An Efficient Web Service Discovery Architecture for Static and Mobile Environments

Shrabani Mallick, Dharmender Singh Kushwaha, *MNNIT ALLAHABAD, India*

***Abstract*—The widely adopted and implemented core web services standards SOAP and WSDL have achieved extraordinary interoperability across highly disparate software systems. The service oriented architecture SOA has become widely recognized for its important role in information technology (IT) projects. A SOA is a style of design that guides an organization during all aspects of creating and using business services (including conception, modeling, design, development, deployment and management). SOA has been the ideal combination of architecture and technology for consistently delivering robust, reusable services that support today's business needs and that can be easily adapted to satisfy changing business requirements. As systems become more complex, the overall system structure-or architecture---becomes a central design problem. A system's architecture provides a model of the system that suppresses implementation detail. Unfortunately, current representations of SOA architecture are informal and ad hoc. Currently many state of the art formal methods have been applied into the modeling, interoperability, dependability and trustworthiness of web services and this could have a significant impact on the ongoing standardization efforts for services and cloud technologies. This paper presents a formal verification of proposed x-SOA based architecture for UDDI based web service discovery framework. The paper attempts to establish the proposed architecture for locating services in mobile computing environment as well. Potentially, extending the state of art formal method techniques could have a significant impact on the ongoing standardization efforts for web services and cloud technologies for both fixed and mobile networks.**

***Index Terms*—SOA, mobile networks, cache, broker, service discovery**

I. INTRODUCTION

PROGRAMS that interact with one another over the web must be able to find one another, discover information allowing them to interconnect that may include request/reply interaction or more complicated process flow such as negotiate qualities of service as security, reliable messaging and transactional composition. The web service community is working to meet all these requirements. A web service is a location on a network that has a machine-readable description of the messages it receives and optionally returns. Standards like XML, WSDL, SOAP, UDDI [1,2,10] define how web services are described, discovered, and communicate with one another. Web services roles include requester and provider.

Shrabani Mallick is with the Computer Science and engineering Department, Motilal National Institute of Technology, Allahabad, India (email: shrabani@mnnit.ac.in)

Dharmender Singh Kushwaha is with the Computer Science and engineering Department, Motilal National Institute of Technology, Allahabad, India (email: dsk@mnnit.ac.in)

The service requester initiates the execution of a service by sending a message to service provider. The service provider executes the service upon receipt of a message and returns the results, if any are specified, to the requester. A requester can be provider, and vice versa, meaning an execution agent can play either or both roles. A service is therefore defined in terms of the message exchange patterns it supports. The main components of service includes the description, the implementation and the mapping layer between the two.

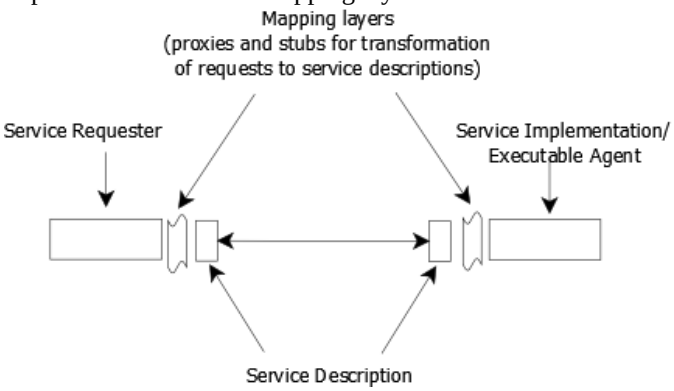


Figure 1: Components of a web service

The major advantage of implementing SOA using web services are that they are pervasive, simple, discoverable and platform neutral. Businesses use SOAP to register themselves or others with UDDI; then the registry clients use the query APIs to search registered information to discover a trading partner, figure 2.

Figure 2 shows the basic 3-tier SOA architecture which consists of specifications (SOAP, WSDL and UDDI)[10] that support the interaction of a web service requester with a web service provider and the potential discovery of the web service description. For the success of a SOA based application, it's important that users are motivated to use them, for which all that matters is that they can understand and formulate XML-formatted query messages.

DOI: 10.5176/2251-3043_3.1.243

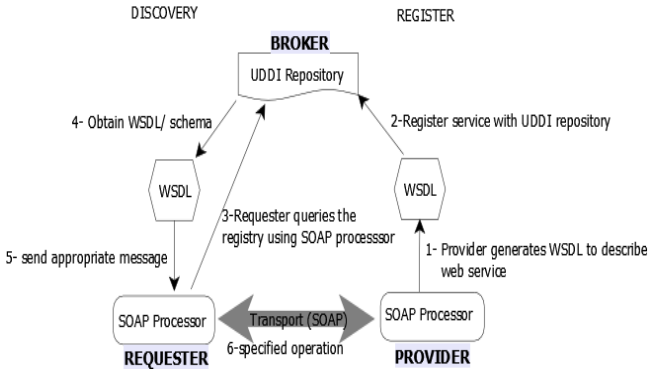
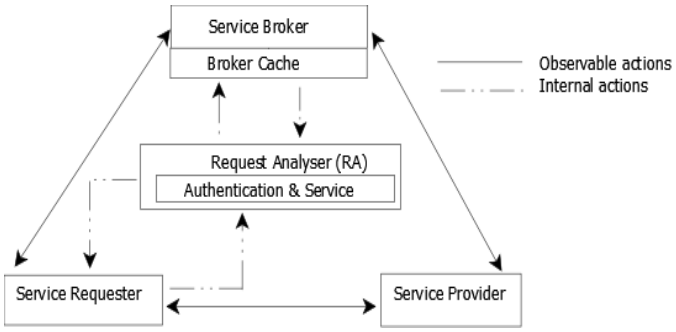


Figure 2: The web service registration and discovery

Under such circumstances companies may struggle to adopt to SOA[4] as users have to be first trained to standards, conventions and data structures for querying of web services. The next section discusses the *modified x-SOA* architecture that provides an enhancement over the basic x-SOA architecture [4]. The modified x-SOA [11] architecture is a better answer in dealing with the potential problems of adapting to SOA and web service standards. The new layer of abstraction called the Request Analyzer provides extended facilities to the Service Requester thereby facilitating a naive user to place the request query in plain text English.



x-SOA Architecture for web service discovery

The proposed x-SOA architectural framework can provide an effective and universal service querying mechanism for any level of users. The new layer RA sits on top of a service requester that facilitates the preprocessing activities of a request query before it is being handed over to the broker for discovery.

II. RELATED WORKS

With the increased need of web services and equating them to software as a service has motivated many works in this field. [1,2] offers an alternative to the UDDI registry for discovery of web services. The authors in [3] proposes a FSM approach for Dynamic web Service. [4] a three layer Service brokering x-SOA architecture for an underlying transparent web service access to the service client is presented. In [5] semantics based request query analysis using a tree-form of data structure to discover the web services by assigning weight values to each node of the tree is discussed. In [6] a keyword based approach is implemented triggered by the partitioning approach that is used in database design. In [7,8] a web service discovery model, based on abstract and lightweight semantic web services descriptions, using the Service taxonomy is focused. Authors in [9] propose an approach for semantic web service discovery and propagation based on semantic web services and FIPA multi agents. Authors in [11] propose an efficient discovery architecture using an additional tier of request analyzer. Based on the works of Authors in [13, 14, 15] we conclude that an efficient way to reduce the network factor of energy saving or consumption is through disconnects, as and when going to inactive state or essentially not using the network. The authors in [16] have estimated the energy costs for mobile and ubiquitous computing.

The architecture proposed so far has not been formalized. Most of the work done by the previous researchers have tried to address the problem of web service discovery problem through approaches based on key words, semantics, neural networks, ordered service search, agent based framework etc. Many of them assume that the user is aware of the web service interfaces names, which is not always true. An organized framework which contains a layer exclusively for processing a simple plain text query based on their linguistic compositional semantics and getting back the most appropriate set of web services is what is focused here. More over this paper has attempted to formalize the proposed architecture so that SOA framework can be standardized. The proposed architecture has been extended for mobile environments which are resource constrained and has shown satisfying results.

III. TRANSITIONAL DISCOVERY SEMANTICS

Next, we shall depict the formalised web service discovery approach using the proposed architecture and the states and interactions of various entities involved. It can be thought of as message passing communication system that consists of request-reply (reply in this context is web service interfaces) paradigm.

Request (Req) : A plain text request query for obtaining web services that is analyzed for its linguistic components.

Service (Req) : The requester on receiving the suitable web service interface names invokes the required web service from the provider.

The Request Analyzer (RA) shall act as an intermediary between the service broker and the service requester, figure 3. The RA accepts the request from the Requester and parses the request query string into the lexical components- verbs, noun, adjectives and adverbs. The RA will first forward the verbs queue to the service broker where two possibilities may occur: Look up its cache, upon which if matching web services are found, the broker will return the web service names. If no matches are found, the broker broadcasts the request on the cloud, using the publish/subscribe paradigm. On getting back the list of web service names (WSNs) from the brokering agent, the RA will compare the WSNs with nouns and adjectives+adverbs, that will serve as the parameters for context based (nouns define contexts) and use-based matching (verbs define use whereas adjectives & adverbs refine the

use). After comparison the most appropriate web service name/(s) will be returned back to the Requester.

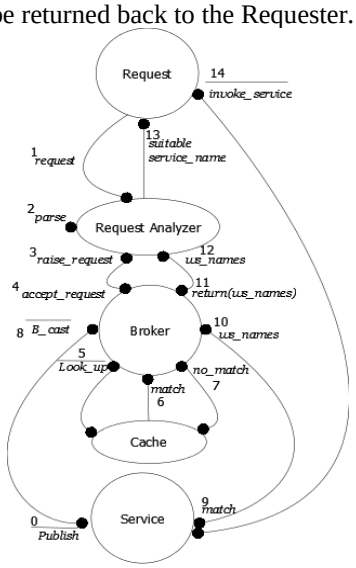


Figure 3 : Flow Diagram of web service discovery algorithm

The Requester then invokes the interface through the provider. The Broker will either look up or broadcast. The operational semantics for the Look_up cache activity will be a pattern matching with the actual request query which is depicted as–

A template T and request query r match when the following function is defined as

$$Look\_Up \triangleq (wsn) \leftarrow \emptyset, @\ m\ (T, r) \quad (1)$$

Where

$$m\ (T, r) = \begin{cases} True & if\ (\forall t)s[< t, t' >]@r \\ False & if\ (\forall f)s[< f, f' >]@r \end{cases} \quad (2)$$

The Look_Up action will return the web service names for which the match $m\ (T, r)$ will return true or $\emptyset$ if no matches are found.

If no matches are found then it will broadcast the request query on the cloud to which the interested service providers may respond and return their addresses.

$$\overline{BCast} \triangleq \uparrow addr(N), (r) \quad (3)$$

$addr(N)$ collects those names that effectively occur in a network N as address of some provider.

$$addr(N) = \begin{cases} \emptyset \\ addr(N_1) \cup addr(N_2), N = N_1 || N_1 \end{cases} \quad (4)$$

On getting back the list of web services the request analyser filters the suitable service names using the nouns and adjectives from the parsed request query which is defined as ,

$$\overline{Accept}(N) \triangleq Test\ N\ p\ r \quad (5)$$

Once the requester gets the required service name and address, it invokes the service by sending appropriate message to the provider using RPC as,

$$\overline{Invoke} \triangleq (S), N, < b, s > \quad (6)$$

where b and s are the binding and status parameters.

IV. TRANSIITONAL SEMANTICS OF THE DISCOVERY PROCESS

Now, we shall define the transition of each agent in terms of their component and input agents. For the Transition System, let A the set finite alphabet contains the symbols r, v, n, a, s that stands for request, verb, noun, adverb/adjective or service respectively. Let τ be the symbol of unobservable or internal action, S_0 be the initial state, S', S" the intermediate states and S the final state.

$$S_0 \triangleq Raise\_req.(r).S'$$

$$S' \triangleq Look\_Up.(r).S'' + BCast.(r).S''$$

$$S' \triangleq Invoke.(N).S$$

The complete set of transitions are given below:

Rule 1: Parse

For parsing the plain text request query, the initial transition can be written as,

$$r.S_0 \xrightarrow{r} S'$$

Now after parsing for a v component

$v.S_0 \xrightarrow{v} S'$, similarly for a,n components $a.S_0 \xrightarrow{a} S''$ and $n.S_0 \xrightarrow{n} S''$

Rule 2: Look_up

$v.S' \xrightarrow{v} S'$ if no match is found

$v.S' \xrightarrow{v} s\ S_f$ if match found

Rule 3: Sum

Let there be n services i.e, I={1,2,3...n} that have matched with the request query. The summation rule can be written as,

$$\sum s.S_i \xrightarrow{a} S_f \quad \text{where } i \in I$$

V. PERFORMANCE EVALUATION

The proposed modified x-SOA architecture using the Request Analyser off-loads the requester from being over-whelmed with the huge list of web services and hence makes the selection of web services easier and faster.

We started with a list of 10 web services. On subsequent refinements the reduction in search spaces are shown in the graph below:

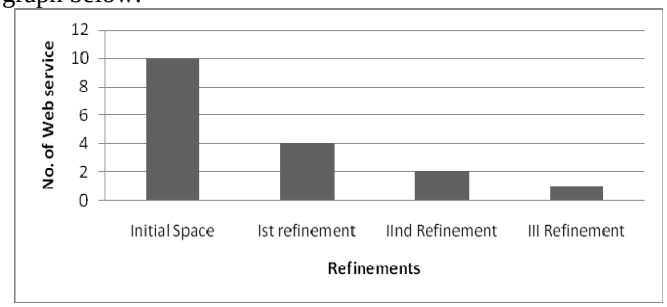


Figure 4 : Plot of various stages of Refinement against the no. of web services

Hence the strategy follows a combinatorial search algorithm to achieve efficiency by reducing the effective size of the search space.

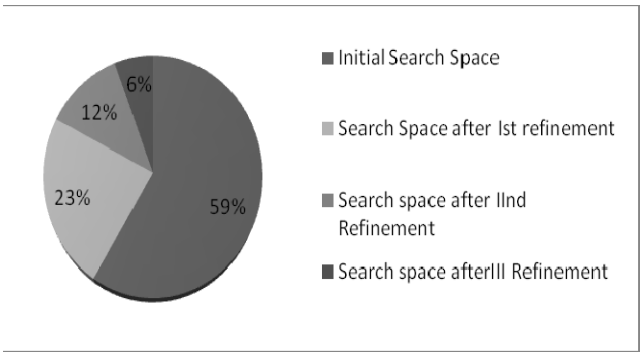


Figure 5: Proportion of search spaces for various refinements

For the above example we see that address space (based on umber of web services) has reduced from n (=10) to log(n) (=1). The running search times tested on a 1-billion-steps-per-second computer for different size of web service sets returned are as follows:

Size of data set (n)	Search Time
1	**0.001 µs**
10	**0.01 µs**
20	**0.02 µs**
30	**0.03 µs**
40	**0.04 µs**
50	**0.05 µs**

Table 5: Search times in µs for web services

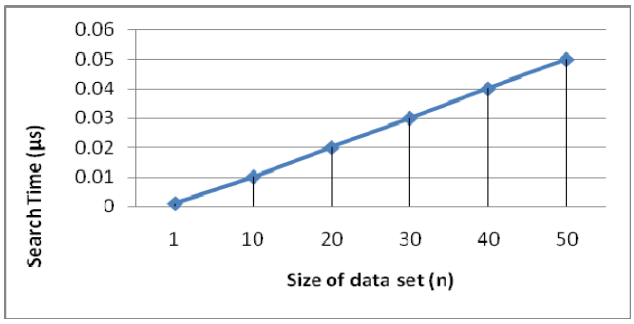


Figure 6: Plot of Search times for different sizes of returned data sets.

The plot shows that the running search would reduce considerably with the reduced and pertinent set of web services.

VI. THE MODIFIED X-SOA ARCHITECTURE AND MOBILE ADOPTION

Web services provide a convenient method for messaging over the internet infrastructure. For example, a given system may allow mobile users to find someone's phone number through a directory service and subsequently get driving directions to that person's place of residence. These two functions may be available as web services. This can be done in two ways-

(i) By using Web Service Proxy (Asynchronous mode)

(ii)By using Direct Connection to the Web Services (Synchronous mode)

In the first case the back end of our mobile system may connect to the web servers and retrieve the information and return it to the mobile device through whatever communication protocol it is using when network connection is fair. In the second case, the mobile devices can directly access the network and use the web services. The latter case requires a considerable advanced mobile device with efficient XML parsing capability. The prime issues with mobile devices are –

1. Energy constraints
2. Resource constraints – Processor and Memory
3. High mobility hence network instability

The Asynchronous mode scores more as a better strategy for finding the web services as it presents a perfect way to deal with a disconnected user in an active way in case of intermittent network connectivity. But the other two problems remain partly unaddressed. The proposed x-SOA architecture serves as a solution for the canonical mobile environment. When the mobile device initiates a web service discovery request through a plain text English query either via a proxy or an agent the request is served with a huge set of matching web services. The job of the web service proxy or the agent is to get back to the query initiator with the results whenever the device gets connected. The huge set of results returned may lead the constrained device in a starvation state while it has to select the appropriate web services to invoke them. The proposed Request Analyser may act as a middleware service in the proxy or agent to filter the appropriate service so that the mobile user is served with an optimized set of service. The Broker Cache may help the device to connect to services faster in case of repeated use of same services.

The X-SOA architecture addresses the following aspects in consideration to various dimensions of resource starved devices-

1. Reduced no. of services returned thereby reducing the memory utility.
2. Selected set of services – optimizing device's processor performance
3. Less Processing – saving energy
4. Asynchronous mode of working – solving intermittent connectivity.
5. Caching in Broker – Reducing discovery time for repeated use of services.

The working of the modified x-SOA architecture in respect of the mobile computing environment for finding the needed services is depicted in figure 7.

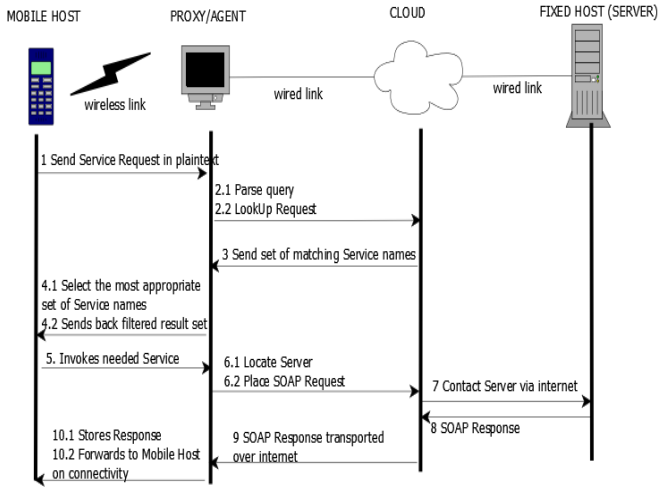


Figure 7: Locating needed services with modified x-SOA in mobile computing environment

The set of actions performed by the mobile host and the proxy/ agent as per the architecture implementation can be stated as –

1 On NewServiceRequest(httpRequest)
2 startInterface() with proxy
3 send plaintext request query
4 suspendInterface()

5 On ReceiptServiceNameFromProxy(httpResponse)
6 if (ServiceName>1)
7 select the required service
8 send invoke request to proxy
9 else
10 send invoke request to proxy
11 suspendInterface()

12 On ServiceResponseFromProxy(httpResponse)
13 use service functionality
12 suspendInterface()

Figure 8.1 Actions performed in the mobile host side

13 On NewSerchRequested(httpRequest)
14 parse query
15 send httpRequest for LookUp in UDDI
16 receive the list of service names
17 filter names based on semi-semantics of request query
18 send (ServiceNames) to mobile Host
19 On InvokeRequest(soapRequest)
20 Locate server
21 send (soapRequest) to server
22 receive (soapResponse) from server
23 cache service
24 Store Result
25 forward Result to mobile Host on resume Interface

Figure 8.2 Actions performed on Proxy side

VII. Performance Estimates Characterization for Mobile Environments

According to [16], the system energy cost for establishing a connection and transferring n bytes based on the volume of the data is as given:

$$E = Ee + n * Et \quad (1)$$

Where Ee is the energy cost for connection establishment and Et is the energy per bit for the transfer.

Based on experimental results the plot of various levels of energy consumed while transmitting varying chunks of data packets is shown in figure 9. Ee is experimentally taken as 5J.

Time	Data Transfer (Bytes)	Energy consumed(J)
9.00	1656	40
9.15	896	520
9.30	2561	650
10.15	785	458
10.30	696	280
11.12	1759	490
12.11	125	120
1.25	1899	460
1.31	132	450
2.20	145	136
2.45	2601	625

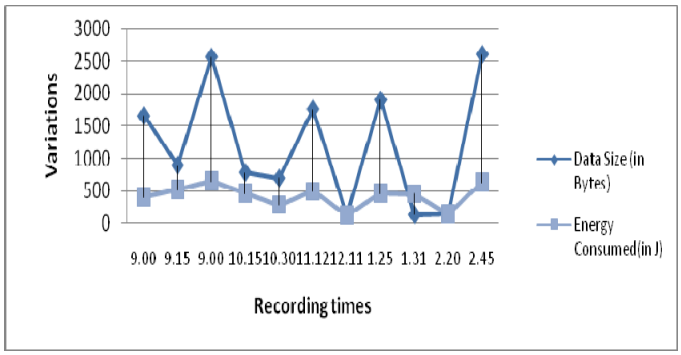


Figure 9: Plot of volume of data transfer and corresponding energy consumption

Thus it can be concluded from the graph that reduced volume of data transfer saves on the battery life substantially. The proposed modified x-SOA architecture scales well with other the mobile environment and exhibits the following performance estimators:

- Reduced Volume of computation
- Phases of connected and disconnected operations
- Less Memory consumption by storing selected services
- Optimized processing for searching from a reduced search space
- Faster service discovery time through broker caching.

VIII. Conclusion

We have formally defined a formal calculus for SOA framework that provides basic primitives to describe the SOA taxonomy. We demonstrated the efficiency of the architecture using a sample case study. This paper presents a formal verification of proposed x-SOA based architecture for UDDI based web service discovery framework. Potentially, extending the state of art formal method techniques could have a significant impact on the ongoing standardization efforts for web services and cloud technologies. The proposed formalized architecture can be a driving force in design of middleware for SOA application. The architecture has been established o be adopted to mobile environment as well. The performance based on the experiments have been found convincing.

References

[1] xMETHOD,[Online].Available. ttp://xmethods.net/ve2/index.po

[2] REMOTEMETHODS,[Online]

[3] San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, Cheng-Hung Chen, On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection 2007 IEEE International conference on Web Services

[4] Mike P. Papazoglou, Willem-Jan van den Heuvel Service Oriented Architectures: approaches, technologies and research Issues, The VLDB Journal (2007), Springer-Verlag Publication

[5] Wuling Ren, Zhujun Xu, A New Web Service Discovery Method Based on Semantic, IEEE 2008 Workshop on Power Electronics and Intelligent Transportation System

[6] Jiangang Ma, Yanchun Zhang, Jing He, Efficiently finding Web Services Using a Clustering Semantic Approach, CSSSIA 2008, Copyright ACM

[7] Georgios Meditskos and Nick Bassiliades, Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S, IEEE Journal Publication, Apr 2009

[8] G. Meredith and S. Bjorg. Service-Oriented Computing: Contracts and types. Communications of ACM, 46(10):41-47, October 2003.

[9] Azadeh Ghari Neiat, Mehran Mohsenzadeh, Sajjad Haj Shavalady, Amir Masoud Rahmani, A new approach for Semantic Web Services Discovery and Propagation based on Agents, April 2009 IEEE International Conference on Networking and Services.

[10] WSDL 1.1: http://www.w3/TR/wsdl

[11] Shrabani Mallick, D. S Kushwaha " An Efficient Web service Discovery Architecture", International Journal of Computer Applications, Volume 3- No 12 , July 2010

[12] Robin Milner , Communication and Concurrency, Prentice Hall, 1989

[13] Giuseppe Anastasi, Marco Conti, Enrico Gregori, and Andrea Passarella, "A Power Saving Architecture for Web Access from Mobile Computers", Springer-Verilag, NETWORKING, LNCS 2345, pp. 240-251, 2002.

[14] M.Stemm e R.H.Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices", Proc. 3° International Workshop on Mobile Multimedia Communication, Princeton, NJ, Settembre 1996.

[15] G. Anastasi, M. Conti, W. Lapenna, "Power Saving Policies for Wireless Access to TCP/IP Networks", Proceedings of the 8-th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks IFIP ATM&IP2000), Ilkley (UK), July 17-19, 2000.

[16] Ahmad Rahmati, Student Member, and Lin Zhong IEEE member, "Context-Based Network Estimation for Energy-Efficient Ubiquitous Wireless Connectivity" IEEE TRANSACTIONS ON MOBILE COMPUTING, *{submitted}*

Authors:

Shrabani Mallick is with the Computer Science and engineering Department, Motilal National Institute of Technology, Allahabad, India (email: shrabani@mnnit.ac.in)

Dharmender Singh Kushwaha is with the Computer Science and engineering Department, Motilal National Institute of Technology, Allahabad, India (email: dsk@mnnit.ac.in)