October 2015

MIDDLEWARE INFRASTRUCTURE FOR PREDICATE DETECTION IN ASYNCHRONOUS ENVIRONMENT

NAGMA FATIMA

Department of Computer Science and Engineering, Institute of Technology and Management, GIDA, Gorakhpur, India, nagmafatima8@gmail.com

SHIPRA SRIVASTAVA

Department of Computer Science and Engineering, Institute of Technology and Management, GIDA, Gorakhpur, India, shiprasrivastava@gmail.com

SONAL SHARMA

Department of Computer Science and Engineering, Institute of Technology and Management, GIDA, Gorakhpur, India, sonalsharma@gmail.com

Follow this and additional works at: <https://www.interscience.in/ijcct>

Recommended Citation

FATIMA, NAGMA; SRIVASTAVA, SHIPRA; and SHARMA, SONAL (2015) "MIDDLEWARE INFRASTRUCTURE FOR PREDICATE DETECTION IN ASYNCHRONOUS ENVIRONMENT," *International Journal of Computer and Communication Technology*. Vol. 6 : Iss. 4 , Article 16.

DOI: 10.47893/IJCCT.2015.1323

Available at: <https://www.interscience.in/ijcct/vol6/iss4/16>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

MIDDLEWARE INFRASTRUCTURE FOR PREDICATE DETECTION IN ASYNCHRONOUS ENVIRONMENT

NAGMA FATIMA¹, SHIPRA SRIVASTAVA² & SONAL SHARMA³

^{1,2&3}Department of Computer Science and Engineering, Institute of Technology and Management, GIDA, Gorakhpur, India
Email:nagmafatima8@gmail.com

Abstract—Context-awareness is an essential feature of pervasive applications, and runtime detection of contextual properties is one of the primary approaches to enabling context awareness. However, existing context-aware middleware does not provide sufficient support for detection of contextual properties in asynchronous environments. The contextual activities usually involve multiple context collecting devices, which are fully-decentralized and interact in an asynchronous manner. However, existing context consistency checking schemes do not work in asynchronous environments, since they implicitly assume the availability of a global clock or rely on synchronized interactions. To this end, we present the Middleware Infrastructure for Predicate detection in Asynchronous environments (MIPA), which supports context awareness based on logical time. Design and Structure of MIPA are explained in detail.

Keywords-Middleware,Pervasive,Predicate,Asynchronous

I. INTRODUCTION

Middleware: Middleware is defined as a set of services that facilitate the development and the deployment of distributed systems in a heterogeneous networking environment. In the context of this paper, we further narrow this definition to be the software substrate which enables transparent remote invocations of services. The design requirements for middleware are still very complex because they cover two orthogonal middleware characteristics, identity coupling and temporal coupling, among applications requesting services (clients) and ones providing computing services (servers). Identity coupling characterizes how much clients and servers know about each other, and temporal coupling characterizes the degree of synchrony of the message exchange between them. Traditional RPC-based middleware, such as DCOM, CORBA, and Java RMI, exhibits strong identity and temporal coupling. Clients and servers of publish/ subscribe middleware, on the other hand, do not know about each others' identities and do not synchronize when

Communicating either[16].

Pervasive applications are typically context-aware, using various kinds of contexts, such as location and time, to provide smart services [1] [2]. Context-aware applications need to monitor whether contexts bear specified property, thus being able to adapt to the computing environment accordingly. This brings the primary issue of contextual property detection. Though detection of contextual properties has been widely studied in pervasive computing and software engineering communities, it still remains a challenging issue, mainly due to the following two observations [1]. Contextual properties of concern to the context-aware applications bear great variety and dynamism [1][2]. Specifically, users are not only interested in local contextual properties, which can be

easily obtained by singular context collecting devices, but also interested in global ones, which involve multiple decentralized devices for context collection[1]. Meanwhile, users are not only interested in static properties of contexts. Though static properties capture interesting aspects of contexts, they inherently lack the delineation of temporal and relative order. In many cases, users are also interested in behavioral properties, delineating temporal evolution of the environment [1].

In contrast to the variety and dynamism of contextual properties, the detection is greatly complicated by the intrinsic asynchrony in pervasive computing environments. Specifically, context collecting devices do not necessarily have a global clock. They heavily rely on wireless communications, which suffer from finite but arbitrary delay. Moreover, due to resource constraints, context collecting devices (usually resource-constrained sensors) often schedule the dissemination of context data. The different context update rates also result in asynchrony. In order to achieve context-awareness in asynchronous environments, the concept of time needs to be reexamined. Instead of assuming the availability of global time or synchronous interaction, we should rely on logical time. The basic rationale behind is to utilize the happen-before relation resulting from message causality and its "on the fly" coding given by logical vector clocks. However, existing context-aware middleware does not provide sufficient support for detection of contextual properties in asynchronous environments [1].

Middleware Infrastructure for Predicate Detection in Asynchronous Environment.

We develop the Middleware Infrastructure for Predicate detection in Asynchronous environments (MIPA) [1] [2]. MIPA is the first open-source context-aware middleware, which provides systematic support for coping with the asynchrony

while achieving context-awareness in pervasive computing environments, as far as we know. Based on MIPA, users can flexibly specify contextual properties by different types of predicates defined over asynchronous pervasive computing environments. MIPA accepts such predicates and supports context-awareness by online predicate detection [1].

MIPA aims at supporting the development and deployment of various predicate detection-based contextual property detection schemes for different pervasive computing scenarios [2].

Some of our work based on MIPA is as follows:

- Runtime Detection of the Concurrency Property in Asynchronous Pervasive Computing Environments.
- Detection of Behavioral Contextual Properties in Asynchronous Pervasive Computing Environments. A(H1N1a0a(H(@0H1019N10)22009009
- A Lattice-theoretic Approach to Runtime Property Detection for Pervasive Context.

A comprehensive case study is conducted to evaluate MIPA. In the case study, They implement over MIPA a smart lock scenario. The evaluation results show the cost-effectiveness and scalability of MIPA in pervasive computing scenarios [1].

Testing and monitoring distributed programs running on a distributed system involve the basic task of detecting whether a predicate holds during the execution. For example, a software engineer might want to detect the predicate “variable x has changed to value 2” to find out at what point in the execution x takes on a bad value. Another example arises in the area of safety-critical systems, where engineers need to maintain a certain invariant state of the system or guarantee a certain order in which events happen. In all these cases, predicate detection can provide information on whether certain conditions have held during the execution of the program on the system [9]

II. STRUCTURE OF MIPA

The middleware layer is the kernel part of MIPA. Its fundamental functionalities include:

- **Checker process:** The checker process collects vector clock timestamps of local contextual activities. It executes the predicate detection algorithm to decide whether the application-specified consistency constraint is satisfied. The checking result is sent back to the application via the predicate broker [2].
- **Non-checker process:** The non-checker process monitors the local predicate value based on the Event-Condition- Action (ECA) mechanism. It corresponding sensor agents. accepts source contextual events from the

Figure1. Structure of MIPA

The local predicate serves as the event condition. When value of the local predicate changes, the consistency checking algorithm on the checker process side is triggered. The non-checker process sends messages to build the requisite happen-before relationship. It also sends checking message to the checker process, which finally decides whether the consistency constraint is satisfied.

- **Predicate broker:** The predicate broker accepts consistency constraints specified by the context-aware application. It first parses the consistency constraint, and then initiates the non-checker processes and the checker process accordingly.

III. MAPPING CHARACTERISTICS OF PROPERTY DETECTION TO MIDDLEWARE STRATEGIES

In this section, we first discuss the characteristics of contextual property detection in asynchronous environments. Then we discuss how such characteristics motivate design of the MIPA. Abbreviations and Acronyms

A. Characteristics of Predicate Detection

The detection of contextual properties is transformed to the detection of logic predicates specified over the contexts. Predicate detection in asynchronous environments has the following salient characteristics:

C1. Dynamic composition of predicate checkers. Predicate detection can be viewed in a top-down manner based on the hierarchy of predicates. In the highest level, detection of a GSE predicate requires detection of the constituent CGS predicates. It also requires that all the CGSs involved form a GSE. Similarly, detection of a CGS predicate requires detection of the constituent local predicates. It also requires all the local states involved form a CGS. Due to the hierarchical structure of predicates, the checker of an upper level predicate can be constructed from checkers for lower level predicates. Moreover, checkers for the constituent predicates may lie on multiple distributed devices. The upper level checker may need to coordinate multiple distributed checkers.

C2. Monitoring the dynamic environment. Checking a local predicate is simple in the sense that it does not require interaction among distributed devices. However, the critical issue is that checking of local predicates must capture dynamic changes in the computing environment.

C3. Integration of new sensors. Users' requirements on context-awareness is open-ended. When new types of context collecting devices are available, users need to integrate such new devices, obtain new types of contexts, and specify predicates over such new contexts.

B. Overview of MIPA Design

Design of MIPA is motivated by the characteristics of predicate detection. From MIPA's point of view, a pervasive computing environment is composed of an application layer, a property detection layer and a context source layer, as shown in Fig. 2. The context source layer persistently collects contexts of concern. The property detection layer receives contexts from multiple decentralized and asynchronous context sources, and detects specified contextual properties at runtime. The key components of MIPA are outlined below. Predicate detection in groups. Motivated by C1 discussed above, i) MIPA supports composition of lower level predicate checkers to obtain higher level checkers; ii) MIPA supports distributed deployment of the checkers. This is achieved by grouping of different checkers. Specifically, the grouping can be decided by the predicate structure, i.e., lower level checkers consisting the same upper level checker belong to the same group. We can also further group the checkers according to the network condition Contextual event notification. In order to achieve accurate and timely monitoring of the environment as required in C2, we adopt the Event-Condition-Action (ECA) mechanism. Changes in the environment are modeled as contextual "events". Local predicates are interpreted as the "condition" to filter raw contextual events. Non-checker process serves as the event listener ("action"). Pluggable sensor agents. To ease the inevitable process of integrating new types of context collecting devices, as required in C3, MIPA treats the sensor agents (in charge of manipulating the hardware sensors) as plug-ins[1].

Figure2. System architecture of MIPA

IV. RELATED WORK

A lot of work has been done in the area of context-aware computing in the past few years. Seminal work has been done by Anind Dey, et al. in defining context-aware computing; identifying what kind of support was required for building context aware applications and developing a toolkit that enabled rapid prototyping of context-aware applications. While the Context Toolkit does provide a starting point for applications to make use of contextual information, it does not provide much help on how to reason about contexts. It does not provide any generic mechanism for writing rules about contexts, inferring higher-level contexts or organizing the wide range of possible contexts in a structured format.

In [6], Jason Hong, et. al., make the distinction between a toolkit and an infrastructure. An infrastructure, according to Hong, is a well-established, pervasive, reliable set of technologies providing a foundation for other systems. Roy H. Campbell, et al proposed middleware for context-awareness builds on Hong's notion of an infrastructure and provides a foundation for developing context-aware applications easily.

Bouquet, et al. addresses the problem of contexts in autonomous, heterogeneous distributed applications, where each entity has its own notion of context depending on its viewpoint. To interact with other entities, an entity should know the relationship between its viewpoint and other entities' viewpoint. Their middleware uses ontologies to achieve this inter-operability in a more generic fashion. Paul Castro and his colleagues have worked on developing "fusion services" which extract and infer useful context information from sensor data using Bayesian networks. Their middleware provides a more generic framework where such learning approaches can be used. Terry Winograd compares different

architectures for context and proposes one that uses a centralized Event Heap. Their system, however, provides a framework where distributed reasoning can take place. In, Brumitt, et al describes their experiences with multi-modal interactions in context-aware environments and how such an environment can respond automatically to different contexts. Their middleware provides an easy way for developers to specify how an environment should automatically respond to different contexts [5].

Recently, a variety of context-aware middleware have been developed, such as Gaia [7] and Cabot [8]. Based on such middleware infrastructures, a number of contextual property detection schemes have been proposed for pervasive context, such as [7]. However, such schemes implicitly assume that contexts being checked belong to the same snapshot of time, thus not working in asynchronous environments.

In 2010, Jianping Yu proposed MIPA supports contextual property detection in asynchronous environments. Coping with the asynchrony has been a critical issue in pervasive computing. In his pioneering work, Anind Dey pointed out that computing devices must share the same notion of time and be synchronized to the greatest extent possible. However, in some cases, just knowing the ordering of events or causality is sufficient [2]. In [8], the possible faults which can be caused by asynchronous update of context data is investigated, but it is not discussed how to cope with the asynchrony. Memory consumption (ECA-side) the design and evaluation of property detection algorithms for CGS and GSE predicate. These works are implemented as property detection services over MIPA. Development of MIPA is based on one of our ongoing research projects 2. A preliminary report based on version 0.3 of MIPA is presented by Y. Huang. In this work, we significantly extend MIPA in the following aspects. In this work, we add the property detection manager, which supports combining CGS checkers and obtaining GSE checkers. They also add detailed design of pluggable sensor agents. Moreover, they add a smart lock scenario to explain the operation of MIPA, and the scenario is implemented over MIPA using both CGS and GSE predicates.

Jiannong Cao, et al. can be positioned against two areas of existing work: context-aware computing and detection of global predicates in distributed computations. As for context-aware computing, in [8], properties were modeled by tuples, and property detection was based on comparison among elements in the tuples. In [3], contextual properties were expressed in first order- logic, and an incremental property detection algorithm was proposed. In the existing work, temporal relations among the collected contexts are not sufficiently considered. In asynchronous computations, the concept of time must be carefully reexamined and logical clocks are

devised to cope with the asynchrony. Chandy et al. studied how to obtain a snapshot, i.e., a meaningful observation, of an asynchronous computation, and then detect stable predicates based on the snapshots.

In [8], Cooper et al. investigated the detection of unstable predicates, which brought combinatorial explosion of the state space. Conjunctive predicates play a key role in detection of unstable predicates and conjunctive predicates under different modalities. Kshemkalyani et al. gave a refinement of the traditional coarse-grained modality classification and corresponding predicate detection algorithms are proposed [3][12]. Middleware for service-oriented communication/collaboration has been developed in projects, such as POPEYE or SPAWN. Similar to RESCUE, these projects are focused on mobile networks and provide a wide spectrum of features, including a flexible handling of underlying networks or storage of service advertisements and discovery requests in distributed tuples spaces. Yet, they are heavy-weight or do not deal with dependability issues. RESCUE, however, aims at reliable communication with maximum responsiveness [11].

In 2009, Yiling Yang et al proposed how to enable formal specification and runtime detection of dynamic properties in asynchronous pervasive computing environments. Toward this objective, the PDAC framework is proposed, which consists of three essential parts: 1) modeling of the temporal evolution of the environment state; 2) specification of dynamic properties; 3) detection of the specified dynamic property. Currently, the PDAC framework only initiates discussions on one potential approach to enabling specification and detection of dynamic properties in asynchronous pervasive computing environments. Many issues still lack further discussions. The future work, they need to investigate how to further reduce the space cost for runtime maintenance of the lattice. We also need to investigate whether PDAC can integrate other existing property detection algorithms.

Various tools, e.g., a GUI for specification of different contextual properties, still need to be implemented to apply the PDAC framework in real context-aware computing scenarios. A more comprehensive and realistic experimental evaluation is also necessary [14].

V. CONCLUSION

In this work, we study how to provide middleware support for achieving context-awareness in asynchronous pervasive computing environments. Toward this objective, our contributions are: i) we introduce runtime detection of contextual properties based on logical time, to cope with the asynchrony in pervasive computing environments; ii) we design and implement MIPA to provide middleware support for runtime detection of contextual properties in asynchronous environments; iii) a comprehensive

case study is conducted to evaluate MIPA. Currently, the MIPA middleware still suffers from several limitations. In our future work, we will investigate how to design a general algorithmic skeleton, to enhance the design and implementation of various predicate detection algorithms. We also need to study how to reduce the message complexity induced by our property detection schemes. Moreover, we will deploy MIPA to multiple distributed devices and conduct more realistic experimental evaluations.

REFERENCES

- [1] Jianping Yu, Yu Huang, Jiannong Cao and , Xianping Tao, "Middleware Support for Context-awareness in Asynchronous Pervasive Computing Environments " in IEEE/IFIP International Conference on Embedded and Ubiquitous Computing 2010,
- [2] Yu Huang, Jianping Yu, Jiannong Cao, Xiaoxing Ma, Xianping Tao and Jian Lu "Checking Behavioral Consistency Constraints for Pervasive Context in Asynchronous Environments"
- [3] C. Xu, S. C. Cheung, and W. K. Chan, "Incremental consistency checking for pervasive context," in Proc. International Conference on Software Engineering (ICSE'06), Shanghai, China, May 2006, pp. 292–301.
- [4] Y. Huang, Y. Yang, J. Cao, X. Ma, X. Tao, and J. Lu. Runtime detection of the concurrency property in asynchronous pervasive computing environments. IEEE Trans. Parallel Distributed Syst., 23(4):744–750, Apr 2012
- [5] Anand Ranganathan and Roy H. Campbell," A Middleware for Context-Aware Agents in Ubiquitous Computing Environments"
- [6] Hong, J. I., et al. "An Infrastructure Approach to Context-Aware Computing". HCI Journal, '01, Vol. 16
- [7] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "A middleware infrastructure for active spaces," IEEE Pervasive Computing, vol. 1, no. 4, pp. 74–83, 2002.
- [8] C. Xu and S. C. Cheung, "Inconsistency detection and resolution for context-aware middleware support," in Proc. ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'05), Lisbon, Portugal, Sep. 2005, pp. 336–345.
- [9] FELIX C. GA'RTNER and STEFAN PLEISCH "Predicate Detection in Asynchronous Systems With Crash Failures"
- [10] . Pierre-Charles David and Thomas Ledoux," An Infrastructure for Adaptable Middleware".
- [11] Lukasz Juszczak," A Middleware for Service-oriented Communication in Mobile Disaster Response Environments".
- [12] Hengfeng Wei^{1,2}, et al" Formal Specification and Runtime Detection of Temporal Properties for Asynchronous Context",2012 IEEE International Conference on Pervasive Computing and Communications, Lugano (19-23 March 2012)
- [13] Y. Huang, X. Ma, J. Cao, X. Tao, and J. Lu. "Concurrent event detection for asynchronous consistency checking of pervasive context", In IEEE Intl. Conf. on Pervasive Computing and Communications (PERCOM'09), Mar 2009.
- [14] Yiling Yang, " Formal Specification and Runtime Detection of Dynamic Properties in Asynchronous Pervasive Computing Environments", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.
- [15] Karabi Konar, "Next Generation Middleware Technology for Mobile Computing".
- [16] Charles Zhang, Dapeng Gao and Hans Arno Jacobsen," Towards Justintime Middleware Architecture", Department of Electrical and Computer Engineering and Department of Computer Science University of Toronto.