CONTEXT-AWARE FOR SERVICE DISCOVERY IN AD HOC ENVIRONMENT

Weng-In Siu, Simon Fong, Zhen Sheng Guo, Kuok-Fan SiTou

Faculty of Science and Technology
University of Macau, Macau
{utakosiu, ccfong, zsguo, antonyst}@umac.mo

ABSTRACT

Service discovery is a critical element to the success of ad hoc computing. The traditional approaches of service discovery in wired networks are not directly applicable to ad hoc environment due to the inherent network dynamics and unstable connectivity. Contexts capture the current physical and virtual environment that uncovers the needs of user. This paper discusses the contexts that are important for service node in service provision, and client node in service acquisition. A new context-aware service discovery mechanism is proposed that exploits contexts to improve the current ad hoc service discovery techniques.

KEYWORDS

Ad hoc Networks, Context-aware, Service Discovery, Pervasive Computing

1. Introduction

With the emergence and wide acceptance of diverse set of personal wireless devices, it is foreseeable that these handy computing widgets will become the mainstream of communication tools in the near future. Utilizing short-range radio technologies such as Bluetooth and Wi-Fi, these widgets can organize themselves dynamically and autonomously into transient network, in which they share resources, provide service and consume service. Every one can move freely, join and leave the network without restraint, and possibly without disruption to others. Typical applications are outdoor special events such as exhibitions, conferences, concerts and festivals; communications in regions with no infrastructure, in emergencies and natural disasters.

One challenge in realizing this ad hoc way of communication is how to discover services when a user moves into a new environment. This problem is similar to the service discovery issue in traditional fixed network. After more than 10 years of research efforts, a number of service discovery methods were standardized and implemented. The more famous methods are Jini [1], Universal Plug and Play (UPnP) [2], Salutation [3] and SLP [4]. Their main approach is to use broadcast to advertise and locate services in the network. Some even employ central directory servers to handle service registration and service query in order to achieve greater efficiency. However, without bearing the ad hoc features in mind, these traditional methods are inherently defective in the fluctuating ad hoc networks and the resource-poor mobile nodes: some require the existence of central directory server to coordinate service provisioning (e.g. Jini); some produce voluminous message broadcasts due to frequent join and leave of nodes that request significant node processing (e.g. SLP, Jini and UPnP); some impose complex and resource-intensive requirements to host a service, as well as using the service (e.g. Jini and Salutation).

Yet another major limitation of current methods is not taking context information into account. Applications in ad hoc networks are ad hoc in nature. They are not pre-determined and only emerge when current situation requires them. Both user and service-providing applications may be frequently changing their requirements even after established connection. Nevertheless, context characterizes the current situation of the entities involved in communication, which includes the user, the hardware platform, the application environment, the computational resources, the physical and temporal factors, etc. And hence, adaptations can be made upon detection of changes.

In this paper, we discuss the importance of context-aware in service discovery, why and what is necessary in the ad hoc environment. We also propose a framework for managing and using contexts in service discovery and service provision of service and client nodes.

2. Service Discovery Protocols for Ad Hoc Networks

Service discovery for wired network has been a hot research topic for many years. A number of popular standards were developed and implemented by academia

and industry [5]. However, protocols that are designed for ad hoc networks are only handful. Some of them are adapted from the traditional approaches, and others buy the concepts from group-based collaboration such as multi-agent communications and peer-to-peer networking.

Based on the lightweight SLP, S. Motegi, K. Yoshihara and H. Horiuchi [6] propose the enhanced version with improved caching and service request distribution mechanism. Unlike the old SLP in which the intermediate nodes respond on services from cache and do not rebroadcast the message, instead the message is propagated to nodes behind in order to increase the number of discoverable services. Both source and destination IP addresses are added to the request and reply messages for improving hop-by-hop processing.

Another peer-to-peer caching service discovery framework for Ad hoc network, Allia [7], suggests formation of alliance between nodes, where the allied node caches the advertisements from its member nodes. It is up to the node's local policy to decide whether functioning as an allied node of the others, i.e. the combination of user's preferences and resources present in the node. The size of cache, request forward and advertisement handling are also determined in same measure. To minimize broadcast traffic, service request is resolved by first searching from local cache, then member nodes and finally broadcast if no answer is found. However, since this framework requires a complete agent platform installed in each node disregarding its policy, the computing requirements could be quite high for low-profile handheld devices.

Improving the service discovery algorithm has been the focus of most researches, but detecting and adapting the current situation in the service discovery process should not be overlooked, especially in the dynamic ad hoc environment. The use of context-aware in service provision firstly appears in [8], which use a uniform document format (CAP) to encapsulate user contexts into service constraints. A CAP document consists of context values obtained directly from sensor measurements, called context configurations, other referenced data and simple scripts. The CAP is sent to the service node in local vicinity for service selection and execution, where it is processed through a nested tree relation to match up the service, or service routing to remote domain if no match is found. Nonetheless, the framework requires the directory server and service nodes running on a static platform, hence it is designed without full considerations on the ad hoc services.

In the ad hoc environment, value of each parameter is changing. It requires a truly efficient framework to deal with the dynamics and technique for quick adaptation. A new vision of applying context-aware in ad hoc networks is discussed in [9]. The ACAN architecture proposes a three-layer protocol stack to provide a common communication and interaction model for service discovery, user presence, mobility, auto configuration, setting security levels and QoS. Contexts obtained from sensors in the mobility layer (layer 1) directly feed to the Ad hoc application layer (layer 3) for adaptation, which can choose either to adapt itself to the current situation or adapt the environment by deploying active agents to resolve conflicts. The Ad hoc active layer (layer 2) responsible for network connectivity and auto-configuration of addresses. Figure 1 shows the main concept of ACAN layers.

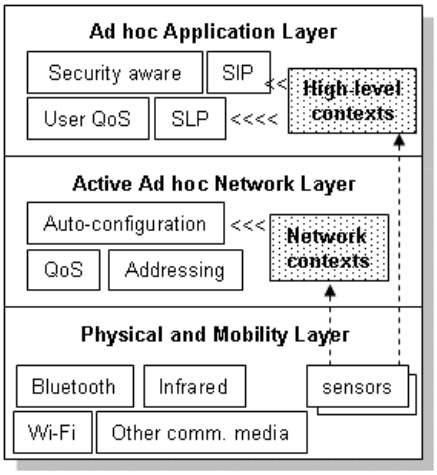

Figure 1. ACAN Layer architecture

As conclude from the discussed works, the use of contexts in service discovery has been very limited.

3. Context-Awareness

To work with contexts, the foremost task is to understand it. Quoted from [10], "context is any information that can be used to characterize the situation of a person, place or object, which is considered relevant to the interaction between a user and an application." And the system that uses context to provide relevant information and services to user is considered context-aware, where relevancy depends on user's task. Mainly, there are four types of context information:

i. Physical context – location, time, presence, etc.

obtained by means of physical sensors.

ii. Network context – bandwidth capacity, round-trip time, topology, QoS, etc. gathered from network sensors.
iii. Computational context – capability of CPU, amount of memory, buffer size, supporting encoding modes, running application, etc.
iv. User preference context – timing perception, threshold time, physical proximity, cost, liability, identity, etc.

It is important for a mobile application to consider all the possible contexts; however, they are used selectively based on the relevancy and importance in the application, and how pervasive the application is.

4. Contexts for Service provider and Service requestor

In ad hoc network, there is no assumption that service nodes are more stable and resource-rich than anyone else; instead, they are as dynamic and resource-constrained as the client nodes. When sharing becomes a casual act, more users are willing to contribute at no cost (like many peer-to-peer network applications) or sometimes at cost. Thus service provider should also be aware of its contextual situation in order to guarantee the performance for local user, yet attain acceptable quality of service offered to remote user. According to the observed contexts, service node can determine if a service request is acceptable, and negotiate with the client node the quality of service provided based on the amount of computing resources that can be allocated for service. E.g. the percentage of incoming and outgoing traffic, the CPU time, etc. Sometimes, context values can be supplied to client to describe the current status of the desired service such as the number of client using the service.

In the client node, context is an important factor in two aspects: (a) express the implicit service need of the user [8], (b) choose the most suitable service from the responded service information. Client node can initiate the service request proactively based on the combined user contexts, e.g. meeting room (location) and presentation software (running application) manifest the need of finding a projector. Contexts can well be used to select the most suitable service when more than one services responded e.g. the nearest (proximity) and the cheapest (cost) service. It may even be better to include some context requirements in the service request to reduce the number of matching services. E.g. the queue size is less than 2 (threshold).

5. Context-Aware Service Provision Mechanism

Contexts are collected by physical and virtual (software) sensors. Context values are stored in one central location for simple management and ease of access.

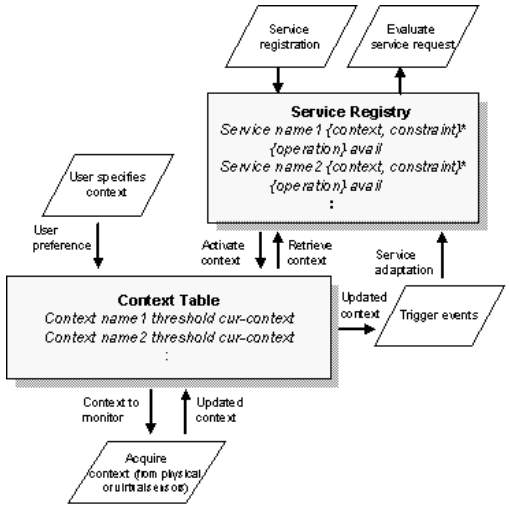

Figure 2. Context-aware service provision

The service node registers its available service in the local **Service Registry**. Each service entry contains three parts: zero or more context and constraint pairs, the operation for analyzing the combined contexts, and finally the service availability. Context to be monitored is entered into the **Context Table** as a single entry. Different context sensors are invoked to actively monitor the environment and periodically update the context entries in the table. When service request arrives, it is evaluated by matching the available services in the Service Registry, and then the service reply is composed.

When current context value of an entry is unfavorable and exceeding the specified threshold, it triggers an event to the related service for immediate analysis, which may result in one of the following actions taken:

i. Stop the service temporarily until favorable contexts return
ii. Reject any service request
iii. Inform service user (if there is client using the service) the adjusted affordable quality of service, and the life time of this service session.

Service discovery request can be invoked in two cases. First, when an application requesting for specific service is explicitly launched by the user. User context entry is created (if necessary) to the context table for immediate

context lookup. Second, when the existing contexts that are created by the system or user by default, are updated that reflect the current needs of the user. Usually only a number of contexts occur coincidentally can trigger the analysis process. Context-Service association rules that are preset or created from past experience are then resolved to reach a conclusion.

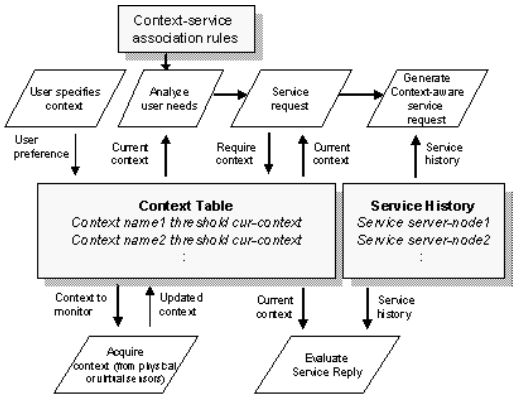

Figure 3. Context-aware service request generation

Service request contains not only the type of the service, but also some user contexts and preferences from service history, which can be evaluated in the service node for decision of service provision. Once service request is composed, it is disseminated to the network by one of the service discovery method discussed in previous section. When the service replies are received, they are filtered by re-examining the current local contexts. Finally, the remaining service replies are compared among their service node contexts to select the possibly best service environment. Again, service history can provide previous service-using experience as a reference. E.g. if a user used to print the color photo with printer X because of the brightest image it can produce, it is given higher rating during the service selection process.

6. Future Works

The benefits of using contexts in service discovery is undeniable. As there are plenty of contexts available, and processing context information consumes certain computing resources, it is important to carefully test and include useful contexts into the mechanism. Currently, we are prototyping the proposed context-aware mechanism into the enhanced SLP of [8]. As future work we will define the standard interfaces for service request invocation, service registration, context evaluation and context information updates. We will also refine the service discovery routing for maximum benefits of context-awareness.

7. Conclusion

Service discovery is the fundamental issue in ad hoc networking. To accurately target user's need and adapt the fluctuating environment, current user contexts are necessary. This paper discusses the important contexts that must be considered in service provision in service node, and service acquisition in client node. The novice mechanism is proposed to handle contexts in both cases.

References

[1] Sun Microsystems Inc., Jini Technology Architectural Overview, *Sun White Paper*, Jan 1999.

[2] Microsoft Corporation, Microsoft Windows CE .NET - UPnP Framework, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceupnps/htm/_wcecomm_win32_upnp_framework.asp, Aug 2002.

[3] The Salutation Consortium, Salutation Architecture Specification v. 2.1 (Part I).

[4] IETF, Service Location Protocol, Version 2, *RFC 2608*.

[5] Sumi Helal, Standards for Service Discovery and Delivery, *IEEE Pervasive Computing*, 1(3), 2002, 95-100.

[6] S.Motegi, K.Yoshihara, H.Horiuchi, Service Discovery for Wireless Ad Hoc Networks, *Proc. 5th IEEE Int. Symposium on Wireless Personal Multimedia Communications*, 2002, 232-236.

[7] O.Ratsimor, D.Chakraborty, A.Joshi, T.Finin, Allia: Alliance-based Service Discovery for Ad-Hoc Environments, *Proc. of the 2nd Int. workshop on Mobile commerce*, Atlanta, USA, 2002, 1-9.

[8] M. Samulowitz, F. Michahelles, C. Linnhoff-Popien, Adaptive Interaction for Enabling Pervasive Services, *Proc. 2nd ACM Int. workshop on Data engineering for Wireless and Mobile access*, Santa Barbara, USA, 2001, 20-26.

[9] M. Khedr, A. Karmouch, ACAN – Ad Hoc Context Aware Network, *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering*, 2002, 1342-1346.

[10] Anind K. Dey, Gregory D. Abowd, Towards a Better Understanding of Context and Context-Awareness, *Proc. Conf. on Human Factors in Computing Systems*, Hague, Netherlands, 2000.