Journal of computer Science 2 (8): 627-633, 2006
ISSN 1549-3636
© 2006 Science Publications

Some Enabling Technologies for Ubiquitous Networks

[1]Junaid Ahsenali Chaudhry, [2]Chaudhry Sajjad Hussain and [1]Seungkyu Park
[1]Graduate School of Information and Communications, [2]Department of Electrical and Computer Engineering, Ajou University, South Korea

Abstract: Every new technology that makes its way in the market is supported by a collection of core components called enabling technologies. These components play a major role in making that new technology a success. In this paper we discuss Autonomic Computing (AC) and especially self management and Service Discovery (SD) as the enabling technologies for ubiquitous networks. The AC puts the 'visible' automation into ubiquitous networks an element which has drawn huge interest both among the public and industry. The envisioned benefits of AC are very promising but its applications are at their inaugural stages. Service discovery is well known for its significance in ubiquitous environments. However practical manifestations of environments such as futuristic smart homes and ubiquitous kitchens remains to be seen as a success. We discuss our research findings and enlist the importance of the two enabling technologies mentioned above.

Key words: Enabling technologies, Autonomic Computing, Service Discovery, ubiquitous networks

INTRODUCTION

Ubiquitous services make use of service sectors both in government and private enterprise arenas. Healthcare, security in terms of personal safety and community sabotage prevention and management and smart homes all are the envisioned applications of ubiquitous environments[1]. Service discovery techniques allow users to discover services, applications and devices that are available in the network and may also facilitate service access and use. It is useful e.g., for two scenarios. One; mobile users in foreign networks or groups of users that form a spontaneous (ad hoc) wireless network poised to share services, two; the services are temporally changing according to e.g., the policy of a corporation to restrict or extend services according to the user's role and the time of the day. A user can trace a service by requesting for a particular service type (e.g. printer service) and may make an intelligent service selection in case multiple services of the desired type are available. A sizeable reduction in administrative load to characterize, advertise and bind services through manual means, by automating the entire procedure the total cost of ownership[2] is made bearable for network operators and service providers.

Building on top of the ubiquitous infrastructure i.e. OSGi[3], the Service Oriented Platform (SOP) is considered to be strong candidate for future ubiquitous networks[4]. IBM being the leader in taking the initiative in Autonomic Computing has worked for making their products more dependable. Tivoli, Lotus, IBM WebSphere and IBM DB2 being the prime examples in software. In hardware IBM eServer[5] is among prime examples. Combining the abilities of both AC and SD technologies will provide ubiquitous networks with the 'visible automation' and 'seamless task customization'. The examples of both are shown in the scenarios below.

Scenario 1: A news reporter is visiting a news room which is established in a convention hall where an international conference is in progress. There are several services available in the news room, such as a registration service, interpretation service for non-English speakers, show-of-hands service for raising questions and a number of computing services such as printing, fax services and internet connectivity. A news reporter, we assume inexperienced in computer dexterity, can be provided seamless access to all such value-added services, only if such an environment provides an autonomic service discovery and invocation mechanism.

Scenario 2: A commuter wishes to travel from place A to B. If we wish to add this scenario to the ubiquitous world we have to consider at least the following constraints. How to choose the route, decide the stops for rest for the commuter, refilling and maintains needs for the vehicle. All this has to be decided through some distributed service or set of distributed services. As the commuter moves and changes positions, new and updated services are added to the itinerary. This process

Corresponding Author: Junaid Ahsenali Chaudhry, Graduate School of Information and Communications, Ajou University, South Korea

of update in itinerary carries on until the commuter reaches the destination.

It is apparent from the scenarios described above that in a ubiquitous application there are many technologies that make a seamless experience possible. Discussing the two enabling technologies in this paper will give us the insight to realize their importance in ubiquitous computing.

Related work: Services discovery is primarily based on publicize and listen-in methods. Either, node or service or both of these are notified through advertisements, seen as an optimization problem. Standard bodies such as IETF and consortia of business giants are addressing it at various levels in different ways. A variety of service discovery protocols are currently under development. Here, we summarize widely known schemes from the perspective of their applicability and acceptance.

Jini[6] is an extension of the programming language Java that allows platform independence through Java Virtual Machine (JVM) environment but offers limited applicability for smart environments, which are characterized by heterogeneous device capabilities, mostly lacking support for JVM. Scoping is done through definition of groups; a concept difficult to realize in the wake of device and user mobility in smart spaces. Service Discovery Protocol[7] is a concerted effort of Microsoft and Intel to address device discovery addresses for Bluetooth environments. Poised for discovery, it does not elaborate device or service accessibility and usage procedures. Smart spaces, envisaged to be intelligent enough for use by consumers must go a step beyond what SDP® offers. UPnP[8] extends Microsoft's Plug and Play technology to the scenario where devices are reachable through a TCP/IP network. Designed for IP-based networks, there is a technological conflict between the address-centric nature of UPnP and address-agnostic feature of smart spaces. A framework that mitigates such a limitation of UPnP is still awaited. SLP2[9] Service Location Protocol 2 is the newer version of the initially proposed version of original service location protocol. Proposed by Internet Engineering Task Force (IETF), it is a decentralized, light weight, scalable and extensible protocol for service discovery within a site. Support for form factor constrained devices is provided by the use of proxies and surrogates.

SELFCON is the architecture for self configuration in ubiquitous networks[10]. SELFCON associates the configuration intelligence with the components of network rather then limit it to a centralized management station. AUTONOMIA is an autonomic computing environment[11]. They propose Autonomic Middleware Service (AMS) as the service that takes care of dynamic needs of autonomic applications. The idea of Personal Home Servers is presented in[12]. They believe that personalization is the key to reduce the complexity. To each user, they have provided a server that deals with the service customization and each user communicates with others via his personal server. The mobility of the user and the service portability is also managed by the same. Embedded micro servers are provided to every physical device in[13] called Pervasive Servers. The emphasis is on user customizability and policy enforcement. They say that the personal server carried by each person coordinates with pervasive servers. This framework is very dynamic and may be a good candidate for dynamic service composition. They have used Universal Plug and Play (UPnP) for service discovery, traditional web server for location management and XML messaging protocol for message passing among the individual modules. The context management is difficult in this environment but they define it as their future goal.

Applications and autonomic self management: Despite of a lot of promise, there is no "killer application" identified for ubiquitous computing yet[17]. Many prototypes are developed addressing many aspects but not a single application that can result in widespread adaptation. It has caused lack of acceptance among the consumers. Seamless mobility, fast handover, connectivity everywhere etc. are the kind of facilities that are promised but they can not attract a common user[3].

System requirements: One of the most important applications of autonomic computing is self management. It is a cost effective approach to manage distributed networks. Self managing systems pose some functional requirements. According to our experience, the following system requirements should be considered as the least capability set when developing a smart space.

Anomaly detection: Behavior-based anomaly detection compares a profile of all allowed application behavior to the actual traffic. Any deviation from the profile is flagged as a potential attack. It is commonly referred to as a positive security model because it seeks only to identify all "known good" behaviors and assumes that everything else is bad. Behavior anomaly detection has the potential to detect attacks of all kind – including "unknown" attacks on custom code and routine[18].

Behavior anomaly detection can also lead to a high rate of false positives. For example, after a behavior profile is created, an application developer may change the application (a new URL, new parameter, etc.) without notifying the security team. In this case, behavior-based anomaly detection wrongly identifies access to these new parameters as potential attacks. Given the extreme complexity and dynamics of enterprise Web applications, the use of behavior anomaly detection as the sole basis for blocking attacks in real time is difficult without continuous tuning. That is where autonomic computing comes into play.

Autonomic diagnosis: After detection, comes the diagnosis part. At this stage, many Artificial Intelligence techniques are used in many applications i.e. neural networks, machine learning etc[19]. The modern day Autonomic Diagnosis techniques categorize fault notes with respect to the frequency with which they are occurring. They categorize the most frequently occurring faults into a separate category and diagnosis part is skipped which proves to be the most time consuming and takes the system directly into the recovery phase where actually modular recovery process takes place. This way, a lot of functional redundancy can be avoided and it helps in fault categorization. Although there can be many cases when diagnosis part is necessary, but the frequency of such cases decreases with the age of system.

Context awareness and solution proposition: Context refers to a concept in human-computer interaction in which a system can interpret explicit acts of communication. For example, a food-ordering system could interpret that a user wants a particular menu item based upon which item the user has selected with their mouse. Context aware computing, a paradigm first introduced over a decade ago, extends the idea of context to also refer to the physical and social situations in which computational devices are located. For example, a context aware cellular phone would automatically turn its ringer off if it were aware the user was in a business meeting (by having access to the users meeting schedule). Other such examples of context awareness are a device being aware of its particular physical or relative location, or its own characteristics such as processing power, battery life, input devices, screen size, etc. In a self managed system, context plays the key role. When it comes in terms of telling the manager that 'what exactly is happening?' in the system, it's the context awareness that conveys the information and manager takes action.

All functional parts within the system and perceived parts of environments can be called context for self managed systems because change in one parameter can change the system behavior altogether. Heuristics is a problem-solving technique in which the most appropriate solution is selected using rules. Interfaces using heuristics may perform different actions on different data given the same command. All systems using heuristics are classified as intelligent. Expert systems were a classical example of heuristics based systems. There are many hard coded solutions are provided where all the situations that system can face are coded in if-then statements. In a ubiquitous real time world we need hybrid approach where both hard coded and inferred solutions are applies to the self managed system[3].

Standardization interaction: Software messages between two entities carry decoded messages inside them. Although those messages are received by the receivers but it is very important that the receiver be able to interpret it correctly. Nowadays, standardization of the network is considered to be one of the most important infrastructures to promote information utilization. In a self managed system the standardized communication is very important in order to avoid the ambiguities. Working in a component oriented distributed environment can be very challenging and in some cases it can be possible that a self managing system itself faces the management problems. So to avoid such situations the standardization issues are important[4].

Dependence on system load: Self Management I not a magic that we turned on and all the system problems would be solved. It doesn't have to be active all the times e.g. when the system load is less then a certain level the self management system should downgrade its functional activities and upgrade the optimization modules and vice versa. Although there is no study done on the work load effects of self management systems yet but we assume the considering self management modules are working modules they logically would take system resources and left turned on when there is no need would result in wastage of system resources.

In addition to the discussion above, the transparent accounting of the software modules in the system (software modules maintaining history and answer to the request asked about the past activities), the constant update of self managed system and division of new rules, policies is very important to keep the system in an optimized state.

Applications: The AC carries a huge promise for large scale distributed system. Since the cost attached to maintain such systems has soared, the need for autonomic systems is increasing too. Several application areas are proposed but in this research we will discuss some of them.

Self-* applications: By far the most interesting application area is self-* applications especially the self management applications. Instead of being managed by human supervision, the system should watch itself. Mainly classified into self fault management, self configuration, self performance management, self account and self healing the self-* applications are the direct applications of autonomic computing concepts[10,12,15].

Pseudo-intelligent administration: Intelligent Administration (IntA) or supervision is desired to be working on top of self-* applications and provide the guidance to self-* applications. Many modern day projects are considering the term 'manager' for IntA[3,10,11]. The term 'manager' is a bit vaguely defined in terms of its functionality. Manager it self should be responsible for the management of all functional and non functional aspects of the system. We deduce that term manager is may be the light weight form of kind of expert systems that infer some directions for self-* systems. These 'managers' can contain policies, services, or components[14]. When called, the managers either provide atomic, co-related, or composed. These solutions can be just one or can be a set of many proposed solutions. The 'manager' executes those solutions and the system functionality is then analyzed through either some functionality models or ontology. Many candidate technologies are proposed for IntA development but we still think that the real time solutions are far from fetched for modern day communication systems.

Automated system testing: Testing systems are the most fragile software systems. Containing the probability of failure and success exactly the same, testing systems are the best to try the functional feasibility of any autonomic healing system[15]. This is relatively a new application area in AC promising to build some automated testing software testing tools that can test some system according to the functional and system requirements.

Autonomic computing in home networks: In the following context we discuss the architecture of u-AMS. There are core services that directly interact with the user. These are platform independent services that are either provided by service provider or generated by the system itself. These services are initiated when they are loaded from the service pool at the initialization of management gateway. Later on the service pool is updated as the new services are added and old services are updated. In u-AMS the services are called service bundles and service pool as u-function bundles pool. In that service pool all services individually and service pool collectively is monitored by the monitoring service. It is the logging system utility that gives the facility to monitor every instance taking place within the system. That monitoring services leads to Context Control Center (CCC). This part of system manages the profiles and reports to the context analyzer. It works out on the log files generated as a result of all the monitoring activities and hence it updates the profiles of all the entities. This part of u-AMS is even driven it goes through all the activities and if an abnormal activities are found, it reports to context analyzer. The Context Control Center is a kind of exchange that diverts the related to the relevant profile or things out of context (exceptions) to context analyzer. The profile, if left unexamined, will become too bulky. To reduce the size and for more efficient processing some data mining functions are performed on profiles. These data mining functions are performed through data mining agents. There is some normal profile of each entity and that is defined when it first gets attached to the system or the system software defines it default operation. When something happen other than the default operation it is separated and recorded. That makes the information size low (data mining agents are included in our future implementation plans).

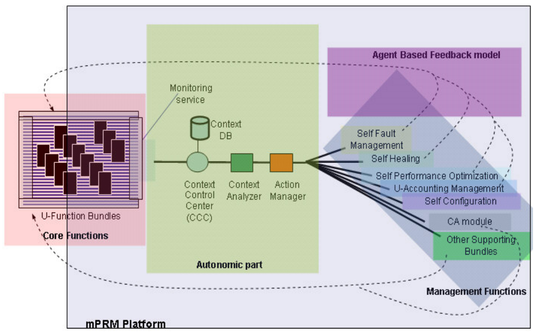

Fig. 1: System Architecture of u-ASH System

When an exception or malfunctioning is reported, the context handler is passed to context Analyzer module. The function of this part is mainly the pattern matching. Using pattern matching, it identifies the problem and handles the context to the Action Manager. Action Manager is mainly the policy checker. It checks the policies and takes the appropriate action as directed in policy. The type of violation is important here, whether it is breach of the sever type, or it is of a type that the system can handle with the help of policies. This part of the system predicts the level of reaction to the breach or malfunctioning. After the analysis of the problem and its identification, the context is handed over to the related management module. A reaction policy is devised by that management module and Agent Based Feedback Model is actuated and that policies are implemented on the services. We have chosen Agent based technology over here by considering the scenario in mind that the services may reside at a remote place so agents may be a good candidate for execution of remote action.

Implementation: We implement our work using toolkit provided by IBM for autonomic systems. We first make resource model and then those resource models are implemented using that toolkit. Behind any autonomic computing system is some form of autonomic management engine. This engine acts as the hub of the wheel, keeping all of the spokes together and pointed in their proper directions. An autonomic management engine such as IBM's Autonomic Management Engine (AME) coordinates the control loop for an autonomic computing system. In a control loop, a system continually monitors the environment, analyzes the information received, plans a response and executes that response. If you had to manage a new autonomic application for every issue you had, you wouldn't be saving much in terms of labor. You'd also start to run into problems in terms of standardization between systems, increased software complexity and so on. Instead, autonomic computing technology works on the idea of a single management engine running multiple resource models. A resource model is a set of criteria and instructions that get plugged in to AME. The resource model defines what resource to monitor (that is, disk space, processor time and so on), what events and conditions to look for and what to do in the event that those conditions do in fact occur.

Service discovery is home networks: Now, we describe the ubiquitous remote manager for the project we completed as part of a government sponsored academic initiative. A test bed for ubiquitous remote manager with home appliances is developed at Ajou University South Korea. This work is the part of Ubiquitous Autonomic Computing and Network (uAuto) project[16].

Thus far, as an initial prototype, ubiquitous remote manager offers service discovery mechanism for a limited range of home appliances. Users can discover, access and use services such as printer, TV, rice cooker, lights and cameras using his/her mobile phone or PDA. The ubiquitous service discovery (USD) has been developed through ubiquitous remote manager (URM). URM controls the home gateways remotely to reduce the complexity of the entire system and achieve more control over the operation and manageability of the network. In ubiquitous remote manager, we have added some value-added services as following services has described.

U-best bid service finds the best choice for the consumer, because consumer will be able to buy a best service in economical cost.
U-device advertiser The Device Advertiser will provide the information of various devices on the gateways. These devices on all gateways can help the service providers to Introduce service “PACKAGE”.
U-service composition User has the ‘device set’ to use a service or application BUT needs an add-on or driver / update. Automatic download for that missing component will be made available through this service.
U-service advertiser The Information concerning to services will be provided to the Users. It will be such marketing for service to consumer, that will more advantageous for service provider as well as consumers.
U-context Service The context/Profile service will keep the preferences information of user according to his/her behavior and life style. Then it will be used by services providers during providing their services. So we can it will useful information for ends. These quality services will allow the system to manage itself and hence virtually no human assistance will be needed for tiny matters.

The control center (CC) unit has vital role in the preliminary and consequent configuration. It is a set of connections with in the Ubiquitous Remote Manager. Only the control center can add/remove backend server hosts, configure the system databases, etc. The initial configuration of the system always begins on the control center. After initial configuration, the CC can go offline and the system can continue carrying out without it. However, the control center (CC) cannot be completely changes in the system configuration can be done only through this module.

A management server handles the communication with service gateways, monitoring changes in their configuration state, schedules management jobs for execution and keeps track of job results replied by service gateways. The management server uses the centralized database for storing configuration information about all gateways it controls, as well as pending for execution management jobs. Each management server is responsible for synchronizing the configuration state of service gateways with the configuration data stored in the database. The management server also provides support for storing and accessing all persistent information from gateways on the backend system, so OSGi[7] frameworks can be deployed on devices without any importunate memory (disk, flash RAM).

Fig. 2: U-service finder: A demo

It allows the service provider and service user to access Ubiquitous Remote Manager remotely. The service providers perform services publishing and administration through RAS. Users can access the Ubiquitous Remote Manager (URM) functions and services. The Ubiquitous Autonomic Management Server (uAMS) component is the Application server that will contain the services in the form of JAR files. These files can be accessed and downloaded by all other applications and servers in the system.

In Ubiquitous Service Discovery (USD) the most important obsession is a service is an entity that can be used by a person, a program, or another service. A service may be a computation, storage, a communication channel to another user, a software filter, a hardware device, or another user. Two examples of services are printing a document and translating from one word-processor format to some other. Ubiquitous Service Discovery (USD) provides mechanisms for service construction, discovery, communication and use in a distributed system. Examples of services include devices such as printers, displays, disk; software such as applications or utilities; information like databases and files; and users of the system. Ubiquitous Service Discovery (USD) architecture contains Service Pointer that will point (discover) the service with in the network to user, Service update service will update the services directory and cache

Services are found and resolved by a service Pointer. The service Pointer is the central mechanism for the system and provides the major point of contact between the system and users of the system. In specifically terms, a service Pointer takes queries from user and point the service from the services database and cache within the network to the user. When the Service provider adds some new services to services directory via Ubiquitous Remote Manager, The Service updates service, update the services directory and service cache and this service Directory and cache maintain by Management Server .These services can be put in through Service Advertise as well.

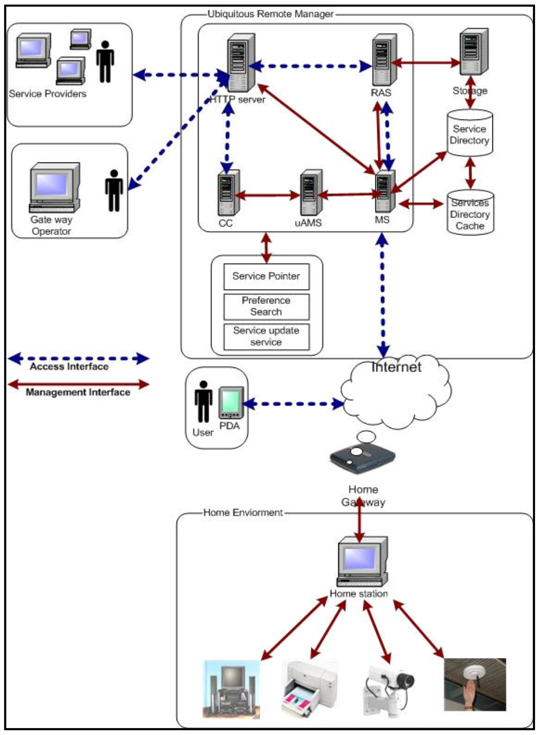

Fig. 3: Architecture of ubiquitous service discovery in home networks

Preference Search Service, expressive entries allow for more fine grained selection of services based on properties understood by peoples. This provides the user defined parameters for the search such as searching by service type or service provider.

CONCLUSION

Ubiquitous Computing needs to prove a lot in order to get into the daily lives of modern day users. Since there are no such application devices that could become the need of a mobile user, it seems that it's the visible automation and network management is the areas where the research community has to concentrate at. We in this study discussed the Autonomic Computing and Service Discovery as enabling technologies for ubiquitous systems. We shared our experience of application development and theoretical issues.

ACKNOWLEDGEMENTS

This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in South-Korea.

We would also like to thank Mrs. Fuchsia Chaudhry for her cooperation in improving the quality and the anonymous reviewers of this paper.

REFERENCES

1. Mark Weiser, 1993. Hot Topics: Ubiquitous Computing IEEE Computer.
2. Sundramoorthy, V., J. Scholten, P.G. Jansen and P.H. Hartel, 2006. On Consistency Maintenance In Service Discovery. 4th Int. Conf. on Information, Communications and Signal Processing and 4th IEEE Pacific-Rim Conf. On Multimedia, vol. 3, IEEE Computer Society Press, Los Alamitos, California,
3. Kawahara,Y., M. Minami, S. Saruwatari, H. Morikawa and T. Aoyama, 2004. Challenges and lessons learned in building a practical smart space Mobile 0.
10. Boutaba, R., S. Omari and A. Virk, 2001. SELFCON: An architecture for self-configuration of networks. Intl. J. Communications and Networks (special issue on Management of New Networking Infrastructure and Services), 3: 317-323.
11. Xiangdong D., S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri and S. Rao, 2003. Autonomia: an autonomic computing environment. Performance, Computing and Communications Conf., Proc. Conf. IEEE Intl., 9-11 April.
12. Nakajima, T. and I. Satoh, 2004. Personal home server: Enabling personalized and seamless ubiquitous computing environments. PerCom, pp: 341-345.
13. Nakajima, T, 2003. Pervasive servers: A framework for creating a society of appliances. Springer-Verlag London Ltd, Vol. 7, No. 3-4.
14. Siddiqui, F.A. and W.-S. Yoon, 2005. IP-based service and device portability across OSGi domains. Inform. Technol. J., 4: 391-397.
15. Chaudhry, J.A., S.-K. Park and S.-K. Hong, 2006. On recipe based service composition in ubiquitous smart spaces. J. Computer Sci., 2: 86-91.
16. u-Frontier: Ubiquitous Korea Project, http://www.uauto.net
17. Keshi, D., Y. Shiraishi, H. Niwamoto, M. Okada and H. Yamamoto, 2005. Is home network application acceptable or not? Circuits and Systems, 2005. ISCAS 2005. IEEE Intl. Symp., 5: 5337-5340.
18. Raz, O., P. Koopman and Mary Shaw, 2002. Semantic anomaly detection in online data sources. 24th Intl. Conf. Software Engineering (ICSE'02).
19. Haydarlou, A.R., B.J. Overeinder and F.M.T. Brazier, 2005. A self-healing approach for object-oriented applications. 3rd Intl. Workshop on Self-Adaptive and Autonomic Computing Systems.