Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.1 No.2, 2012, pp.1-8

DOI: https://doi.org/10.51983/ajcst-2012.1.2.1700

Service Selection Using Non-Functional Properties in MANETs

K. Ponmozhi[1] and R.S. Rajesh[2]

[1]*Research Scholar, Computer Science and Engineering, Mother Teresa Women's University, Kodaikanal - 624 101, Tamil Nadu, India*

[2]*Computer Science and Engineering, Manonmanium Sundarnar University, Tirunelveli - 627 012 Tamil Nadu, India*

E-mail: arulchezhiyan96@gmail.com, rs_rajesh1@yahoo.co.in

(Received on 05 July 2012 and accepted on 10 September 2012)

***Abstract* – Mobile ad hoc networks are the ones which allow mobile nodes to spontaneously form a network and share their services. The dynamic environment of MANETs demands service selection should not only based on functional properties but also be driven by non-functional requirements. In this paper we present a modelling method of Non-Functional properties. The degree of impact of the property on QoS may vary. So, we allow the application designer to define weight for each property. The evaluation function for the properties cannot be uniform as the type of the property may be Boolean, string or numeric depending upon the nature of the property. Three different evaluation functions based on the type of the property have been defined. The sum of the weighted metrics is used to select the services.**

Keywords: **Mobile Ad hoc Networks, Service Provisioning, Service Selection**

I. Introduction

Ad hoc networks are distributed networks of mobile nodes without any fixed infrastructure. In these networks the nodes involved have to provide and access services of each other. The device printer can provide a printing service as that of any software service like temperature conversion service. So service means either hardware of software which provides services to others in the network. One of the problems in distributed networks like MANETs is to cope with the dynamic environment.

Service Oriented Architecture (SOA) has been emerged as a solution for distributed systems, and they are suitable especially for loosely coupled system. SOAs enable modularizing the more complex systems in a way that they are composed of independent software components that offer services to one another through well defined interfaces. The advantages [1] of SOAs are Modularity, Interoperability and Extensibility.

There are a number of ways to implement SOA such as, Web Services based on SOAP, GRID Services based on OGSI and REST services based on HTTP and XML. Among them Web service is the predominant implementation of SOA. The advantage of SOA architecture is invocation using late binding (i.e.) binding taking place at the time of execution. One among the major step in SOA implementation is finding a service. In the case of MANETs most of the devices involved are resources constraint, which leads the need of finding only the relevant services. So far there is no automatic selection of services. They need human intervention to select one among the choices of services by different providers.

Match making is done only on the functional properties of the service. In the case of mobile users they need services that are relevant to their current situation. They prefer services which are nearer to them with up-to-date information [4] like news, railway enquiry reply. This mandates the consideration of the non-functional properties such as context properties which specify the current situation of the user will be the suitable one to achieve better performance and user satisfaction.

Service selection becomes a complex task if we need to consider many functional and non-functional properties. The issues of concern in service selection are:

1. How to specify service requirements.
2. How to evaluate the services provided based on the specified requirements and brings out single aggregated value.

Non Functional requirements of a service denote all the aspects which can be used by clients in order to evaluate service quality [11], they play an important role to differentiate among services of same functionality but which differ with respect to the user's current situation [12]. In this paper we propose methods to formalize and send Non Functional properties along with the service functional descriptions. We discussed methods of analysing and matching the attributes to select services.

II. Related Works

Non-functional properties are used a filtering mechanism to find a best match among the choices of services. They increases the rich information provided as a form of pre-requisites for automated service discovery and selection. The integration of these parameters in web services standards [8] has been investigated in order to improve the specification. Ran [3] provides a large number of non-functional properties and organizes them in to several categories, but how they quantified is not elaborated. The work in [6] provides evaluation and proposes QoS based selection, but the sources of the criteria are not mentioned. Konark [7] uses interface of string based matching. Non-functional features such as security transactionality and reliability are considered [2] and developed a framework using WS-Policy which supports transactionality and reliability. The key difference between the other work is we assign values based on the data type supported and each type of attribute is evaluated differently thus gives a general evaluation scheme.

A. Service Architecture

In order to specify the services we can categories the properties as functional and non-functional properties based on the technical information and others. Examples of functional properties are input, output, operations provided, how to access these services etc. Non-functional properties are the one which specifies the quality of the services which are similar to "adjectives". These can be used to define the quality of the service as well as goodness of the services for example the price, performance; bandwidth the consumption etc. of the service. The QoS service incorporates requirements of the consumer, provider, and the network participants [9]. On the one hand the provider can use these properties to specify the services' quality. On the other hand the client/ requester uses them to specify their constraints.

Service providers describe their services and advertise them. These service descriptions will be received by the nodes. They will store them in the repository and used according to the service discovery architecture. The request will be formed by the requester by using the same schema which is used to define by the provider. The application designer will assign values for the attributes based on the need of the application. The requesters current context can be accessed transparently form the user and device profile [10]. The values on the request will be matched against the services' values and ranking will be done in order to select most relevant service.

B. Classification of Non-Functional Properties

Any service can be described by its functional and non-functional properties. Functional properties are those properties which specify about the input, output, interface description, accessing protocols etc. in short we can say that they define the technical aspects of the operations it provide. Non-functional properties are properties which define all aspects which can be used by client in order to specify the service quality.

We classify the Non-Functional Properties as: Quality of Service, and Context properties. The QoS properties are used to specify the service quality that will be provided by the particular service like cost, performance, reliability etc. Each application will have its own set of priorities on the properties. For example Mission-critical applications may prioritize energy efficiency and speedy service response time, where the applications like building automation may prioritize monitoring quality and network utilization. A same set of services' constraints cannot satisfy every application. So we should provide mechanism to specify the weighing factor for each property depending upon the need.

Context Properties are properties that reflect the current context of the application, client and the service provider. Context captures the dynamic nature of the problem environment in way suitable for processing. Context is information that can be used to characterise the situation of an entity. An entity may be place, person, object that is considered relevant to the interaction between the user and an application including the user and the application themselves [13]. Context awareness in service discovery demands the use of implicit information related to the requesting user' constraints, and provider specification, which can affect the usefulness of the returned results [5]. They can be further grouped into (static) domain specific and dynamic. The domain specific properties are the one which may vary for each application domain. In a particular application we may need color printing, at some application we may need laser printing instead of dot-matrix printing. These can be captured in the design time itself, and thus can be defined as static context properties. And thus the application specific properties will have different values for different application context. But these properties values can be captured at the time of application design time and can be described at the time of service description itself. Dynamic properties are the one for which the values can be captured only at runtime such as battery power; load of the server, moving sped of the node

etc. These contexts which may vary even after the application starts have to be captured at the execution time.

C. Modelling Non Functional Properties

Modelling the non functional properties means representing the NFPs of services using any language or using some structured way. The model defined will be used by both the service providers to describe their service qualities and consumers to specify their constraints in a convenient way. The desirable criteria of the model are:

(i) It should support for new addition of properties easily as the developers cannot predict all kinds of NFPs at the time of creation itself and can facilitate the clients to specify the requirements based on the application domain.

(ii) Since the data type of each property may vary the evaluation function should take into consideration of this and the sorting based on this model must be generic even if we add new properties.

(iii) The user may be in different situation while invoking a service. The model should facilitate the user to specify their preferences for each of the NFPs depending upon the situation.

(iv) The Model must be usable for both client and service provider to express their needs and offers respectively. Otherwise they have to be translated into a common language.

(v) The functional and non-functional description of services should be systematically separated, so that the changing requirements can be specified easily.

(vi) The description of the service should not be restricted to a single application domains, it should be generally applicable to all the application domains.

There are many approaches available based on semantic web technology. WSDL-S [15] and OWL-S [14] are used to describe NFP. These approaches mandate the availability of semantic web standards. The other ways to represent NFPs are by extending the UDDI information.

We define schema for the service representation. We assume that the same schema will be used by the client to specify their requirements. The requirement for a service may vary from application to application. For example if the application needs a financial service then the security is having highest importance where as if the application needs a printing service for which low cost is preferable than security. Similarly it may depends on the user also, i.e. one user may want good quality but she/he may not bother about cost, some user may need less cost as highest preferable one than the quality. So we need to allow the application to specify its own requirements. In order to facilitate this, we assign every operation of a service to a category, for example printing service. Each category will have a set of properties assigned to it. Each property is defined as a set of four values.

{<name>, < type>, <weight>, <value>}.

Every property will be of different data types. For example the property color in the case of printer may have yes/no i.e., Boolean value, where the mechanism should be one among the following laser, dot-matrix, ink-jet.

Each property will have its own impact factor if available to an application. For example at some time quality will be more important than cost, at some time the otherwise. So, we provide facility to assign weight to each of the property from the following weighing table. Weighing table we use gives the user the freedom of specifying one among ten preferences. The weighing table may be changed if we need. This will not affect the process of selection as for as the condition .

$$\sum_{i=1}^{n} |wi| = 1$$ is satisfied.

The preferences can be specified as weights. Based on the weighting value we differentiate the properties as Mandatory and Secondary properties. The mandatory properties will have weight as 1, which is the maximum value for weight. It means the satisfaction of this property is essential, failing which the service cannot be utilized. So this is used to filter the services. The value can be either positive or negative. Positive value represents the higher value in this attribute is preferable, and negative represents the lower value is preferable. For example the property performance should be high where as we need less cost.

The Value attribute may take any value based on the data type of the attribute. It is used to specify the value expected to be for this particular attribute. For example, if the data type of the attribute is Boolean, then it can have either yes or no to be matched, e.g, the property color will have yes/no as its value; if yes we need color printing service, Where as if the data type is numeric then the value should be in number.

The value of the Boolean and string type are used to match with the value specified in the service's properties exactly. But the value specified in numeric type is used somewhat differently. Consider this situation where we need less cost for the printing service. If we specify value 10 to the value it does not mean that we don't want printing service lesser than that price. If lesser than that is available then we prefer that, but the cost should not exceed what we specified. So, for numeric evaluation we find the services which provides lesser value if the weight is negative. So, we compare all the selected services and find the position in which every service's value belongs to among the availability. The value property of the numeric data type is used as filter, i.e if the weight is -.6 or any numeric attribute it means we assign the preference as .6 but the value we expect is lower (as the weight is negative), and if the value attribute is specified or example value is 20, then we filter all the service which are higher than this, because we need services which provides lesser or equal to the value specified. While calculating the metric we omit this value. As we defined the schema for the attribute value specification we can allow the user to select the values one among the set of values. This can be done if specify the data type as enumeration. If the number of matching element is high then the service's metric is high, i.e we expect maximum number of entries to be matched. Different data types that can be permitted are limited by XML Schema. In the case of context attributes the address/ method how access these values should be specified in the schema.

1. NFPs in the Provider's Perspective

During service description the provider has to specify the functional and non functional properties of the services. The Non functional properties can be classified as static and dynamic properties. Static properties are the one for which the values can be specified at the time of design itself. For the dynamic properties the values has to be captured at the execution time. For example the performance, memory requirement etc. can be specified at the time of designing. The properties like the availability, response time, distance (number of hops), battery power, moving speed etc. are dynamic in nature. These have to be captured at the execution time only. The dynamic attributes are the context attributes. The context may be service requesters' context constraints and the service providers' context requirements. The service providers context values can be captured and sent at the time of sending the advertisement.

2. NFPs in the Requester's Perspective

Based on the schema defined the requester can specify their requirements. Some properties may be very much essential as for the application is concerned in those cases we can give weight as 1, which is the maximum value for weight. This represents that if this particular property is not matched with the value specified then that service can be filtered out. These non-functional attributes can be considered as mandatory constraints. There may be more than one mandatory property specified in the request. For all the mandatory properties the match should be exact. If there is no match for a single mandatory attribute we will filter that service from list of selected services. The static requirements can be specified during request formation, and the dynamic context attributes can be either specified explicitly or to be accessed either implicitly form the user preferences, device profiles as each device will have its own specification like screen size, input method etc. and vary from one device to the other. These context attributes can be accessed transparently form the user. It can be accessed profiles like device profiles, user profiles. These are the context data that we use in our applications.

D. Assessing the Services

The importance of the attributes for a service may vary for different clients. For example one client may prefer to have speedy reply where as other may be keen on the location of the service. We provide the freedom of specifying importance of an attribute by the application designer itself.

Since we allow different data type for the attributes. The calculation varies slightly for each attribute.

```
<Name> price </Name>
<type>Currency</type>
<weight> -0.6</weight>
<value/>
<Name>color</Name>
<type>Boolean</type>
<weight> 1</weight>
<value>yes</value>
<Name>payment</Name>
<type>Enum</type>
<weight>.6</weight>
<value>"credit card , debit card"</value>
```

Fig. 1 Sample Attribute specification in request message

For Numeric Type : The data types integer, double and currency comes under this. In our example price is of type currency. In the case of data like bandwidth, price etc. Though they are numeric data we need the lower value. In those cases we use the "weight" of the attribute itself specifies that we need the lesser value by specifying a negative value. If the weight of any attribute is less than -1 then it means that the lower value on this is expected.

We calculate the metric of the provider by comparing with the maximum and minimum possible values that can be provided for this attribute in the market. If "Smax" is the maximum value and "Smin" is the minimum value for this attribute the we calculate the metric of a service provider as

For a positive weight
metric is = 1 - (Smax – value) / (Smax – Smin);

For the negative weight
metric is = (Smax – value) / (Smax – Smin);

For Boolean type of attribute we will do the exact match on the value with the service's value.

If there is a match then metric = 1, else 0.

For the enumeration type data the string specified in the "value" will be matched with the service's value.

Metric = (v1+v2+..+vn)/n vi = 1 if a match else 0

For a match we add 1 to and for mismatch we add 0 and the final score will be divided by the number of elements in the set. Here in our example the payment attribute of the service will be matched against the value of the payment in the request. If the service provides "credit card" alone then the value will be (1/2) which is .5, if it matches both credit card and debit card then (1 + 1)/2 i.e. 1 is the credit for this attribute.

Suppose say service "a" provides { color printing, credit card and debit card payment and the price is 5 } and for the example request in fig. We calculate the metric as follows:

For the first attribute "color" which is a Boolean attribute, we find an exact match so the metric for the first attribute is 1.

For the second attribute "payment" the enum attribute, we find the service provides both the requirement of the requester so we have 1 as explained above.

For the third attribute "price" which is a numeric attribute, and the weight specifies that we need lower value for this attribute that is we need a service which prints for lesser price. If the maximum printing price is 20 per page and the minimum is 2 per page then the price we provide satisfies the customer to .833 ((20-5)/(20-2)).

Color metric(m) = 1; payment metric(m) = 1, and price(m) is .83 Like this we will calculate the metric for each attribute specified in the request against the service's values. To find the metric of the service by the provider the metric values will be multiplied by the weight specified assuming that the sum of weight will be equal to 1.

Service selection

During service discovery phase, the services will be compared with the queries. After populating services based on the functional requirements of the user, if there are more than one services for the given request, for the inclusion in the selection list each service should satisfy all the mandatory constraints. If any one of them is not satisfied then the service will be filtered out. Then we have to select the service based on their preferable constraints /secondary constraints. The quality of requirements like service response time, reliability, and availability [3], which are otherwise called as Non-functional properties will be specified along with the query itself.

We categories the Non functional attributes specified in the request as Mandatory and secondary based on their weights specified. Mandatory attributes are used to filter out the services if there is no exact match. The weight for these attributes will be assigned as 1. In the case of numeric type the weight can have 1 and the value attribute to some non-negative value to represent that the attribute must have value less than the value specified in the value attribute of the requirement. For example if the requirement is "the cost must be less than or equal to 50" then the services which are providing the service for more than 50 should be eliminated. Then the requirement can be as { Name="cost" weight=1 value=50 type = numeric}.

Step 1: Filtering the services based on mandatory attributes.

Case 1: Numeric type property with weight is negative and value specified

Select the service if its value it provides is less than or equal to the value specified in the request.

Eg. { Name="cost" weight= -.6 value=50 type = numeric}. The services which provides less or equal to 50 only will be added to the list for further processing.

Case 2: Boolean type with weight 1 (value must be present)

Select the service if its value is exactly equal to the one specified in the request.

Eg. {Name=”security” weight=1 value=”high” type=Boolean}. The services which are providing high security will be added to the list for further processing.

Case 3: Enumeration type weight =1 (value must be specified)

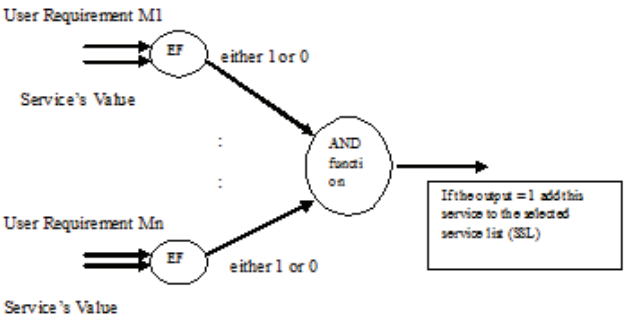
Select the service if all the items in the requirement set matches to the availability, else 0.

Step 2: Evaluate on the weights assigned and order the services.

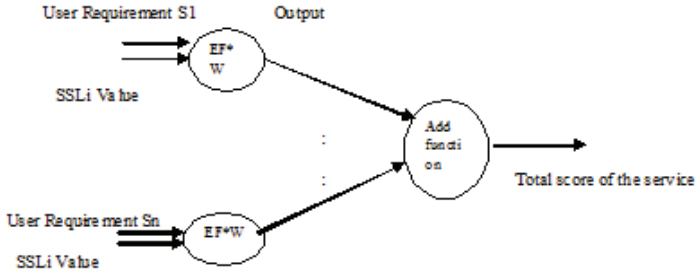
Sum (Mi*Wi) will be calculated for all the secondary properties, and sorted based on the overall value. The fig 3 below elaborates the steps. I will be calculated as discussed in assessing the services.

III. Experiment Results

To understand the feasibility of the aspects, we have implemented the functions. We used 10 parameters with the preference specified. We assumed that 5 same service instances exist and the providers values also specified. We have take two cases where all the parameters have equal weights and the other is assigned with some preference value and the result shows that the specification of weight affects the selection of the services. Table I,II and III are evaluated without considering the weight.



Step1 : Filtering the services based on Mandatory requirements. EF is one of the three functions based on the data type



Step 2: For every service in the SSL calculate the weighted sum. Si is the secondary requirement value we will apply the functions and find their metric that will be multiplied with the weight specified in the requirement.

Fig. 2 Steps in selection

Table I Sample Services

#	Color	Price	Available	Location	Performance	Reliability	Security	resolution	No. in queue	File type
S1	yes	3%	yes	X	.8	.6	High	high	No	Pdf,doc,bmp
S2	No	2%	yes	A,Y	.5	.5	high	high	No	Pdf, doc, bmp
S3	yes	5%	No	Z	.9	.8	medium	medium	Yes	doc, bmp
S4	no	2%	no	X,Y,A	.9	.9	high	high	No	Pdf, doc, bmp
S5	yes	4%	yes	B,A	.5	.6	High	low	Yes	doc, bmp

TABLE II BASED ON PERFORMANCE

#	Value of the service	EF
S1	.8	.75
S2	.5	0
S3	.9	1
S4	.9	1
S5	.5	0

TABLE III BASED ON LOCATION

#	Value of the service	EF
S1	X	.33
S2	A,Y	.67
S3	Z	0
S4	X,Y,A	1
S5	B,A	.33

TABLE IV BASED ON QUEUE

#	Value of the service	EF
S1	No	1
S2	No	1
S3	Yes	0
S4	No	1
S5	Yes	0

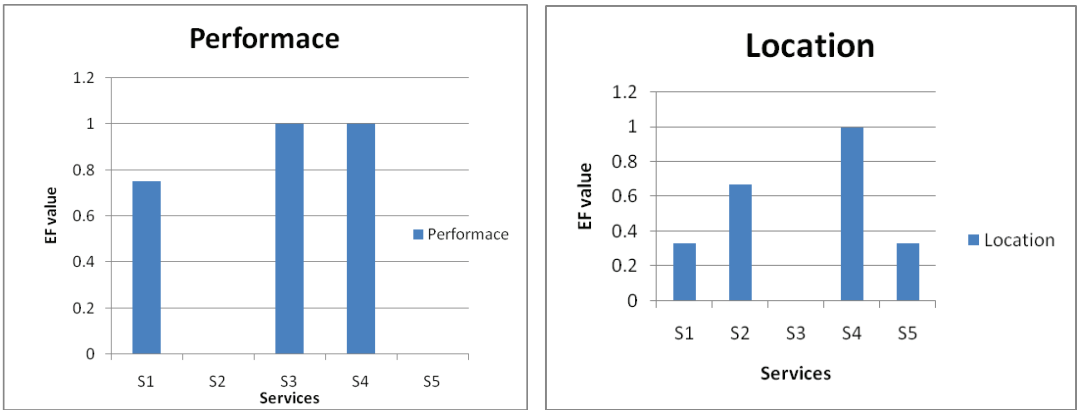


Fig. 3 Considering only the Performance and Location

TABLE V REPRESENT THE TOTAL VALUE CONSIDERING THE FOLLOWING REQUEST

#	Weight	EF	W*EF	Weight	EF	W*EF	Weight	EF	W*EF	Total
S1	.3	.75	.225	.2	.33	.066	.5	1	.5	.791
S2	.3	0	0	.2	.67	.134	.5	1	.5	.634
S3	.3	1	.3	.2	0	0	.5	0	0	.3
S4	.3	1	.3	.2	1	.2	.5	1	.5	1.0
S5	.3	0	0	.2	.33	.066	.5	0	0	.5

{Name="Queue" weight=.5 value=NO type=Boolean}
{Name="Location" weight=.2 value="'X,Y,Z" type=Enum}
{Name="Performance" weight=.3 type=Numeric}

When we consider only performance S3 and S4 are having equal values, where as considering location alone S4 is at the top and S2 comes next. When we consider all the requirements with varying preferences weights the value will be different. The calculation is given in table V.

From Figure 4 we can say that S4 is the service on the top then, S1, S2, S5 and finally S4. So, if we specify the request with the preferences we can select the appropriate service for our need.

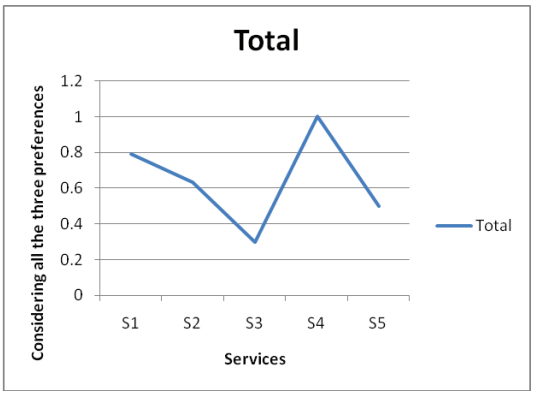


Fig. 4 Considering all the needed preferences with varying weights

IV. Conclusion and Future Work

Due to the increased popularity of web service technology and the availability of may providers for a same service increase. The consumers are therefore concerned about finding services that are relevant to the current context. This paper proposed and exemplifies the influence of the non functional properties in service selection. The model proposed can be used for capturing all the non functional properties, adding new one if need arises without making any major work. Though the context properties can also be captured in this stage itself, we elaborate the method of accessing and deriving the values from the existing attribute values. Our future study is focused on capturing methods and representation of dynamic properties.

References

[1] R.T. Fielding, and R.N. Taylor, "Principled design of the Modern Web architecture", *ACM Trans. Inter. Tech.,* Vol.2 No.2, May 2002, pp.115-150.

[2] N.K. Mukhi and P. Plebani, "Supporting Policy driven behaviours in web services: experiences and issues", in *proceedings 2nd International conference on Service Oriented Computing* ICSOC'04, NewYork, USA, 2004.

[3] S. Ran, "A model for web services discovery with QoS", *ACM SIGecom Exchanges,* Vol. 4, No.1, pp. 1-10, Spring 2003.

[4] C. Doulkeridis and M.Vazirgiannis, " A System Architecture or context-aware service discovery", 2005.

[5] C.Doulkeridis and M.Vazirgiannis, "Querying and updating a context aware service discovery in mobile environments", *In Proc. 4th VLDB Workshop on Technologies on E-services* (TES'03), 2003.

[6] Y. Mou, J. Cao, S.S Zhang, Interactive web services choice-making based on extended QoS Model, CIT 2005, pp. 1130-1134.

[7] C. Lee A. Helal, N.Desai, V. Verma and Arslan B., Konark, "A system and protocols for device independent, Peer-to-Peer discovery and delivery of Mobile services", *IEEE Transactions on systems, Man and Cybernetics,* Vol. 33, No. 6, November 2003.

[8] M. Comerio, F.D. Paoli, A. Maurino, and M.Palmonari, "NFP-Aware Semantic web services selection", *in 11th IEEE International Enterprise Distributed Object Computing Conference,* 2007, pp 484-491.

[9] Chien-Liang Fok, Christine julien, Gruia-Catalin Roman, and Chenyang Lu, Challenges of satisfying multiple stakeholders: Quality of Service in the internet of Things, SESENA'11, May 22,2011, Waikiki, Honolulu,HI,USA.

[10] K.Ponmozi and R.S. Rajesh, "Bringing context awareness into MANETs" in the *National on Explorations & iNnovations in Advanced Computing,* National Engineering College, Kovilpatti, India,2008,pp.138-144

[11] G. Dobson, R.Lock, and I.Sommerville, QosOnt: A QoS Ontology for service-centric ystems, *In 31st EUROMICRO Conference on Software engineering and Advanced Applications,* pp. 80-87, 2005.

[12] Kristof Hmann, Sebastian Steenbuck, and Sonja Zaplata, Towards NFC-aware Process Execution for Dynamic Environments, Proc. WowKiVS 2011,EASST Vol 37(2011)

[13] Anind K. Dey, "Undertanding and using context, Future computing environments group", college of computing & GVU center, Georgia Institute of Technology, USA, 2000.

[14] Amigo Consortium, Detailed design of the amigo middleware core, Project Deliverable D3.lb. (2005).

[15] S.Ben Mokhtar, A.Kaul, N. Gerogantas, V. Issarny, "Efficient Semantic service discovery in pervasive computing environments," in: *Proceedings of ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06),* 2006.