysed by the back-end engine [splitting],
3. The correct iNodes are located by the back-end engine [locating],
4. The reasoning tasks are scheduled on the iNodes,
5. The 5 iNodes collect the local data and convert it into ontology A-Box data [populating],
6. The 5 iNodes execute the reasoning on the local data and return the results to the back-end [reasoning],
7. The back-end merges the results and returns it to the monitoring application.

Of the 7 tasks enumerated here, only task 2, 3, 5 and 6 significantly contribute to the overall round-trip time. The others can be neglected. After all, Task 1 only triggers the action. Secondly, since there is only a single concurrent reasoning task being executed, the scheduling has no effect. Task 7 marks the end of the process, by returning the list of merged results.

The results can be seen in the graph in Figure 6. The reasoning process was triggered 30 times. The average was recorded in the graph for each of the offices. A number of important conclusions can be drawn from this graph. First of all, it is clear that two main contributing phases in the workflow are the *"populating"* and *"reasoning"* tasks. The time the reasoning takes for even a limited number of sensor nodes per iNode clearly underlines the need for a distributed approach in constrained environments. Secondly, the influence of the amount of reports to be included in a single iNode is also of great influence. Office 1 contains the most sensor nodes, namely 13, while office 2 contains the least nodes, namely 3. Thirdly, the populating phase of the reasoning process has a significant contribution to the overall processing times. However, by implementing the intended transition towards an online approach, the need for this population approach can be avoided. We expect this to result in a lower overall round-trip time. Finally, the distribution mechanism implemented in this platform penalises the quicker offices, by not returning the results to the monitoring client until all contributing reasoning tasks have completed. Therefore, the iNode in the office taking the most time to complete will define the overall round-trip time. However, because a post-processing merger phase is included to eliminate potential duplicates, this would otherwise have to be handled by the invoking client.

On the iNodes, Java 1.6.0 update 13 was used to run an Apache Tomcat 6.0 Webserver. The reasoning and populating modules deployed in this container were implemented using Pellet 2.0.0rc4 [17] as reasoner and d2rq6 [18] as populator to convert the raw data from the MySQL database into an ontology A-Box.



	Office 1	Office 2	Office 3	Office 4	Office 5
■ reasoning	87181,2	10041,07	27419,53	55282,4	47328,27
■ populating	123099,97	2628,53	34768,4	6897,93	6670,3
■ locating	443,9	443,9	443,9	443,9	443,9
■ splitting	6,23	6,23	6,23	6,23	6,23

Fig. 6. Total round-trip time including a component break-down for the 4 major contributing phases

8 Conclusions and Future Work

In this paper, we have presented how an ontology-based monitoring framework was developed for IBBT's WiLab.t infrastructure. The adoption of the ontology technology supported by distributed reasoning mechanisms facilitates the transparent monitoring in a heterogeneous environment, where the rules according to which nodes and the environment variables need to classified can be changed according to the locally deployed hardware, specifications and environment. We have detailed the implementation of the contributing components on sensors, iNodes, back-end and front-end. The platform has been evaluated on the testbed through analysis of the processing times of the different contributing components. Moreover, the presented platform is currently in daily use for the monitoring of the WiLab.t testbed. It demonstrates that by introducing intelligent and advanced technologies in a cross-domain manner, great improvements can be achieved both in extendibility and maintainability. As can be concluded from the presentation of the monitoring platform in this paper, changes over time of classification rules are easily implemented through adaptation of the ontology deployed on the iNodes. New hardware can also be included with minimal effort if new ontology models are created modelling the new hardware. The inclusion of this new information, or the adaptation of local rules based on the local environment is achieved transparently from the monitoring application.

We plan to further exploit the developed platform to introduce a permanent monitoring application, by migrating from an offline database backed deployment towards an online fully distributed and autonomous platform. This will not only result in faster response times, but will also avoid the inclusion of potentially superseded information which might still be present in the database, even after a given detected fault has been rectified. Finally, we plan to develop mechanisms to take changing network topologies and deployments into account in an online manner and optimise the deployment based on certain networking metrics.

Acknowledgement

Stijn Verstichel and Eli De Poorter would like to thank the IWT (Institute for the Promotion of Innovation through Science and Technology in Flanders) for financial support through his Ph.D. grant. Tim De Pauw would like to thank the University College Ghent Research Fund for financial support through his Ph.D. grant. This research is partly funded through the IBBT-DEUS project.

References

1. Xu, N.: A survey of sensor network applications. IEEE Communications Magazine 40(8), 102 (2002)
2. Murty, R.N., Mainland, G., Rose, I., Chowdhury, A.R., Gosain, A., Bers, J., Welsh, M.: CitySense - An Urban-Scale Wireless Sensor Network and Testbed. In: IEEE Conference on Technologies for Homeland Security, Waltham, MA, pp. 583–588 (2008)
3. Ertin, E., Arora, A., Ramnath, R., Nesterenko, M., Naik, V., Bapat, S., Kulathumani, V., Sridharan, M., Zhang, H., Cao, H.: Kansei: a testbed for sensing at scale. In: The Fifth International Conference on Information Processing in Sensor Networks, IPSN 2006, Nashville, TN, pp. 399–406 (2006)
4. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab - a wireless sensor network testbed. In: Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, pp. 483–488 (2005)
5. WISEBED - Wireless Sensor Network Testbeds. Part of the Seventh Framework Information Communication Technologies program from the European Commission under project number 224460, `http://www.wisebed.eu/`
6. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition (1993)
7. da Rocha, A.R., Delicato, F.C., de Souza, J.N., Gomes, D.G., Pirmez, L.: A semantic middleware for autonomic wireless sensor networks. In: Proceedings of the 2009 Workshop on Middleware For Ubiquitous and Pervasive Systems, WMUPS 2009, Dublin, Ireland, June 16, vol. 389, pp. 19–25. ACM, New York (2009), doi:http://doi.acm.org/10.1145/1551693.1551697
8. Hu, Y., Wu, Z., Guo, M.: Ontology driven adaptive data processing in wireless sensor networks. In: Proceedings of the 2nd International Conference on Scalable Information Systems, June 6-8. ACM International Conference Proceeding Series, vol. 304 (2007)
9. Delicato, F.C., Pirmez, L., Pires, P.L., Ferreira de Rezende, J.: Exploiting Web Technologies to Build Autonomic Wireless Sensor Networks. In: Mobile and Wireless Communication Networks, vol. 211, pp. 99–114. Springer, Boston (2006)
10. TMoteSky, `http://www.moteiv.com/`
11. Alix 3C3, `http://www.pcengines.ch/alix3c3.htm`
12. Voyage Linux, `http://linux.voyage.hk/`

13. Verstichel, S., Ongenae, F., Volckaert, B., De Turck, F., Dhoedt, B., Dhaene, T., Demeester, P.: An autonomous service-platform to support distributed ontology-based context-aware agents, In: Special Issue of Expert Systems: The Journal of Knowledge Engineering on Engineering Semantic Agent Systems (2009) (accepted for publication)
14. Sparql: Query Language for RDF, W3C Candidate Recommendation (2007), `http://www.w3.org/TR/rdf-sparql-query/`
15. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection Tree Protocol. In: SenSys 2009, Berkeley, California, USA, November 4-6 (2009)
16. Avancha, S.: Sensor Node Ontology, `http://ebiquity.umbc.edu/resource/html/id/131/Sensor-Node-Ontology`
17. Parsia, B., Sirin, E.: Pellet: An OWL DL Reasoner. In: Proceedings of the International Workshop on Description Logics (2004), `http://pellet.owldl.com/`
18. Bizer, C., Cyganiak, R.: D2R Server, Publishing Relational Databases on the Semantic Web. In: Proceedings of the 5th International Semantic Web Conference. Athens, GA, USA (2006)