Context Aware Framework

Sridevi S., Sayantani Bhattacharya, and Pitchiah R.

Centre for Development of Advanced Computing,
Chennai, India
{sridevis,sayantanib,rpitchiah}@cdac.in

Abstract. In a dynamic environment, one of the challenges in building context aware applications is the communication paradigm between context providers and end applications. Synchronous communication may not be suitable since all the entities may not be live all the time. In this paper we present the architecture of Context Aware Framework developed using publish / subscribe paradigm to provide asynchronous communication between sensor layer which provides the context information and application layer which consumes the context in order to act according to the context. Sensors keep publishing the data in the XML format using Publish-API and Applications subscribe to the interested context using Subscribe-API. The framework also maintains a knowledge base to store context information in OWL (Web Ontology Language) format and does context interpretation, context modeling, rule based reasoning and event notification. It helps in rapid development of context aware applications across various domains.

Keywords: context awareness, publish / subscribe, ontology based middleware, context modeling and reasoning.

1 Introduction

Three important aspects of context are: where you are, who you are with, and what resources are nearby. Context encompasses more than just the user's location, because other things of interest are also mobile and changing [1]. According to Anind K Dey [2], Context is any information that can be used to characterize the situation of an entity. Context Aware Computing is a key aspect of the future computing environment, which aims to provide relevant services and information to users, based on their situational conditions [3]. Building and deploying context aware systems in open, dynamic environments raises a new set of research challenges. We have developed a framework addressing various challenges of context aware systems to reduce the difficulty and cost of building context aware applications. The rest of the paper is organized in the following manner: - Section 2 elaborates the architecture and sub systems of Context Aware Framework, Section 3 describes about implementation of publish / subscribe paradigm in the framework, Section 4 explains about the framework being used in a medical application, Section 5 shows the result of performance test of the Context Aware Framework, Section 6 gives an insight about related works and finally Section 7 concludes with our future work plan.

2 Architecture and Sub Systems

Fig. 1. shows the overall architecture of Context Aware Framework. It has three sub systems: - Context Management, Device Management and Service Management. Context Management sub system interacts with the Service Management sub system to know the available services and with the Device Management sub system to deliver the required services to the applications based on context. In our model, contexts are described using ontology written in OWL. Modeling context using an ontology based approach allows us to describe contexts semantically in a way which is independent of programming language, operating system or middleware.

Device Management manages the metadata information about the devices (sensor, actuators or any other devices) used in the applications. Sensor has to be registered in to the framework using a Context Builder Tool to provide the metadata information of the data provided by the sensor. Once the sensor is registered, a XML template is auto generated and provided to the sensor layer to publish data. Any new device can be dynamically added to the system. It collects information like devices used type of data the devices transmit, how the data relate to the entity present in the environment, possible commands to control the devices, number of rooms in the environment, the devices available in each room etc. For example Room A may have one temperature sensor and one Air Conditioner (AC). If the temperature sensed by sensor in Room A is very high and AC is off then our framework will send an event notification to the application layer to switch on the AC in Room A. It also provides an interface to create rules for the Environment. Some rules may be generic and can be applied to any environment (e.g. temperature is high then switch on AC). Some rules may be specific depending on application (e.g. upload presentation materials to a computer in

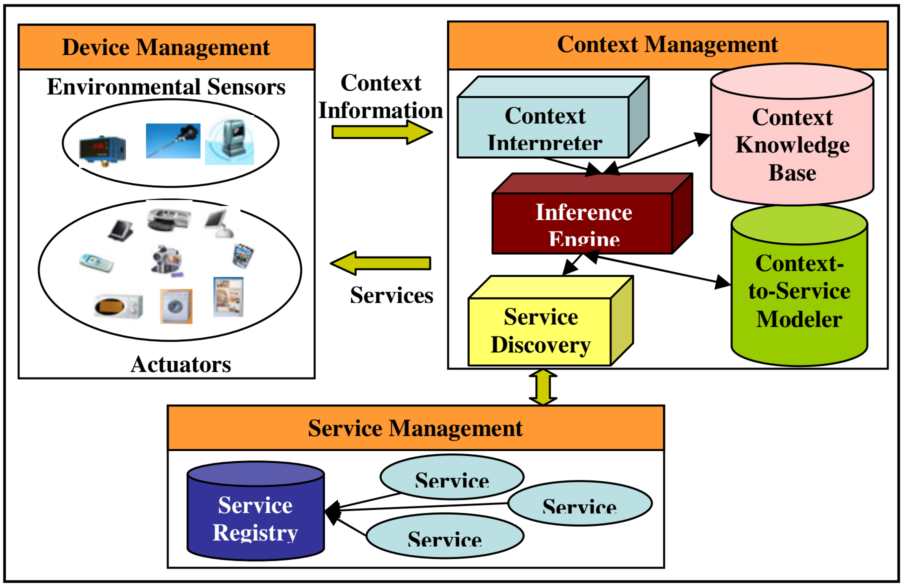

Fig. 1. Architecture and sub systems

a conference room if the speaker enters the room). Rules can be added dynamically and classified as low level (if-then rules) and high level context rules. High level rules are constructed by combining various low level rules.

Service Management maintains a registry for various services available in the framework. Any application can query the context knowledge base and know about the status of any existing context entity. Framework exposes the interfaces in various forms such as Java APIs, Web Services and OSGi services. At the context server end, one can see the context changes of various entities using Entity Watcher Graphical Tool. Framework also contains in-built services for sending SMS, sending E-mail, for knowing the current weather condition, for receiving weather forecast for a location and for getting news updates.

Context Management manages context parameters of various entities such as people, places, smart artifacts / objects etc. Various modules of the context management are the Context Interpreter, Knowledge Base (Rule-base), Inference Engine, Service Discovery and Context-to-Service modeler. The Context Interpreter gathers contextual information about a certain entity from Context Providers (sensors or external sources), aggregates gathered context, converts the gathered context into OWL format and sends to Inference Engine for further processing. Context Knowledge Base is a persistent store which manages the context information in the form of an OWL file and also stores the information about the context provider as meta-data in the XML form. It keeps the context history, since it is essential to allow context inference based on past occurrences. Rule based inference engine is developed using Jena framework along with SPARQL queries. Whenever any context information is given from the Context Interpreter, Inference Engine analyzes the context and decides on the action to be taken using context-to-service modeler. Rule matching is an important task of Inference Engine. Whenever a context of an entity E is changing, it has to fetch all the low level and high level rules defined for the context parameters of E. Once the rules are fetched, it has to find all the matching low level rules. With the matched low level rules, it has to identify matching high level rules, and then decide on the services to be provided to the application layer. Service Discovery module communicates with various service providers or service registries. It checks periodically about the availability of the existing services and also checks for any new services that may have been added. Context-to-Service Modeler is a mapping module in the Context Builder Tool where the context is mapped with relevant services. Context-to-Service modeler finds the list of all context information from Context Knowledge Base, and gets available services in the environment by communicating with Service Discovery module.

3 Publish Subscribe in Context Aware Framework

As distributed systems on wide area networks grow, the demands of flexible, efficient, and dynamic communication mechanisms increase. The publish/subscribe communication paradigm provides a many-to-many data dissemination [4]. In systems based on the publish/subscribe interaction paradigm, subscribers register their interests in an event or pattern of events, and are subsequently asynchronously notified of events generated by publishers [5]. In our framework, Publish / Subscribe paradigm has been developed using RMI Callback and interfaces are exposed for publish, subscribe, unsubscribe operations. Whenever a publisher publishes context information, Context Interpreter parses the published data, searches for a Context Entity E associated and updates the entity E if it exists or creates a new Context Entity E. Then it interprets the context based on the data gathered from various other sensors. If there is any change in the Entity due to the newly published data, then Context Manager generates a Context Event for Entity E and notifies all subscribers of E. Sensors act as publishers and publishes information like health parameters, location of a person, temperature of a room etc. Applications like Medicare, Intelligent Home and Discussion Room subscribe for information required for them. Subscription Manager manages the subscription using a Hash table <K, Vector<S>> where K is a context item and S is the reference of a subscriber. Subscriptions are added and removed by calling subscribe(c, k) and unsubscribe(c, k) operations where c is the connection to context server and k is the interested context item. Whenever there is a change in a context item K, Subscription Manager forwards the event to all the set (Vector<S>) of subscribers of K using the reference S.

4 Usage of Framework in Healthcare Application

Context aware computing is a research field which often refers to healthcare as an interesting and potential area of application [6]. We have developed a Context Aware Health Monitoring System [7] which shows the usage of our framework in health domain. In our experiment, we used Bluetooth enabled wireless medical devices: - Pulse Oximeter, Heart Monitor and Blood Pressure Monitor. Patients visit the Primary Health Care Centres and a nurse records the patient's health parameters using these devices. These health parameters are collected by a mobile phone and transmitted to our framework. Health Service Provider is a service running in the application layer which subscribes for any change in the health parameters. Health parameters are parsed by the Context Interpreter and the Inference engine compares the parsed health data against the Rule-base to find if any abnormities are present. If any abnormality is detected, an event will be triggered and notified to the application layer requesting to send SMS / Email to the domain experts depending on the state of the abnormities.

5 Performance Test on Context Aware Framework

Performance Test has been conducted to test the scalability of the framework and Fig. 2 shows the results of the test. Context Server has been tested on a computer with Intel Pentium 4 CPU @ 3.00GHz, 1.48GB RAM. Sensor Layer and Application Layer are simulated, tested on a computer with Intel Core2Duo E6750 @ 2.64GHz processor, 1GB RAM. Totally 6 sensors were simulated to publish data for various number of context entities at 10 minute intervals in a local area network with 100mbps speed. Time delay in event notification has been recorded after loading 100, 200, 400, 600, 800, 1000 context entities in the context server. Results depict that even though the server is loaded with more number of entities, time delay is only in the order of a few milliseconds.

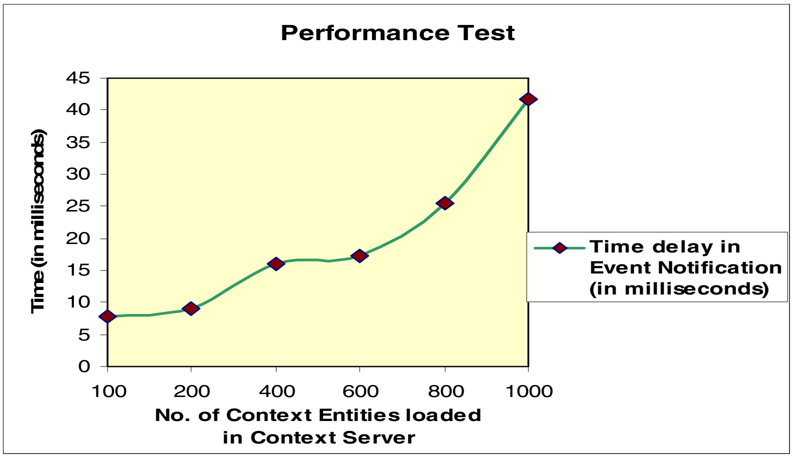

Fig. 2. Performance Test Results

6 Related Work

Java Context-Awareness Framework – JCAF, [8] is a Java-based context-awareness infrastructure and programming API for creating context-aware computer applications. Service-oriented Context-aware Framework [9] generalizes the location-based services to context-based services in mobile or desktop environments. On Context-Aware Publish-Subscribe [10] paper proposes a context-aware extension to publish subscribe model of communication. A Rule Based Publish/Subscribe Context Dissemination Middleware [11] provides the base functionalities for mobile devices to publish their context information, and in addition subscribe to the context information published by their mobile peers. Context Toolkit [12] aids in development and deployment of context aware services. Context Broker Architecture (CoBrA) [3] describes about the architecture for smart spaces. Service Oriented Context aware Middleware (SOCAM) [13] is designed for smart home environment.

Key features of our Context Aware Framework are: - it is an ontology based middleware which can be used for building context aware applications for various domains / applications like health care, intelligent home and smart meeting rooms; It addresses various challenges in developing context aware applications like context data acquisition, context interpretation, context data modeling and reasoning; Context Rules can be dynamically added to the framework. It acts as a communication medium between the sensor layer and the application layer; It is developed using publish-subscribe paradigm and provides event notification service with the actions (turn on/off AC, switch on/off light, send SMS, send E-mail etc.) to be taken based on the context.

7 Conclusion and Future Work

We believe an infrastructure is necessary for building context-aware systems and it should provide adequate support for context modeling and context reasoning. Based on the experience and the feedback on the Context Aware Health Monitoring System developed using the framework, we are improving our reasoning engine with enhanced reasoning capabilities and testing the framework on more metrics like load, reliability, fault tolerance etc. We aim to deploy our framework in buildings to make them smart and energy efficient. Future work includes development of a light weight Java API for Mobile based Context Aware Applications.

References

1. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1994), Santa Cruz, US (1994)
2. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing Journal (2001)
3. Chen, H., Finin, T., Joshi, A.: A Context Broker for Building Smart Meeting Rooms. In: Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, AAAI Spring Symposium (March 2004)
4. Son, H., Li, X.: PARMI: A Publish/Subscribe Based Asynchronous RMI Framework for Cluster Computing. In: Perrott, R., Chapman, B.M., Subhlok, J., de Mello, R.F., Yang, L.T. (eds.) HPCC 2007. LNCS, vol. 4782, pp. 19–29. Springer, Heidelberg (2007)
5. Eugster, P.T., Felber, P.A., Guerraioui, R., Kermarrec, A.-M.: The Many Faces of Publish/Subscribe. ACM Computing Surveys 35, 114–131 (2003)
6. Bricon-Souf, N., Newman, C.: Context awareness in health care: A review. International Journal of Medical Informatics 76(1), 2–12 (2007)
7. Sridevi, S., Bhattacharya, S., Pal Amutha, K., Madan Mohan, C., Pitchiah, R.: Context Aware Health Monitoring System. In: Proceedings of International Conference on Medical Biometrics, ICMB 2010, Hong Kong, June 28-30 (2010)
8. Bardram, J.E.: The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) Pervasive 2005. LNCS, vol. 3468, pp. 98–115. Springer, Heidelberg (2005)
9. Kovács, L., Mátételki, P., Pataki, B.: Service-oriented Context-aware Framework. In: The 4th European Young Researchers Workshop on Service-Oriented Computing, Pisa, Italy (2009)
10. Cugola, G., et al.: On Context-Aware Publish-Subscribe. In: Proceeding of Second International Conference on Distributed Event Based Systems, DEBS 2008, Rome, Italy (2008)
11. Gehlen, G., Aijaz, F., Muhammad, S., Walke, B.: A Rule Based Publish/Subscribe Context Dissemination Middleware. In: Proceedings of Wireless Communications and Network Conference, WCNC 2007, Hong Kong, China, p. 6 (2007)
12. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI Is the Limit, Pittsburgh, Pennsylvania, United States, May 15-20 (1999)
13. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 28 (2005)