A Policy-based Storage Model for Sensor Networks

Nuno M. F. Gonçalves, Aldri L. dos Santos and Carmem S. Hara
Universidade Federal do Paraná
Curitiba-PR, Brazil P.O.Box 19081 81531-980
Email: {nunom, aldri, carmem}@inf.ufpr.br

***Abstract*—Policies are used for developing adaptable and flexible systems in a variety of areas. They are especially suitable for reducing the complexity of managing tasks, by providing a mechanism for automatically tuning the system without human intervention. Policy-based systems have been applied for wireless sensor networks (WSNs) for controlling several functionalities. However, none of them has been proposed as a storage model, by making a clear distinction between storage functions and their behaviour. In this paper we propose SeSP, a *Se*nsor *S*torage model based on *P*olicies. SeSP explores concepts that are common in storage models proposed for WSNs in order to reduce the number of message transmissions and thus minimize the sensors' energy consumption. We have conducted a case study applying our policy-based system on two existing storage models: Scoop and DYSTO. Our experimental study, based on simulations, shows that SeSP can effectively reduce the number of transmissions, compared to the fixed values considered by both systems.**

I. Introduction

Policies are defined by rules that determine the behaviour of a system [1]. They are used for developing adaptable and flexible systems in a variety of areas such as distributed systems, and data and network security. As the complexity of these systems increases, managing their resources also becomes a hard task. Thus, it is desirable to rely on a mechanism that can provide automatic tuning of the system without human intervention. Policy-based mechanisms are a suitable approach since they make a clear distinction between the system's functionality and behaviour. As a consequence, it is possible to adapt the system's behaviour according to its needs without changing its functions, and without interruptions. Hence, such capabilities are essential to support the management of several types of networks and their services.

Wireless sensor networks (WSNs) are composed of small devices with low processing power and data storage. They have been applied for monitoring applications, which involve capturing sensing data from the monitored environment, and then storing and querying the sensed information. In general, sensor devices present limited energy and communicate with each other via short-range radio signals [2]. In WSNs the energy consumption for transmissions is the main factor for shorter network longevity and consequent loss of data [3]. Solutions have been proposed taken into account energy awareness. Thus, an important topic of investigation for WSNs is on minimizing the number of transmissions both for storing and querying sensed data.

Policies have been applied in the context of WSNs for providing an adaptable routing mechanism [4], for providing reliable message delivery [5], for controlling the network functionality [6], and as a means for application development [7]. To the best of our knowledge there exists no policy-based storage model proposed for WSNs. However, there are a number of models that apply policies implicitly. As an example, consider models that group sensors into clusters with a designated sensor for storing the group members' readings, denoted as the group *repository*. Some existing models have implicit rules for determining group membership. CAG [8] and SIDS [9], for instance, have an implicit rule which determines that a sensor should migrate to a different cluster, or start a new one, if its reading is not within the expected interval of values defined for the cluster. Others, determine the periodicity in which sensor readings are updated in the repository. It can be in fixed intervals, such as in Scoop [10], or based on the difference between the current reading and the previous reported one, as in HDMST [11] and DACS [12]. Models also apply implicit rules for moving the repository to a different sensor. The decision can be based on the sensors' residual energy [13], rotation among cluster members [14], proximity to other nodes [15], or caused by a failure on the current repository [16], or by changes on the number of queries and sensor readings updates [10].

In this paper we propose a sensor storage model based on policies called SeSP (**Se**nsor **S**torage model based on **P**olicies). SeSP explores concepts that are common in storage models proposed for WSNs in order to reduce the number of message transmissions and thus minimizing the sensors' energy consumption. Periodicity of sensor reading updates and frequency for determining new repositories are examples of concepts. This set of concepts compose the basis of our policy-based system. That is, a policy is a set of user-defined rules that determine the behaviour of the system by setting values for each concept. Moreover, rules can be dynamically changed without interrupting the system. We have conducted a case study applying our policy-based system on Scoop [10] and DYSTO [17]. Our experimental study, based on simulations on a real data set [18], shows that our system can effectively reduce the number of transmissions, compared to the fixed values considered by both systems.

The paper is organized as follows. Section II presents related work. Section III describes SeSP architecture and language. Section IV presents the case study and Section V reports the performance evaluation. We conclude in Section VI highlighting future works.

II. Related work

In recent years WSNs evolved significantly. From simple networks with few computational capabilities and communication, they have become complex networks with various levels of data abstraction and with a constant increase on the volume of generated data [2]. This increase on the volume of data rose new challenges to data management and storage.

The selection of the location for data storage is critical for optimizing the available resources. If the system scenario changes significantly during its execution then it is intuitive to think that the ideal solution to reduce the cost of transmissions both for data storage and querying may also change. A policy can be defined as a set of rules that provide adaptability and flexibility to the system. Many works have proposed different solutions for WSNs based on policies, both in data management and routing mechanisms.

SRM [5] is based on a hierarchical management architecture and policy-based network management paradigm. SRM consists of four modules: a user policy specification module, an evaluation module, a decision making module and an action module. The interaction among these modules ensures that the network provides adequate information to the users while reducing energy consumption. SRM uses the Policy Framework Definition Language (PFDL) with simple policies[19]. PARAW [4] offers a high-level and flexible way to execute management tasks related to routing in WSNs, which can be defined progressively as more knowledge from the environment is acquired or as the application requirements change. PTSN [6] presents a policy-based paradigm for fine-tuning and optimization of the WSN runtime environment. CaPI [7] is an expressive policy language that supports the specification and management of behavioural concepts by administrators or domain experts. CaPI provides a clear decoupling between application logic and behaviour, enabling efficient customization and dynamic reconfiguration of the application functionality and behaviour.

DYSTO [17] is an in-network and dynamic data storage, inspired by Scoop [10], in which the location of the repository is chosen based on information collected over the network. Intuitively, both Scoop and DYSTO choose to store sensed data close to where it is most frequently needed: close to the query entry point when the query rate is high, and close to data sources for coping with high sensed data production rates. Moreover, DYSTO analyzes the query load in order to keep values that are frequently queried together in the same repository. DYSTO also focuses on reducing the power consumption of the network based on an adaptive approach for repository selection, and a user-defined update strategy based on data thresholds. Another system that dynamically switch between different storage models is proposed in [20], based on an implicit rule that determines the appropriate one for the current context.

In our proposed approach, on the other hand, storage models change based on user-defined rules. Thus, SeSP provides a storage system that can adapt to different scenarios in order to optimize the available resources without decreasing the overall quality of the system output. Our policy definition language is also based on the concepts defined by the Policy Framework Definition Language (PFDL) [19], but follows an XML format with elements defined based on the components of our storage model, as described in Section III.

III. A Policy-based Storage Model

In WSNs value-based queries are particularly hard to process. For instance, if the system employs a local storage model, in which each sensor stores its own reading, in order to obtain the set of sensors (or their geographical location) with values in a given interval, every sensor on the field have to be contacted. Common approaches for reducing the number of transmissions for processing such queries, and thus their energy consumption, consist of electing sensors to store the readings of a group of devices, and create indexing structures on the repository sensors. Following this approach, in this section we first identify the components of a sensor storage model, and their functionality. Our policy-based storage model, called SeSP (**Se**nsor **S**torage model based on **P**olicies), is defined on these components for controlling their behaviour.

A. The Storage Model

A WSN is represented as a graph $G = (S, L)$, where $S = \{s_1, \ldots, s_n\}$ is a set of sensors spread over a monitored area M, and L is a set of links such that (s_i, s_j) is in L if s_i and s_j are within the radio communication range of each other. We say that the distance between s_i and s_j is *one-hop* and that s_i and s_j are *neighbors*. Communication between two arbitrary sensors requires the existence of links $\{(s_1, s_2), (s_2, s_3), \ldots (s_{n-1}, s_n)\}$ such that s_1 is the sensor that originates the message and s_n is the message final destination. That is, WSNs are based on *multi-hop* communication, and rely on a *routing* protocol for determining paths that establish communication between sensors. In our model, we assume that sensors are static, and thus have a fixed geographic coordinate. To simplify our discussion, we also assume that each sensor is responsible for monitoring a single measurement from the environment. We denote the current reading of a sensor s as *reading(s)*, and its domain of values as D.

Some sensors in the network are designated as *repositories*. That is, they are responsible for storing the readings of a group of sensors such that only repositories may have to be contacted in order to answer value-based queries. In our model, there exists a commodity computer, the *base station (BS)*, which is the single access point for query injection. The BS is also responsible for grouping sensors into clusters and for determining a repository for each of them. Observe that sensors can be grouped based either on their location or on their readings. If the model is location-based, we can define clustering as a mapping $C_l : S \rightarrow S$ from each sensor to its group's repository. That is, given two sensors s_1 and s_2, if $C_l(s_1) = C_l(s_2)$ then s_1 and s_2 belong to the same cluster, and they send their readings to the same repository. On the other hand, if grouping is based on the sensors' reading, clustering can be defined as a function $C_v : (D \times D) \rightarrow S$. That is, a sensor repository s in S is chosen for each interval of values $i = [v_1, v_2)$ in $(D \times D)$. That is every sensor with a reading v in the interval i notifies the repository $C_v(i)$ of its current reading.

As an example, consider the set of sensors in Figure 1, where each sensor is depicted with its identifier and current reading. Observe that there are three repositories (s_4, s_6, and s_8), each one in a cluster composed of $\{s_2, s_3, s_4\}, \{s_1, s_5, s_6\}$, and $\{s_7, s_8\}$. The clustering can be

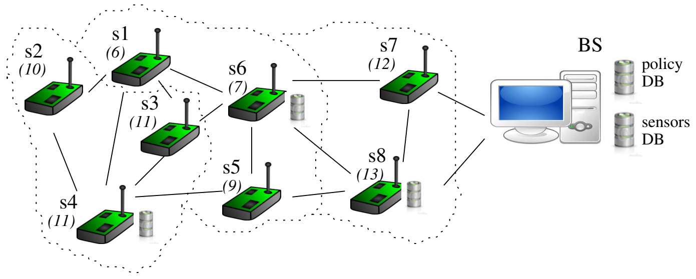


Fig. 1. SeSP components

defined based on the sensor location by the following function: $C_l = \{s_1 \mapsto s_6, s_2 \mapsto s_4, s_3 \mapsto s_4, s_4 \mapsto s_4, s_5 \mapsto s_6, s_6 \mapsto s_6, s_7 \mapsto s_8, s_8 \mapsto s_8\}$. The same clustering can be by defined by the following value-based function: $C_v = \{[6-10) \mapsto s_6, [10, 12) \mapsto s_4, [12, 14) \mapsto s_8\}$.

Information on all components of the storage system is stored in the *Sensors DB*. Besides, the behaviour of the system relies on four types of transmissions: monitoring, mapping, update, and query/reply messages. *Monitoring* messages are sent from each sensor to the BS for communicating its current status. It contains information required by the BS for clustering sensors and choosing the repositories. The clustering function is communicated to all sensors by *mapping* messages. Based on this function, sensors can determine to which repository they should send their current reading by an *update* message. Inquiries on these readings are issued on the BS, which sends a *query/reply* message to the repositories.

To illustrate, consider a clustering function based on the sensors' readings. Each sensor sends a *monitoring* message to the BS with information such as a histogram of its last readings. The periodicity in which sensors send this type of message can be defined by a local policy, defined for each sensor, or by a global policy, defined for the entire WSN. Based on the information collected from the entire network, the BS defines how sensors should be grouped, and for each of them, determines which sensor is defined as the group repository. Suppose that the result of this step is function $C_v = \{[6-10) \mapsto s_6, [10, 12) \mapsto s_4, [12, 14) \mapsto s_8\}$. This function is broadcast to all sensors on the field by a *mapping* message. Similar to *monitoring* messages, the frequency in which the BS regroups sensors or change the repository may be defined by the policy system. Every sensor locally stores the clustering function, such that whenever it obtains a new reading, it may communicate the appropriate repository. Suppose, for example, that sensor s_5, which previously had a reading of 9, gets 11 as its new reading. Thus, instead of reporting its value to repository s_6, it now sends an *update* message to repository s_4, along with the reading timestamp. The periodicity each sensor sends an *update* message can also be controlled by the policy system. Value-based queries are issued on the BS. Consider, for example, a query to determine which sensors have readings between 8 and 11. Based on the last clustering function, the BS determines that only repositories s_6 and s_4 have to be contacted. Thus, it sends *query* messages to these sensors, which in turn send back *reply* messages to the BS with the query answer.

Once the basic functionality of the system has been defined, they can be refined by a set of rules, as described in the next section.

B. A Policy-based Storage System

A policy is a set of rules that express decisions made to achieve specific goals [1]. Observe that an isolated command to execute an action is not a policy. Policies are persistent and should dynamically change the behaviour of a system without interrupting its functionality. Thus, in our policy-based system, the BS stores the rules that compose its policy in a *Policy DB*, as depicted in Figure 1.

1) Policy Definition Language: A policy is a set of rules of the form (E, T, C, A) where E is the rule's subject, T is the target, C is the condition to be satisfied for the application of the rule, and A is the action to be performed. In SeSP these elements are defined in XML format. XML has been chosen because it is a flexible format, allowing rules with new actions and variables in conditions to be considered by extending the XML Schema used to validate the input document.

The subject E is the element that authenticates the rule and executes the action. In SeSP, the subject can be either the BS or a set of sensors. The target T is the element on which the action is applied. The target can be defined as `global` if the rule applies to all sensors on the field or `local`, if it targets only a set of sensors. The condition C defines the requirements for the application of the rule. A requirement may be `temporal` and/or involve system defined `variables`. Temporal conditions provide a means for expressing distinct behaviour on different periods of time. They may be defined with a `start` and `end` time or only with the `start` time. Variable conditions, on the other hand, are defined on the sensor's reading (`current_val`) or on variation of values of the storage model components: the variation on the sensor reading (`var_update`), the variation on the system monitoring information (`var_monitoring`), and the difference on the result of the clustering function `var_mapping`.

The action element A identifies the set of actions to be applied when the requirements are met. Currently, we consider two types of actions: send alert messages (`sendMsg`), and determine the periodicity on which each type of message is transmitted (`setParam`). The periodicity can be defined as fixed intervals, or based on the variation of values, similar to the condition element. For instance, it can be determined that monitoring messages should be transmitted within fixed intervals of 110 seconds (`fix_monitoring = 110`) or when it varies by 10% (`var_monitoring = 10`). Similarly, we can define the periodicity of sensor readings update (`fix_update, var_update`) and mapping messages (`fix_mapping, var_mapping`).

Examples of rules are given in Figures 2 and 3. The rule in the first example sets the frequency of transmissions for each type of message. Update and monitoring messages are transmitted based on their variations, which are set to 1% and 30%, respectively (Lines 12 to 19). Mapping messages, on the other hand, are transmitted within fixed intervals of 240 seconds (Lines 20 to 23). These frequencies are valid for all sensors on the field since the rule is defined to be global

(Line 4), and apply from the first second of the system runtime until the 2600th second (Lines 6 to 9). Observe that the ability to define temporal conditions is particularly interesting when the system presents behavioural seasonality. The example in Figure 3 shows a rule that targets only sensor s_3, determining the transmission of an alert message when its sensing value rises to a value above 50 degrees.

```
1.  <policy>
2.     <rule>
3.        <subject> <eb /> </subject>
4.        <target> <global /> </target>
5.        <condition>
6.           <temporal>
7.              <start> 1 </start>
8.              <end> 2600 </end>
9.           </temporal>
10.       </condition>
11.       <action>
12.          <setParam>
13.             <parameter> <var_update /> </parameter>
14.             <value>1</value>
15.          </setParam>
16.          <setParam>
17.             <parameter> <var_monitoring /> </parameter>
18.             <value>30</value>
19.          </setParam>
20.          <setParam>
21.             <parameter> <fix_mapping /> </parameter>
22.             <value>240</value>
23.          </setParam>
24.       </action>
25.    </rule>
26. </policy>
```

Fig. 2. Example of policy for setting parameters

```
1.  <policy>
2.     <rule>
3.        <subject>
4.           <sensor> s3 </sensor>
5.        </subject>
6.        <target>
7.           <local>
8.              <sensor> s3 </sensor>
9.           </local>
10.       </target>
11.       <condition>
12.          <variable>
13.             <parameter> <current_val /> </parameter>
14.             <operator> &gt;  </operator>
15.             <value> 50 </value>
16.          </variable>
17.       </condition>
18.       <action>
19.          <sendMsg>
20.             Emergency!! Risk of fire!!
21.          </sendMsg>
22.       </action>
23.    </rule>
24. </policy>
```

Fig. 3. Example of policy to send an alert message

2) Architecture: The functionality of our policy-based storage system relies on four main components, as shown in Figure 4: XML input, the *Policy DB*, policy points of decision (PPD) and policy points of application (PPA). The *XML Input* contains a set of rules, as described in Section III-B1. The rules are interpreted by the BS and stored in the *Policy DB*.

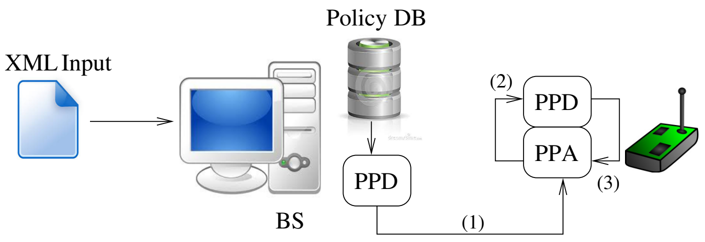


Fig. 4. Policy components

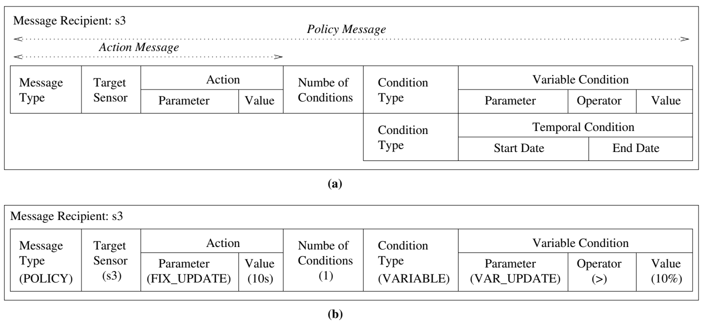
The PPDs are responsible for actively checking whether the requirements of the active rules are met. When the conditions are satisfied, PPDs are responsible for dispatching an action message to a PPA, which in turn implements the action. PPD agents are executed both by the base station and by sensors. The base station's PPD is responsible for verifying all active policies stored in the repository, and it is the only agent with access to the *Policy DB*. A sensor PPD, on the other hand, only verifies local rules that have as subject the sensor itself.

The base station PPD can dispatch actions to be executed by a sensor PPA and also remotely update the rules defined in one or more sensors. In general, local rules are simple, since sensors have limited resources. The PPAs are executed only by sensors. As depicted in Figure 4, the PPA agents listen to the network for action messages sent by the base station's PPD (1) and also rules to be executed by its own PPD, and implements the action. If the message contains a policy update, the PPA updates the sensor local rules storage (2), which are interpreted by the sensor's PPD for activating actions executed by the PPA (3).

Observe that there are two types of messages that are sent from a PPD to a PPA: action and policy update messages. Action messages consist of: a target sensor and the action to be performed. In addition to these fields, policy update messages contain a set of conditions, as illustrated in Figure 5(a). The number of conditions is defined by the field `number of conditions`, while the `condition type` can be either `variable` or `temporal`. As an example, the policy update message in Figure 5(b) is sent from the BS to sensor s_3 determining that the sensor reading updates at its repository should be increased to fixed intervals of 10 seconds when the variation between the last value monitored and the current value is above 10%.

IV. Case Study

We have applied our storage model to implement DYSTO [17], a dynamic storage model that chooses the data repositories based on the frequency of sensor reading updates, and the volume of queries issued to the system. DYSTO has been inspired by Scoop [10] and they are both based on a clustering function that maps value intervals to repositories (as function C_v defined in Section III-A) and their placement is based on the following observation: repositories should be close to where they are needed more often in order to reduce the number of transmissions. That is, with an increase of queries on a given interval, the corresponding repository should be


Message Recipient: s3

Policy Message

Action Message

Message Type	Target Sensor	Action		Numbe of Conditions	Condition Type	Variable Condition		
		Parameter	Value			Parameter	Operator	Value
					Condition Type	Temporal Condition		
						Start Date	End Date	

(a)

Message Recipient: s3

Message Type	Target Sensor	Action		Numbe of Conditions	Condition Type	Variable Condition		
		Parameter	Value			Parameter	Operator	Value
(POLICY)	(s3)	(FIX_UPDATE)	(10s)	(1)	(VARIABLE)	(VAR_UPDATE)	(>)	(10%)

(b)

Fig. 5. Action and policy update messages

moved closer to the BS, which is the query injection point. On the other hand, if the frequency of sensor reading updates is high, then the repository should be placed close to the sensors that produce values within the interval.

Since clustering and repository placement are value-based, the BS stores information on the frequency of queries on the domain of values produced by the sensors. In addition, sensors send *monitoring* messages to the BS containing a histogram of their last readings. The size of the buffer containing the readings and number of entries (or bins) in the histogram depend on the sensor's storage capacity. Both DYSTO and Scoop report experiments with a buffer of 30 readings and a histogram with 10 equiwidth bins. That is, the interval of the minimum and maximum readings in the buffer is divided into 10 subintervals of the same size. Then, each bin is filled with the quantity of readings in the buffer within its corresponding subinterval. Based on the sensors' histograms and the query frequency the BS computes the clustering function C_v which is communicated to all sensors by *mapping* messages. *Update* and *query/response* messages provide the means for updating a repository with the sensor's current reading and to answer queries, as described in Section III-A.

The time interval adopted by DYSTO and Scoop between transmissions of each message type differs. Scoop defines fixed intervals for sending mapping, update, monitoring messages (240, 75, and 110 seconds, respectively). In SeSP they can be implemented by defining action rules to assign values to `fix_mapping`, `fix_update`, and `fix_monitoring` parameters. On the other hand, DYSTO is based on user-defined thresholds that determine the maximum difference between the contents of two consecutive messages. For instance, the variation between two consecutive readings is defined by $|1 - (current\_reading/previous\_reading)|$, while the variation between two monitoring messages is based on the value associated with a histogram h, which is defined as the average value of all sensor readings in its buffer. We denote this value as $M(h)$ and define the difference between two histograms h_1 and h_2 as $|1 - (M(h_1)/M(h_2))|$. The difference between two mapping functions can be computed similarly based on the number of sensors that moved to a different repository. Given these definitions, they can be implemented in SeSP by defining action rules to assign values to `var_mapping`, `var_update`, and `var_monitoring` parameters.

Observe that our policy-based storage system is quite general and can be applied to model a number of storage models proposed in the literature, as long as functions that implement their basic components are provided. Namely, the BS should be provided with functions for computing the clustering of sensors based on the monitoring messages, and determine the difference between two clustering functions. Sensors should be provided with functions that: compute the contents of the monitoring message, determine the difference between the results of two monitoring messages, and determine the difference between two consecutive readings. Observe that in our study case, monitoring information is based on the sensors' readings. However, a common approach for modifying sensor clustering and repository location is to define a function based on the sensors' remaining energy. In this case, monitoring messages received by the BS simply contain the sensors' residual battery level in order to elect new repositories.

V. Experimental Study

We have conducted simulations in order to validate SeSP, and to determine whether it does have an impact on the number of transmissions of a WSN. SeSP has been implemented on NS2 network simulator version 2.34 [21], along with functions that implement the components of both DYSTO [17] and Scoop [10], as described in Section IV. Given that energy consumption on WSNs is dominated by the communication overhead [3], our cost metric is the total number of transmissions sensors and the BS send collectively. For a study on the impact of the fixed intervals adopted by Scoop compared to the threshold-based intervals defined by DYSTO, we refer the interested reader to our previous work [17]. Since the impact of the two different forms of determining transmission rates has already been assessed, here we report experiments on the system ability to correctly interpret and apply policies (Section V-B) and on the impact of defining different rules at distinct periods of time (Section V-C).

A. Simulation Settings

We have evaluated SeSP using a real data set made available by the Intel Lab Data [18]. The trace was collected from 54 Mica2 motes deployed in the Intel Research Lab over a period of 35 days. The data set comprises a timestamp and information on temperature, humidity, light, and voltage. However, only temperature measures along with the sensor locality information were used in our simulations. We refer to this setting as the *real scenario*.

In order to evaluate the model on a larger WSN, we have also generated a *synthetic scenario* composed of 500 sensors on a 500×500 square meter field, with similar characteristics on their sensing data as the real trace. Based on the observation that the metrics collected in the real scenario present highly spatially correlated readings, we have generated the sensors' initial readings using a Matlab tool [22], which has been especially designed to produce data with this property. The tool receives as input a correlation coefficient (h) and the size of the monitoring area (m). It generates as output a matrix D of dimension $m \times m$, used to determine sensors readings as follows: each sensor s, randomly placed at a position (x_s, y_s), gets as reading the value at $D[\lfloor x_s \rfloor], \lfloor y_s \rfloor]$. The correlation coefficient determines the level of similarity. That is, $h = 0$ generates data with no spatial similarity, and higher values of h induces higher spatial correlation. All simulations described in this section have been executed on a scenario generated with high correlation ($h = 9$). Each of the values initially calculated were then updated applying a random variation of zero to 30%, applied on the initial value, based on the sensor location in order to preserve the spatial similarity.

The simulation parameters are presented in Table I. All results reported in this section consider these values as default unless otherwise specified, and consist of the average value collected from five executions on each simulation setting.

Parameter	real scenario	synthetic scenario
Network devices	54 sensors + 1 BS	500 sensors + 1 BS
Data source	real data[18]	synthetic data
Sensor communication range	30 meters	
Simulation duration	40 minutes	
Sample rate	1 sensor reading every 15 seconds	
Query rate	1 query every 15 seconds	
Mapping messages	transmission every 240 seconds	

TABLE I. SIMULATION PARAMETERS

B. Correctness of the Policy Implementation

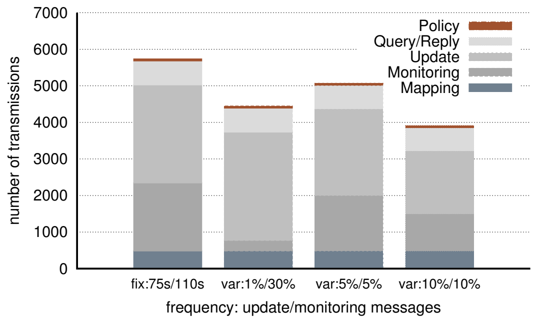
The goal of this experiment is twofold: first, to determine the impact of policy messages on the overall number of transmissions; and second, to determine the correctness of the system's behaviour according to the rules given as input.

Figure 6 shows the number of transmissions of each type of message in four different settings. In each of them we consider that a policy similar to the one presented in Figure 2 has been defined for setting the frequency of message transmissions and that the same rate has been maintained throughout the experiment. We have chosen four settings based on the results reported previously by [10] and [17] in order to determine whether our results are consistent with them.

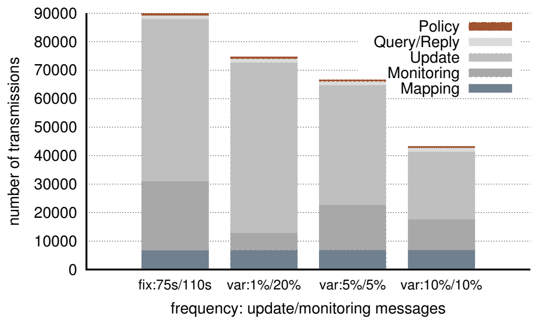
The first column in both graphs of Figure 6 corresponds to the frequency chosen by Scoop: fixed intervals of 240 seconds for mapping messages, 75 seconds for update messages, and 110 seconds for monitoring messages. The same rate for mapping messages has been adopted for all the other settings, but defining the frequency of update and monitoring messages based on the variation of their contents: `var_update` of 1% combined with `var_monitoring` of 30% for the real scenario and 20% for the synthetic scenario; both parameters set to 5%; and both parameters set to 10%. The first combination has been chosen combining parameters that produce query results with maximum relative error of 1% and 0.5% in average, with monitoring message rates that produce the maximum reduction on the total number of transmissions, based on the experiments reported in [17].

In this experiment since a single policy has been sent once from the BS to all sensors, the impact of policy messages is only 1% of the total number of transmissions in both scenarios. The values collected for the other types of messages are consistent with the ones reported by previous works. That is, compared to the fixed rates adopted by Scoop, variable rates reduce the overall number of transmissions by 22.72%, 11.74% and 32.15% in the real scenario, and 17.02%, 26.01% and 52.15% in the synthetic scenario.

We have also run experiments with rules that send alert messages, similar to the one presented in Figure 3. In this case, we tried to simulate a real situation of service break due to a



(a) Real Scenario



(b) Synthetic scenario

Fig. 6. Policy messages impact

high temperature in a server room at night. That is, the rule was defined both with a temporal condition (`start` 700 and `end` 1700) and a variable condition on the sensor current reading (`current_val` > 40). In the experiment we have injected several readings that exceeded the maximum value inside and outside the policy's action time using the real scenario. The results can be seen on Table II. They show that the sensor correctly interpreted the policy and sent the messages correctly when necessary, although two of them did not arrive at the BS. We conjecture that the cause for the missing messages was the lack of synchronization on the start and end time the alert rule was active.

System Time(s)	Readings above limit	Alert Messages at the BS
0-700	12	0
700-1700	20	18
1700-2400	14	0

TABLE II. EXPERIMENT WITH ALERT MESSAGES

C. Dynamic Thresholds

The goal of this experiment is to determine whether SeSP correctly modifies the message transmission frequencies when there are different rules that apply at distinct periods of time. We have defined a rule that changes the frequency of monitoring messages of a single sensor s_3 as follows: from time 600 to 1200 `var_monitoring` is set to 10%; and after time 1500 the same parameter is set to 5%. Figure 7 shows the number of monitoring messages dispatched from s_3 at different execution times, comparing it to a policy with fix transmission rates of 110 seconds.

Observe that the fix rate applies in all the other periods of time in which parameter `var_monitoring` is not set.

Thus, from time 0 to 600 both bars report the transmission of 7 messages in the period. At time 600, the policy that sets `var_monitoring` to 10% is activated, reducing the number of messages in the period by 57%. In the beginning of the third interval the fix transmission rates is restated, but at the end of this period `var_monitoring` is set to 5%. The activation of the rule reduces the number of messages in the period from 7 to 6. In the last interval 4 messages is transmitted, which represents an improvement of 43% in comparison to the fix rate rule.

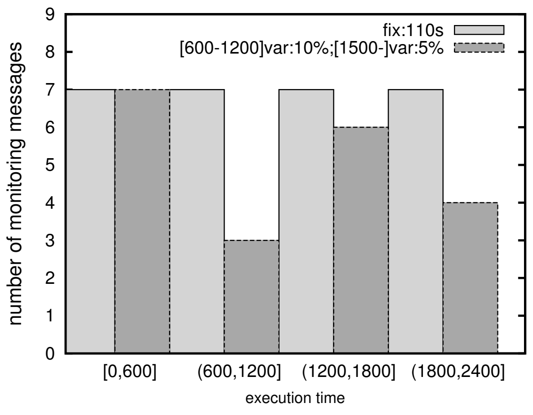


Fig. 7. Impact of changes on `var_monitoring`

This experiment shows that the our system provides adaptability to a storage model. Changes on the system behaviour are usually hard to be implemented without coding and interruption of the system.

VI. Conclusion and future work

In this paper we proposed an adaptable and flexible storage model called SeSP (**Se**nsor **S**torage model based on **P**olicies). Our first contribution is on identifying components and types of messages that are common to several storage models that rely on a single entity (the BS) for choosing a data repository, and for determining sensor clustering, which can be based on any information sent from the sensors to the BS in a monitoring message. Thus, no matter the algorithm for performing such tasks, the types of messages exchanged are exactly the same. These components are frequently found in storage models proposed for WSNs in the literature, and define the basic functionality provided by WSN's storage systems. Based on the types of messages exchanged between the components, we have defined a set of parameters to control their transmission rates. By making a clear distinction between the components' functionality and a control mechanism defined by policy rules, we are able to provide dynamic adaptability to a number of existing storage models. We have described a study case that shows how SeSP can be applied to implement two similar storage models: Scoop [10] and DYSTO [17]. Our experimental study, based on simulations, shows that, unlike these two previous models, SeSP can dynamically change the storage system behaviour, providing a means for adopting distinct configurations suited for different scenarios. We believe that this is a first step towards self management and self configuration capabilities for WSN storage systems.

There are still a number of issues to investigate in the future. Some of them are related to the policy system itself. We plan to develop a mechanism to check whether there exist conflicts among policy rules, and provide the ability to define precedence between rules. A possible approach would be to determine that for each sensor, local rules have precedence over global rules. We also intend to extend the parameters and monitoring actions provided by the system. With respect to the storage model, we would like to generalize it to handle multiple sensor attributes, and to consider hierarchical storage models that combine location and value-based mappings.

Acknowledgment. This work is supported by the following Brazilian research agencies: RNP, FINEP, CAPES, and CNPq.

References

[1] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, "Terminology for policy-based management rfc 3198," The Internet Engineering Task Force (IETF), Tech. Rep., 2011.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102 – 114, aug 2002.

[3] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, pp. 51–58, May 2000.

[4] C. M. S. Figueiredo, A. L. dos Santos, A. A. F. Loureiro, and J. M. Nogueira, "Policy-Based Adaptive Routing in Autonomous WSNs," *Ambient networks: 16th IFIP/IEEE International*, 2005.

[5] T. D. Le, W. Hu, S. Jha, and P. Corke, "Design and implementation of a policy-based management system for data reliability in Wireless Sensor Networks," *2008 33rd IEEE Conference on Local Computer Networks (LCN)*, pp. 762–769, Oct. 2008.

[6] N. Matthys, C. Huygens, and D. Hughes, "Policy-driven tailoring of sensor networks," *Sensor Systems and Software, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 57, pp. 20–35, 2011.

[7] N. Matthys, C. Huygens, D. Hughes, S. Michiels, and W. Joosen, "A Component and Policy-Based Approach for Efficient Sensor Network Reconfiguration," *2012 IEEE International Symposium on Policies for Distributed Systems and Networks*, pp. 53–60, Jul. 2012.

[8] S. Yoon and C. Shahabi, "The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 3, March 2007.

[9] S. S. Furlaneto, A. L. dos Santos, and C. S. Hara, "An efficient data acquisition model for urban sensor networks," in *Proc. of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, Apr. 2012, pp. 113–120.

[10] T. M. Gil and S. Madden, "Scoop: An Adaptive Indexing Scheme for Stored Data in Sensor Networks," in *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, Apr. 2007, pp. 1345–1349.

[11] K.-C. Yang, Y.-C. Yang, C.-L. Lin, and J.-S. Wang, "Hierarchical Data Management for Spatial-Temporal Information in WSNs," *2010 Fourth International Conference on Sensor Technologies and Applications*, pp. 435–440, Jul. 2010.

[12] N. Hoeller, C. Reinke, J. Neumann, S. Groppe, F. Frischat, and V. Linnemann, "DACS: A dynamic approximative caching scheme for Wireless Sensor Networks," in *Digital Information Management (ICDIM), 2010 Fifth International Conference on*. IEEE, 2010, pp. 339–346.

[13] Y. Zuang, H. Wang, and L. Tian, "Energy and data aware clustering for data aggregation in wireless sensor networks," in *Proc. of the IEEE Int. Conf. on Mobile Adhoc and Sensor Systems Conf.* Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 1–6.

[14] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Trans. on Wireless Comm*, vol. 1, no. 4, pp. 660–670, 2002.

[15] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, “Optimal energy aware clustering in sensor networks,” *Sensors*, vol. 2, pp. 258–269, 2002.

[16] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, “Data-centric storage in sensornets with GHT, a geographic hash table,” *Mobile networks and applications*, vol. 8, no. 4, pp. 427–442, 2003.

[17] N. M. F. Gonçalves, A. L. dos Santos, and C. S. Hara, “Dysto - a dynamic storage model for wireless sensor networks,” *Journal of Information and Data Management*, vol. 3, no. 3, pp. 147–162, 2012.

[18] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, “Intel Berkeley Research Lab,” http://db.csail.mit.edu/labdata/labdata.html, Intel Berkeley, 2004.

[19] J. Strassner and S. Schleimer, “Policy framework definition language,” 1998. [Online]. Available: http://tools.ietf.org/id/draft-ietf-policy-framework-pfdl-00.txt

[20] G.-J. Yu, “Adaptive storage policy switching for wireless sensor networks,” *Wireless Personal Communications*, vol. 48, no. 3, pp. 327–346, Feb. 2009.

[21] “The Network Simulator ns-2 (v2.34),” http://www.isi.edu/nsnam/ns/, 2008.

[22] A. Jindal and K. Psounis, “Modeling spatially correlated data in sensor networks,” *ACM Transactions on Sensor Networks*, vol. 2, pp. 466–499, 2006.