An Adaptive, Lightweight and Energy-Efficient Context Discovery Protocol for Ubiquitous Computing Environments

Stephen S. Yau[1], Deepak Chandrasekar[2], Dazhi Huang[1]
[1]Department of Computer Science and Engineering
[2]Department of Electrical Engineering
Arizona State University
Tempe, AZ 85287-8809 USA
{yau, deepakvcd, dazhi.huang}@asu.edu

Abstract
In ubiquitous computing (ubicomp) environments, it is often necessary for the applications to have situation-awareness capability. Current research on situation-aware applications is mainly based on situations with a pre-defined set of contexts. To achieve more flexible situation-awareness, situation-aware applications should have the ability to discover and collect contexts in a timely and organized fashion. Due to the dynamic and ephemeral nature of ubicomp environments, a context discovery protocol (CDP) that defines the entire process of discovering, acquiring, aggregating and storing contexts, is needed. Because of the serious resource constraints on ubicomp devices, the CDP must be adaptive, lightweight and energy-efficient. In this paper, a CDP with such characteristics for ubicomp environments is presented. The experimental results to show that our CDP is more energy-efficient and has a small increase in latency, compared to a simple context discovery protocol, are also presented.
Keywords: Context discovery protocol, ubiquitous computing, middleware, situation-awareness, adaptiveness, and energy efficiency.

1.Introduction

In ubiquitous computing (ubicomp) environments, it is often necessary for the applications to have their mobile devices with situation-awareness (SA) capability. We consider that a *situation* as a set of past *contexts* and/or actions of individual devices relevant to future device actions, and a *context* is any instantaneous, detectable, and relevant condition of the environment or the device [1]. Contexts may be sensed and extracted by a set of sensors deployed in the environment or connected to mobile devices with various capabilities to allow multi-modal sensing. The execution of SA applications in ubicomp environments normally involve the following phases as shown in Figure 1:

(1) Context sensing and acquisition. The SA applications collect contexts from ambient environments.
(2) Situation analysis. The SA applications analyze the collected data and determine what the situation is.

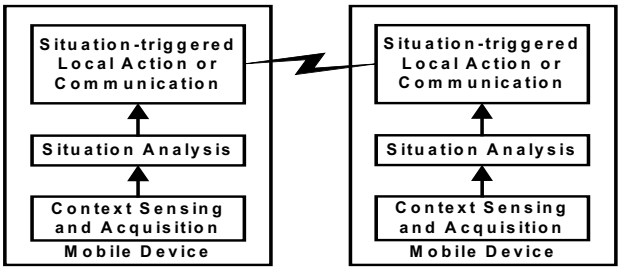

Figure 1. Phases in SA application execution

(3) Situation-triggered action (local) or communication (remote). Based on the current situation, proper local actions of the SA applications or communication with other SA applications in remote devices will be triggered.

Application layer packages and protocols have been developed for Phases (2) and (3) [1-5]. However, contexts used for situation-awareness in [1-5] are predefined and provided by a static context server. To achieve more flexible situation-awareness, the SA applications at the device level must have the capabilities to discover and collect contexts in a timely and organized fashion because wireless devices in ubicomp environments may frequently move in and out of the communication ranges of other devices. It will be very difficult, if not impossible, to maintain a complete and coherent view of mobile devices with sensors in the network. The values of contexts which can be numerical, nominal or structural also make the processing of contexts difficult. To process various contexts efficiently, a mechanism that can convert raw contexts into a well-organized format is needed. Since contexts may continuously change over time, to ensure the correct execution of the SA applications, updated contexts need to be provided to the SA applications within a reasonable delay. Finally, since an SA application may dynamically change the requirements of contexts needed for situation analysis, mechanisms for context discovery must adapt to the needs of the SA application. This implies that these mechanisms are better placed in the application layer. Due to these reasons, a Context Discovery Protocol (CDP) in the application layer that

Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04)
0-7695-2118-5/04 $20.00 © 2004 IEEE

defines the entire process of discovering, acquiring, aggregating and storing contexts is needed. Because of the serious resource constraints on ubicomp devices, the CDP must be *adaptive*, *lightweight* and *energy-efficient*.

So far, the research in context discovery has not addressed the above-mentioned issues and mainly focused on the network layers. Trailblazer [6] developed an energy-efficient routing protocol that makes use of leader nodes to discover shortest paths to support data-centric routing of context for context-aware applications. The protocol does not provide the ability to adapt with the needs of the application. In SPIN [7], the protocol assumes that all devices have identical capabilities and the devices advertise meta-data descriptors of sensor data by flooding the network. In the Directed Diffusion [8] protocol, sensor data are associated with attribute-value pairs, wherein the nodes request data by advertising interests for these attributes. However, when multiple sources for the requested data exist, extra energy is needed to obtain the requested data through different paths. The Dataspace [9] protocol is used in sensor networks comprising of heterogeneous nodes and attempts to organize correlated sensors nodes into "dataflocks" in a three-dimensional space, wherein queries on sensor data are forwarded using a statically hierarchical structure, which is not suitable for ubicomp environments.

In this paper, we will present an adaptive, lightweight and energy-efficient context discovery protocol (CDP), called R-CDP, for ubicomp environments. This protocol can be used in context-sensitive middleware, such as Reconfigurable Context-Sensitive Middleware (RCSM) [1-5]. Context-sensitive middleware can provide necessary software and hardware infrastructures for using a CDP, including (1) interfaces for context acquisition, (2) interfaces for extracting application requirements on contexts and notifying applications on the availability of contexts, (3) interfaces for propagating contexts to the application layer, and (4) mobile ad hoc communication capability. The implementation of R-CDP, and the evaluation of energy-savings and computational overhead of R-CDP with experimental results will also be presented.

2. An Example: Smart Classroom

Before presenting our R-CDP, we would like to illustrate the need for context discovery by an example.

Consider a Smart Classroom, in which students and instructors use their PDAs to engage in collaborations such as group discussions, appointment making, and course material dissemination [10]. Some of these PDAs are connected to peripheral sensor units that collect context data from ambient environments. A set of SA application software is used in PDAs for collaborative learning. The following scenarios show the need for a context discovery protocol:

1. The instructor enters his office with his PDA before heading to his class. His PDA detects his location, current time and the presence of the desktop computer in his office, based on which the PDA autonomously connects to the computer in his office, notifies him of an upcoming class and provides him a list of lecture slides on the desktop PC for him to download. In this scenario, the instructor's PDA needs to collect three contexts: location, time, and the identity of a neighboring device.
2. The instructor then heads to his class with his PDA. On entering the classroom, the PDA takes attendance by discovering the PDAs of the students. The instructor's PDA then attempts to collect light intensity and noise level in the classroom, based on which a decision to disseminate the lecture slides to students is made. In this scenario, the instructor's PDA needs to collect the location, time, identities of students' PDAs, light intensity, and noise levels in the classroom.

In these two scenarios, when the instructor is in different application environments, the instructor's PDA needs to collect different contexts. Obviously, a specific set of contexts for a particular application can be pre-defined for collection by the PDAs. However, this solution limits the usage of PDAs to this particular application, i.e., when the application changes, PDAs cannot adapt to the different requirements for contexts. In order to allow such adaptations, the PDAs must have the capability to discover available contexts in previously unknown environments, which can be provided by the usage of a CDP.

3. R-CDP: An Adaptive, Lightweight and Energy-Efficient Context Discovery Protocol

Before introducing the design of R-CDP, let us define the terms to be used in the following sections:

- ***Requester***: A device that requests contexts from remote devices for an SA application software running on the device.
- ***Provider***: A device that provides requested contexts for ***Requesters***
- ***Request***: A beacon sent out by a ***Requester***, requesting for contexts.
- ***Result***: A beacon sent out by a ***Provider***, containing requested context.
- ***Neighbor***: A device in the vicinity of the network for another device.
- ***Sensing Unit (SU)***: A software or hardware unit, which can provide one or more contexts.

The design of R-CDP is based on the following ideas:

- Combinations of pull and push communication paradigms for dynamic context discovery and efficient context retrieval. The R-CDPs in *Requesters* dynamically discover contexts from *Providers* using pull communication paradigm, while the R-CDPs in

Providers proactively push contexts to *Requesters* whenever the contexts undergo measurable variations.

- Reduction of *Request* retransmissions and redundant context *Result* transmissions to improve energy-efficiency. R-CDP is designed to be aware of network conditions, and adaptively changes the interval of *Request* advertisements to alleviate network congestions, thus reduce retransmissions of *Requests*. It also uses transmission probabilities to reduce redundant context *Result* transmissions and adaptively changes transmission probabilities as the number of devices in the network varies.
- Energy-aware and adaptive control on context updates. R-CDP adopts and extends the Refresh Priority (RP) function [11] to determine when *Providers* should send updates in context, by calculating the RP based on the divergence of contexts and energy consumption of *Providers*. R-CDP also adapts to the needs of SA applications by allowing them to control the degree of divergence for which the context updates should be sent.

Figures 2 and **3** depict the different steps involved in the discovery process in *Requester* and *Provider* devices. Detailed descriptions of these steps will be presented below. We will use "R" and "P" to represent the steps in the *Requester* and *Provider* devices respectively:

Step 1. Context Retrieval Channel Setup

(Step 1R) As shown in Figure 2a, in the *Requester* device, an SA application registers with R-CDP by sending a list of required contexts for situation analysis. Then, R-CDP lookups the Context Repository (CR), which stores information of all contexts on the device, to see if the requested contexts are available locally. If a requested context C_0 is available locally, C_0 can be retrieved from the CR. Otherwise, the *Requester* sends out a *Request* beacon, advertising its need for C_0, and waits for *Result* beacons. If R-CDP cannot receive any *Result* beacon for C_0 within a time-out, the application is promptly de-registered. On receiving any *Result* beacon for C_0, go to (Step 2R).

(Step 1P) As shown in Figure 2b, a new SU registers with R-CDP by sending the name and type of the contexts provided by the SU, the methods of acquiring contexts, and the range of supported frequency of context acquisition. On receiving an SU registration, R-CDP updates the CR. These contexts are indexed by their sources SUs, and a time-stamp to indicate when the contexts become invalid and should be removed from the CR. On receiving a *Request* for C_0, the *Provider* searches the CR for C_0. Depending on the availability of C_0, a context match may occur, and trigger the sending of a *Result* beacon containing C_0 to the *Requester*. After sending out the *Result* beacon, go to (Step 2P).

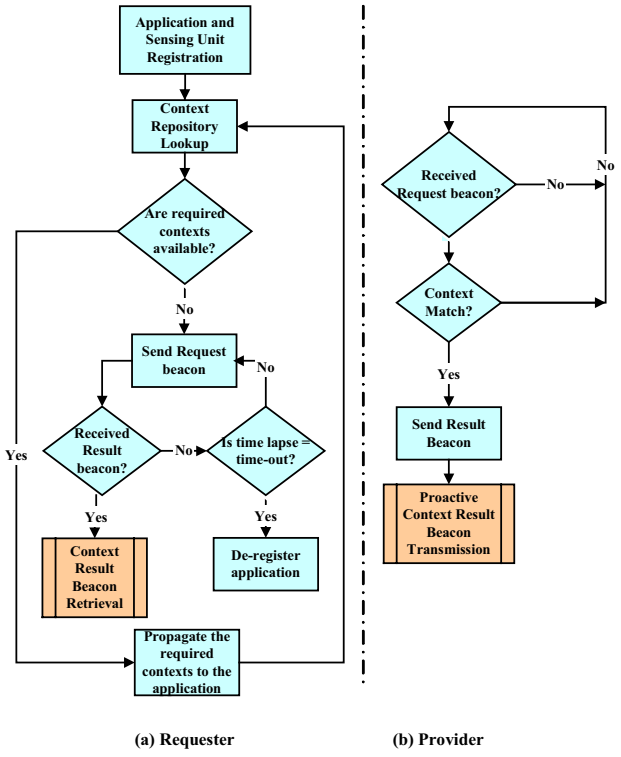

Figure 2. Context Retrieval Channel Setup

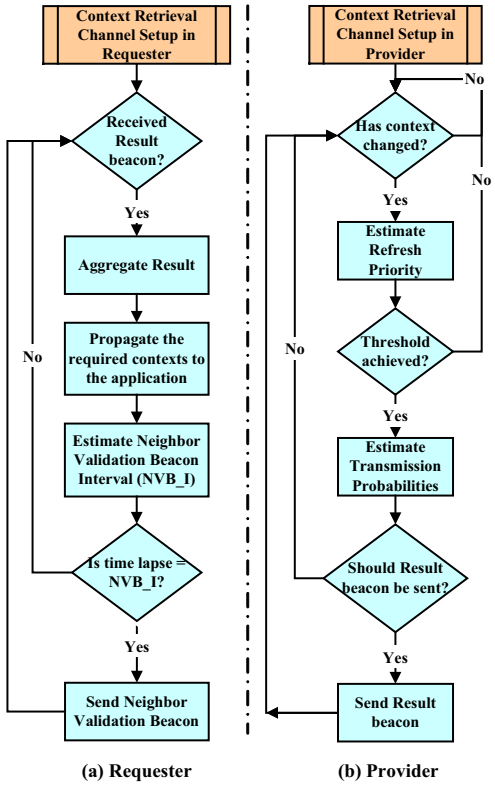

Figure 3. Context Retrieval Channel Maintenance

Step 2. Context Retrieval Channel Maintenance and Update

(Step 2R) As shown in Figure 3a, in the *Requester* device, R-CDP aggregates received *Results* when receiving *Result* beacons for C_0 from multiple *Providers*, and propagates the aggregated result to the SA application. The *Requester* also sends out *Neighbor Validation Beacons* (NVBs) at regular intervals to let the *Provider* be aware of the *Requester's* presence and its continued need for future updates on C_0. The NVBs contain feedback on N_p, which is the number of *Providers* with matching C_0's, and T_0, which is the value of threshold, to be used in determining when the context updates should be sent. On receiving NVBs, devices that have recently joined the network and are capable of providing C_0 also respond, thus allowing the *Requester* to discover new *Providers* in the network. The *Requester* device adaptively changes the period of sending NVBs to avoid collisions with NVBs sent by other *Requesters*.

(Step 2P) As shown in Figure 3b, the *Provider* constantly analyzes the data from the CR by calculating RP_0, the refresh priority of C_0. By comparing RP_0 with T_0, the *Provider* determines whether a context update for C_0 should be sent to the *Requester*. If RP_0 is greater than T_0, the Provider decides whether the Result beacon should be sent based on a transmission probability adaptively varied based on N_p.

Based on the above process, we can easily see that R-CDP is lightweight because it only involves simple device operations, such as CR lookup and maintenance. R-CDP is adaptive and energy-efficient due to the mechanisms used in Step 2, which will be discussed in Section 4.

4. Mechanisms for making R-CDP adaptive and energy-efficient

In general, the energy consumption due to wireless communications is proportional to the size of data sent and received. Unicast transmissions are better suited in terms of the overall energy consumption in the network of devices, and hence unicast transmissions are used to send *Result* beacons in this protocol. On the other hand, *Requests* and NVBs are chosen to be broadcasts because they are used to discover new devices and contexts for which only broadcast beacons can be used. However, broadcast beacons are more prone to collisions and may result in retransmissions, thereby increasing energy consumptions. To achieve energy-efficiency, three mechanisms, Refresh Priority Function, Adaptive Transmission Probabilities, and Adaptive Selection of NVB Intervals, are developed in R-CDP.

4.1 Refresh Priority Function

R-CDP uses an RP function to determine (a) when a *Provider* needs to send the updated context to a *Requester*, and (b) the priorities among different contexts being sent to various *Requesters*.

For (a), a simple solution would be always to provide the application with up-to-date context data, which might over-tax the network, computational or energy resources. Also, it may be unnecessary or problematic to send all context updates due to the two reasons: First, while certain applications might require rapidly fluctuating values of context for triggering certain actions, e.g. monitoring and control systems for critical facilities, other applications may prefer context values that have stabilized over time, e.g. light controls. Second, sensing units may provide an inaccurate measure of contexts due to some random events, e.g. a person passing by blocks a light sensor. These random events do not conform to the trends of the actual context changes.

For (b), a *Provider* may provide different contexts for various *Requesters* at the same time. When these contexts change together, the *Provider* needs to determine which context should be sent first, i.e., which *Requester* needs to be served first.

In order to address (a) and (b), we use a threshold-based approach. The *Provider* monitors the changes in the requested context, and estimates the RP for this context based on the divergence, which is the difference of a changing context value at the *Provider* over a period of time. The following modified version of RP function [11] is adopted in R-CDP to suit ubicomp environments:

$$RP(Ci,t_{now}) = (t_{now} - t_{last}).D(Ci,t_{now}).W(Ci,t_{now}) - \int_{t_{last}}^{t_{now}} D(Ci,t).W(Ci,t).dt \quad (1)$$

where

- $D(C_i,t_{now}) = B(t_{now}).C(C_i,t_{now})$ is the divergence function.
- $B(t_{now})$ is the divergence of the battery level at t_{now}
- $C(C_i,t_{now})$ is the divergence of the i^{th} context data at t_{now}
- $W(C_i,t_{now})$ is the weight of the context C_i at t_{now}, and is a function of the number of times C_i is requested.

The core of this function is the estimation of divergence. A *Result* beacon refreshes the context value to the latest, and hence the divergence becomes zero. Between such refreshes, however, the divergence value may become greater than zero. The value of the RP function for a particular context at a certain instant in time is used to determine whether a sufficient level of change on this context, the *threshold*, has occurred over time, thus requiring an update to be sent to the *Requesters*. By adaptively changing the *threshold*, the frequency of context updates can be varied. This can be used to satisfy different application needs for either stabilized or fluctuating contexts.

When different contexts need to be sent to various *Requesters* at the same time, the RPs of these contexts are

used to determine the order of sending these contexts. From (1), the value of the RP function mainly depends on the weight and the divergence. Given the same divergence, a higher weight would indicate a higher RP and hence a context with higher weight is considered more important and expected to be refreshed more frequently.

It is also noted that the net divergence reflects the product of the divergence of the device's battery level and the context. This implies that as the battery level reduces, the product with the diverging context will increase, thereby decreasing RP and the frequency of updates. Thus, the above policy also has energy-awareness incorporated.

4.2 Adaptive Transmission Probabilities

In ubicomp environments, there may be many *Providers* that have the requested context. On receiving a *Request* beacon, if N *Providers* with the requested context flood the network with N *Result* beacons, it will result in a huge loss of energy in both the system and individual devices. To determine which *Provider* sends *a Result beacon* is a challenging task since ubicomp environments are dynamic and lack the support for a centralized infrastructure. Usage of any mechanism to coordinate the relaying of *Result* beacons will result in additional communication overhead for the coordination, which may negate any benefits on energy savings thus achieved.

Instead of trying to develop a coordination mechanism, a simple technique that uses transmission probabilities as a basis for relaying *Result* beacons is provided by R-CDP. The basic idea is, as the number of *Providers* for the requested context increases, the transmission probability decreases. Reduction in transmission probabilities is directly related to the probability of the *Requester* receiving at least one *Result* beacon. For instance, let us assume a *Requester* R_0 requested light intensity of a room, where 3 *Providers* for light intensity are in the vicinity of R_0. By reducing the probability of R_0 receiving at least one *Result* beacon from 100%, i.e., all 3 *Providers* must send *Result* beacons to R_0, to 99%, i.e., each *Provider* chooses a probability of $[1-(1-0.99)^{1/3}]=0.78$ with which it transmits a *Result* beacon, we can expect to achieve a 22% savings in energy consumed by transmitting *Result* beacons. As the number of *Providers* increase, the expected percentage in energy savings increases. This probability-based mechanism is very lightweight and fair to all *Providers* in the network.

4.3 Adaptive Selection of NVB Intervals

As discussed earlier, only broadcast beacons can be used for sending NVBs to other devices in the nearby vicinity. However, when multiple *Requesters* send NVBs simultaneously, it results in collisions and thereby a possible loss in beacons, which will then require retransmissions resulting in increase in both energy consumption and latency. In order to estimate the losses incurred, we have conducted an experiment showing that the percentage of lost beacons increased from 1% to 77% when the number of *Requesters* in the network increased from 1 to 5, and the *Requesters* are sending out beacons every 250ms. In order to reduce the energy consumption and improve the performance of R-CDP, we have developed a mechanism that reduces the probability of collisions by adaptively varying the intervals of sending NVBs in R-CDP. The interval of sending NVBs (INVB) is divided into slots. The length of each slot would be the time taken by a device in sending NVBs. Thus, if we have T slots, and the number of *Requesters* in the network is X, then the probability of no two devices sharing the same slot is

$$P(X)_{No\ -\ Collision} = \left[\frac{(T-1)!}{(T-X)!(T)^{(X-1)}} \right] \quad (2)$$

By selecting a desirable $P(X)_{No\text{-}Collision}$, we can determine the value of T and hence the period of transmission for a given acceptable value of $P(X)_{No\text{-}Collision}$. By listening to NVBs and *Requests* from other devices during the previous NVB_I, a *Requester* can estimate the number of *Requesters* in the vicinity and use this number to adaptively change the next NVB_I.

5. Implementation of R-CDP in RCSM

We have chosen RCSM to implement R-CDP because RCSM provides the required software and hardware infrastructures to support R-CDP. RCSM [1-5] provides development and run-time support for SA applications in ubicomp environments. It includes the two main components:

- RCSM Object Request Broker (R-ORB): R-ORB is the key component for providing SA ad hoc communication and low-level context acquisition transparency in SA application software. For low-level context acquisition, R-ORB provides the interface between PDAs and sensors that are connected to FPGA units, and interfaces for acquiring context data from software-based sensing units.
- Situation-Aware Adaptive Object Containers (SA-ADC): RCSM provides SA-ADCs, which are application-specific situation analyzer components. A Situation-Aware Interface Definition Language (SA-IDL) is provided for software developers to specify SA object interfaces and required contexts for situation analysis. By compiling an SA-IDL specification, an SA-ADC for a particular SA object is generated automatically.

6. Evaluation of R-CDP

RCSM with R-CDP is implemented using Microsoft eMbedded Visual C++ on Windows CE platforms. Cassiopeia's E-200 and Dell AXIM were used for experimentation. The E-200s are used for connecting Field Programmable Gate Array (FPGA) boards interfaced with sensor units. The E-200s and the AXIMs are connected

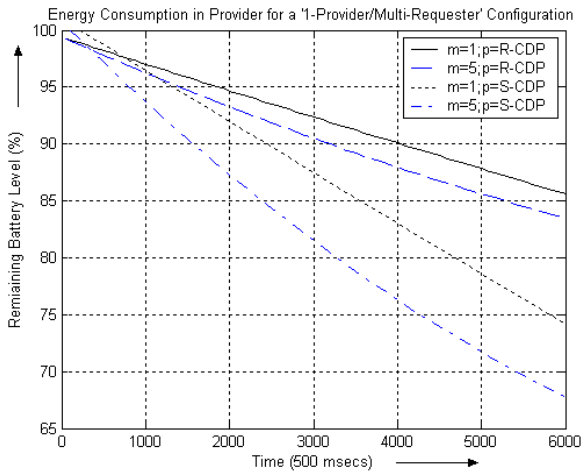

Figure 4. Energy consumption in *Provider* for one *Provider* multiple *Requesters* configuration

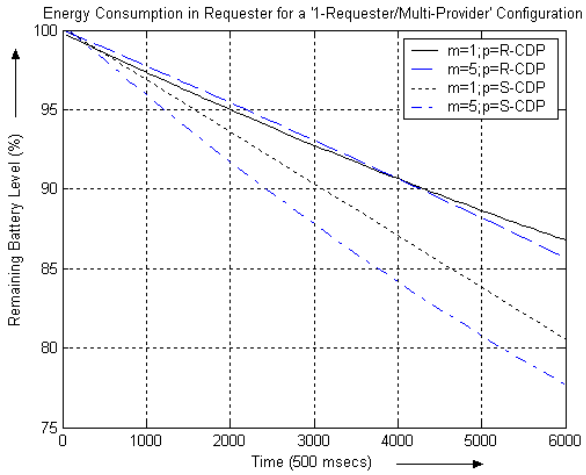

Figure 5. Energy consumption in *Requester* for one *Provider* multiple *Requesters* configuration

over an 802.11b network (ad hoc mode). The experiments conducted were specifically aimed at evaluating energy consumption and latency metrics of R-CDP. The following two configurations were used: One Provider – Many Requester, and One Requester – Many Provider.

For each configuration, the energy consumption and latency of R-CDP are measured. We compared our results with the results obtained from a simple CDP, called S-CDP, in which the RP function and transmission probabilities were not used. Thus, in S-CDP, the Provider sends *Result* beacons when the context changes. The comparison shows that R-CDP is energy-efficient with only a small increase in computational overhead. This computational overhead is measured in terms of the incremental latency incurred due to usage of energy-efficient mechanisms in R-CDP.

Energy Consumption:

The metric evaluates the rate of energy consumption in *Requesters* and *Providers* as the number of *Requesters* and *Providers* in the network vary. In Figure 4, 'm' corresponds to the number of *Requesters*, whereas in Figure 5, 'm' corresponds to the number of *Providers*. In both Figures 4 and 5, 'p' refers to the protocol used.

- From Figures 4 and 5, we can see that R-CDP consumes much less energy than S-CDP.
- Figures 4 and 5 show that as the numbers of *Providers/Requesters* increase, energy consumption in both the *Requester and Provider* show marginal increase in R-CDP, whereas energy consumption in S-CDP shows considerable increase. The increase of energy consumption in both R-CDP and S-CDP is attributed to the fact that, as the number of *Providers/Requesters* increases, the number of *Result/Request* beacons sent/received increases.

As shown in Figure 4, the divergence in the rates of energy consumption in S-CDP as the number of *Requesters* increases from one to five is more than the divergence in the rates of energy consumption in R-CDP. This shows that the usage of the RP function in R-CDP adaptively reduces the number of *Result* beacons sent to each *Requester* as the number of *Requesters* increases.

As discussed in Section 4, when the rate of energy consumption increases, the value of the divergence function D increases, and hence the RP decreases. As shown in Figure 5, the number of *Result* beacons received by the *Requester,* as a result of context updates in the *Provider*, increases in S-CDP, whereas in R-CDP this number remains about the same. This is because the *Providers* use transmission probabilities to control the transmission of these *Result* beacons. As discussed in Section 4.2, the transmission probability is a function of the number of *Providers* in the network. This probability decreases as the number of *Providers* increases, and hence the probability of the *Requester* to receive a *Result* beacon remains at the same level.

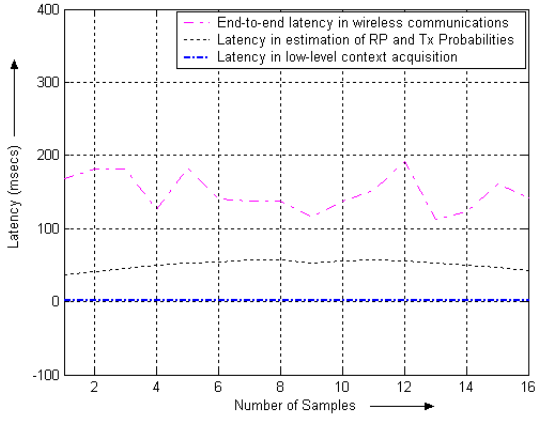

Figure 6. Latencies in R-CDP

Latency:

Figure 6 shows the latencies in various phases of R-CDP.

The latencies caused by the various components are uniform over time. It follows from this figure that R-CDP experiences a small increase in latency when compared to S-CDP. The increase in latency is caused by the estimation of the RP and transmission probabilities. However, the latency in estimation of RP and transmission probabilities, in the *Provider* is small in comparison to the end-to-end latency in wireless communications. Therefore, the end-to-end latency in wireless communications, which would occur in any kind of context discovery, remains the bottleneck in the process of context discovery.

The results shown here are from the initial experiments we performed on R-CDP. Simulations of R-CDP with large number of nodes are needed to evaluate the scalability of R-CDP.

7. Conclusions and Future Work

We have presented an R-CDP for ubicomp environment. It is a lightweight, adaptive, and energy-efficient protocol for discovering contexts for SA application software in ubicomp environments. It has been implemented and evaluated in RCSM. The experimental results have shown that R-CDP can achieve energy-efficiency with a small overhead in latency. The comparison of experimental results as the numbers of *Requesters* and *Providers* vary also show that R-CDP can adapt to variations in network density. The design of R-CDP supports device mobility by dynamically discovering and continuously updating context *Providers* and *Requesters*. The design of R-CDP is lightweight because it does not require the usage of resource-consuming mathematical computation and complex algorithms for collaborations among devices. Future work on R-CDP includes real-time context delivery, implementation of R-CDP in reconfigurable FPGA to improve performance, inclusion of mechanisms to authenticate context data, *Requesters*, and *Providers* in order to enhance security, and investigation of other effective energy-efficient methods for coordinated context discovery among devices.

Acknowledgment

This research is supported by National Science Foundation under grant number ANI 0123980. We would like to thank Microsoft Research for donating part of the PDAs used in our experiments.

References

[1] S. S. Yau, Y. Wang, and F. Karim, "Development of Situation-Aware Application Software for Ubiquitous Computing Environments", *Proc. 26th IEEE Int'l Computer Software and Applications Conf. (COMPSAC 2002)*, August, 2002, pp. 233-238.

[2] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S.Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," *IEEE Pervasive Computing*, 1(3), July-September 2002, pp.33-40.

[3] S. S. Yau and F. Karim, "An Energy-efficient Object Discovery Protocol for Context-Sensitive Middleware for Ubiquitous Computing", *IEEE Trans. on Parallel and Distributed Systems*, vol. 14(11), November, 2003, pp. 1074-1084.

[4] S. S. Yau and F. Karim, "A Context-Sensitive Middleware-based Approach to Dynamically Integrating Mobile Devices into Computational Infrastructures", *Jour. Parallel and Distributed Computing*, vol. 64(2), February 2004, pp. 301-317.

[5] S. S. Yau and F. Karim, "An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments," *Real-Time Systems*, 26(1), 2004, pp. 29-61.

[6] H. C. Hsiao, C. H. Hou, and C.T. King, "A Context Discovery Middleware for Context-Aware Applications with Heterogeneous Sensors", *Technical Report,* 2002, Parallel and Distributed System Lab, National Tsing-Hua University, Taiwan.

[7] J. Kulik, W. Rabiner, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", *Proc. Int'l Conf. on Mobile and Computing and Networking*, August, 1999, pp. 174-185.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. 6th Ann. Intl Conf. on Mobile Computing and Networking (MobiCOM2000)*, 2000, pp. 56-67

[9] T. Imielinski, and S. Goel, "DataSpace – Querying and Monitoring Deeply Networked Collections in Physical Space", *Proc. ACM/IEEE Int'l Conf. on Mobile and Computing and Networking on Physical World*, October 2000, pp. 4-9.

[10] S. S. Yau, S. Gupta, F. Karim, S. Ahamed, Y. Wang, and B. Wang, "A Smart Classroom for Enhancing Collaborative Learning Using Pervasive Computing Technology", *Proc. 6th WFEO World Congress on Eng. Education & 2nd ASEE Int'l Colloquium on Eng. Education (ASEE2003)*, June 2003.

[11] C. Olston, and J. Widom, "Best-effort Cache Synchronization with Source Co-operation", *Proc. ACM SIGMOD Int'l Conf. on Management of Data 2002*, June, 2002, pp.73-84.