A Middleware for Highly Dynamic Distribution in CPS Environment

Jie Sun
Ningbo University of Technology
Ningbo, Zhejiang, China
+86 574 87616023
sunjie@nbut.cn

Yongping Zhang
Ningbo University of Technology
Ningbo, Zhejiang, China
+86 574 87616023
ypz@nbut.cn

ABSTRACT
A cyber-physical system is an integration of physical world devices and cyber-world computing and communications capabilities, making the environment smarter. This paper proposes a middleware that provides interoperability between heterogeneous mobile devices in a CPS networks. The paper defines CPS middleware components and modules that represent the features and characteristics specific to CPSs. The proposed middleware focuses on the decoupling of service requests and service execution for devices in mobile environment.

Categories and Subject Descriptors
D.2.11[**Software Engineering**]: Software Architectures–*Domain-specific architectures*

General Terms
Design.

Keywords
CPS, middleware, interoperability.

1. INTRODUCTION
A Cyber-Physical System (CPS) consists of a network of elements that interacts with physical and computational environment. The physical world consists of smart devices, sensors and actuators with computation, communication, and control capabilities. The increasing diversity and heterogeneity of smart devices has created a need for middleware that provides the interoperability of devices. The interoperability should be achieved in technical, semantic and service aspects by proper integration of communication technology, semantically semantic abstraction technology and service oriented architecture.

The use of conventional middleware to provide device interoperability in a CPS network poses numerous challenges. Since these physical devices have limited resources, it is not possible to execute complex computation and processes. To cope with this challenge, we apply Service-oriented Architecture (SOA). To realize a service-based CPS, we first define overall architecture with three tiers. And then, we present key methods to design each tier in the architecture. A horizontal middleware structure that achieves device interoperability without heavily relying on a single server is required to implement a CPS network.

The paper is structured as follows. In Sec. 2 we discuss selected middleware from the domain of Internet of Things and CPS and general requirements for middleware in the field of intelligent environments. In Sec. 3 we propose the architecture of the middleware and discuss the applicability of this middleware for intelligent environments against the previously identified criteria. The implementation details is discussed in section 4. We conclude our paper in Sec. 5 by giving a short overview on future work.

2. RELATED WORK
Literature [1] has studied the middleware systems that have been applied in Internet of Things (IoT) -based systems. They classify the required functionality of middleware to manage interaction with a variety of devices in four functional components, namely (1) interface protocols, (2) device abstraction, (3) central control, context detection & management, and (4) application abstraction. We therefore investigated the available middleware especially with a focus on heterogeneous support, and service adaption architecture.

SOA is widely applied in IoT and CPS area, which is not just a set of standards, but also a design philosophy developing systems that are loosely coupled, flexible, reusable and adaptable.

SODA [2] is an adaptation of a service-oriented architecture (SOA) which models devices as services. The implementation provides an abstract services model of a device by talking to proprietary and standard device interfaces on the one side and, on the other, presenting device data as SOA services over a network through a bus adapter. A standard specified device service can have a wide variety of underlying hardware, firmware, software, and networking implementations that don't affect the consumer of the service.

LinkSmart [3] is an IoT middleware that support the interoperability and seamless integration of various external devices, sensors, and services. It is developed by combining the service-oriented architecture, peer-to-peer networking, and semantic web services technologies. The SOA and its related standards provide interoperability at a syntactic level. The Semantic Model Driven Architecture (SeMDA) is designed to promote semantic interoperability for on-line services and devices. The SeMDA concept makes all devices accessible in a uniform way - as the semantic web services.

WebMed [4] aims at facilitating the service-oriented architecture for physical devices. The middleware contains high-level, logical representations of physical devices, computing elements and software services that are not necessarily linked to physical devices. WebMed consists of five components: device adapter (named WebMed node), Web service enabler, service repository, engine, and application development.

TinyDB [5] middleware focus on gathering data from devices. It was the first project to propose the idea of abstracting from devices. TinyDB allows end-users to interact with devices without knowing about the details of the devices specification, such as the

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.257522

communication protocols that are supported by these devices. TinyDB he provides a Domain Specific Language (DSL) for end-users to interact with devices. Its DSL is a query language that supports selection, join, projection, and aggregation to work with an embedded sensing environment. TinyDB supports the following types of queries: Monitoring Queries, Network Health Queries, Exploratory Query, Actuation Query. The last kind of query can be used to ask for a physical action. For instance, an end-user of a system wants to turn off a fan in a room.

WISeMid [6] is a middleware which enables the interaction between WSN and Internet. It uses an Interface Definition Language (IDL) to describe a service in this middleware. IDL is a unified language to describe a service irrespective of where (Internet or WSN) or what implementation language is used. The IDL contains a module (package) that is as a container for specifying service interfaces. Each service interface includes name and the operation that can by supported by the service. Each operation contains input/output parameters types and may raise exceptions.

DIM (Dynamic Integration Middleware) [7] is a middleware framework for global CPS networks. DIM classifies devices into controllers and controller managers (CMs). DIM views a global CPS network in two logical layers. The lower layer network is composed of controllers in local networks. The upper layer network is composed of CMs in a global network. DIM achieves interoperability across the heterogeneous platforms by the protocol conversion between different CMs.

All the above discussed projects inherit the benefits of using service-oriented architecture, which is providing a unified method to interact with physical elements, i.e., services, and thus facilitating to build flexible systems while preserving efficiency and scalability. But they do not mention how to meet the service requests in a highly dynamic environment. When users roam from a room to another room, how to find appropriate device in the new room to fulfill the same service? In this paper, we put focuses on the decoupling of service requests and service execution for devices in mobile environment.

3. ARCHITECTURE OVERVIEW

After the investigation of available middleware systems for intelligent environments, we chose a service-oriented middleware to develop a distributed CPS system, which focused on the encapsulation of heterogeneous between variety of physical devices and providing a uniform way for service access. The architecture is shown in figure 1.

3.1 Cyber Layer

This layer aims to provide high-level abstraction to end-users to access physical devices. The platform guarantees the service requested to be executed by some physical device, no matter where and what the device is. Two components are involved:

Invoke engine. It provides a platform-independent description for end-users's requests, such as playing video and editing text, which is an abstract service. The request is received by the Event Processing Module, and dispatched to the execution engine. So allows an end-user to request for execution of a task in the same way in different platforms.

Execute engine. It is responsible for execution of services without knowing about the details of the devices specification, such as the communication protocols that are supported by these devices. It also allows multiple service invocations to run at the same time.

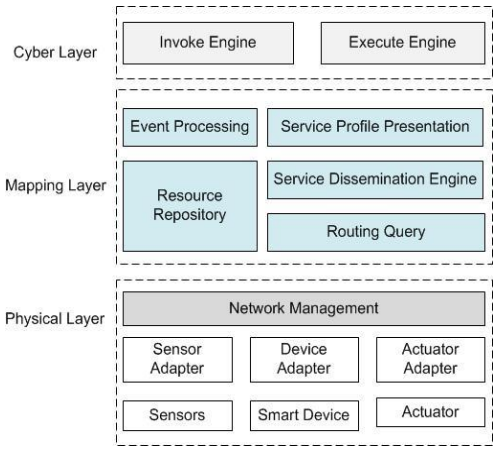

Figure 1. Architecture Overview

Above components are designed for service request/provision decoupling. The service request dissemination has to be transparently routed to all the potential providers connected to the mobile systems. To foster system scalability and service availability, service request and provision can be possible at different times (time decoupling), and do not have to know each other (space decoupling); in other words communication should be asynchronous and anonymous among service producers and consumers. The process is shown in figure 2. It requires capabilities for discovery, deployment, and execution of services in a pre-defined or ad-hoc manner, which is the focus of service dissemination module.

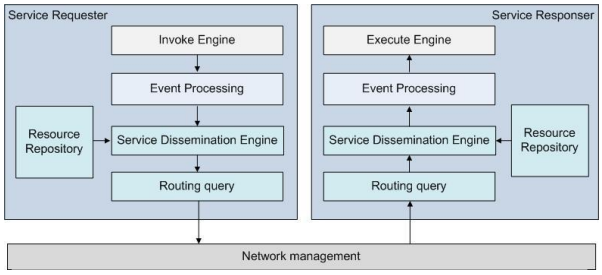

Figure 2. Service Request and Provision Decoupling by Invoke Engine and Execute Engine.

3.2 Mapping Layer

3.2.1 Service Profile Representation

This component is responsible for encapsulating devices as services, dealing with the device-specific connections and protocol as well as with network interfaces needed to publish the data over a defined SOA protocol. A standard specified device service can have a wide variety of underlying hardware, firmware, software, and networking implementations that don't affect the consumer of the service. The description of a service, called an interface, lists its inputs and outputs, explains the provided functionality, and describes non-functional aspects of execution.

Services are represented using DPWS (Devices Profile for Web Services) [8], which is ideal to build seamless integration between mobile devices and sensor networks. The definition of the service is adopted from the formal specification of DPWS version 1.1:

Definition: A service is a set of related ports. A service port is represented by a set of operations $S_p = \{s_1, s_2, \ldots, s_n\}$. Each port is a realization of a port type. A port type is an abstract collection of operations that describe actions and related message exchange patterns. A message is an abstract, typed definition of the data being communicated.

A service port and a corresponding operation will be expressed as:[device: client|server]service.port.operation[message exchange patterns] (inparameters, outparameters).

For an example, there are several operations that can be performed with a fan: TurnOn, TurnOff, SpeedUp, SpeedDown, etc.

3.2.2 Service dissemination engine

This component is responsible for service discovery. It manages which available device provides needed services, and where services can be deployed. According to different physical resources available, it supports different ways to execute the service request. To facilitate the access for resources, this component encapsulates the detailed information about execution in a distributed environment.

The service dissemination supports mobile heterogeneous wireless scenarios. Mobile nodes requiring people and devices carried on them move in and out, sometimes randomly, by also introducing sudden variations. Hence, the service dissemination has to promptly adapt to mobility. At the same time, this function has to comply with heterogeneous systems including nodes with different computational capabilities, wireless standards, and wireless modalities; hence, the adaptation to currently available resources is fundamental to avoid system saturation.

By changing the context of the end-user such as the location, the deployment of the supplier in the new location can be changed. The engine provides a run-time adaptation support. The dynamic management is based on the current run-time conditions, e.g., deployment environment and monitored resource conditions, which is the responsibility of the resource repository.

3.2.3 Resource repository

This component monitors the change of resources such as sensors, actuators and smart devices to achieve adaptation to mobile situations. There are four kinds of physical resources in the environment and the repository should include the information such as:

- Actuator: This kind of query can be used to ask for a physical action. For instance, an end-user of a system wants to turn off a fan in a room when the temperature of the room is lower than a threshold.
- Device: It shows the status of a specific device or a set of device at a specific time。
- Sensor: It asks the value of one or more attributes periodically and continuously, such as, e.g., reporting the temperature of a warehouse every hour.
- Network: it shows the status of the network, and reports the change of topology and the failure of the network nodes. It also encapsulates the communication details.

3.2.4 Routing query

According to the resource repository, the requested service can be executed locally or remotely. The routing query module will maintain a list of resources available including local and remote. This module will lead the request to appropriate route from the current node to the responde node, where the execute engine will call the application on a certain device to responde the service request. The resource repository enables the module to automatically find devices

3.3 Physical Layer

This layer aims to manage the different physical resources and the network they connected. A device adapter standardizes the heterogeneous devices' hardware, data structures, communication protocols and device control issues. It is responsible for consolidating underlying devices' data into a common model. It is the point of entry for the devices to our middleware. Besides, it acts as an aggregator when adding/removing newly connected/ disconnected physical devices to the middleware. Once a device is connected to the middleware, the adaptor will register to the resource repository so that the other components can recognize it as a new component.

The network consists of all kinds of wired/wireless networks such as WSN (wireless sensor network), 3G, or smaller sub-networks. Each sub-network can contain an aggregator that in turn connects to the Internet gateway (router). Examples of such sub-networks are Zigbee sub-networks (electrical appliances, air conditioner (AC)), Wi-Fi sub-networks (laptop, printer, and media server), UWB sub-networks (HDTV, camcorder), smart grid sub-networks (smart meters, smart thermostat, smart switch), body area sub-network (smart phone, monitoring instrument, body sensors) and Bluetooth sub-network (music center, portable audio player). Possible aggregators include a cellular phone for the body area subnet and power meters for the smart grid subnet.

The networked devices expose their functionality directly as web services using DPWS. Each device has one or more device control interfaces that can be accessed by software. These interfaces are wrapped as a method of service.

The physical characteristics of the devices, such as device ID, location, processing power, memory and battery status etc., and the services they offer are useful in selecting the appropriate device. For simplicity these characteristics are not included and displayed in the proposed architecture; rather they are stored separately in the resource repository.

4. DISCUSSION

We divide the devices into two kinds: controlling devices and controlled devices. Sensors, actuators and appliances are controlled devices and they can provide services for specific purpose. While smart phone, laptap and PDA are controlling devices for they can send command to controlled devices. They are controlled devices in the same time for the services they can provide.

When a controlled device is added to the system, it will advertise its services on the network and then its information will be added into the resource repository. Similarly, when a controlling device enters the network it performs a search for controlled devices and a search request is sent out and then the devices that match the request send a corresponding reply. Usually, searching and advertising are implemented using multicast messages, whilst replies from devices are unicast.

Once a controlling device with a specific service request has discovered a controlled device in the resource repository, the controlling device will retrieve the controlled device's description from a location indicated by the routing engine. For each service exposed by a device, the device description defines the actions and the message formats.

Multiple accesses of the exclusive devices must be avoided. The device service will denies requests if the device is occupied.

5. CONCLUSION AND FUTURE WORK

We have presented a middleware that provides interoperability between heterogeneous mobile devices in a CPS networks. The middleware is service-oriented to coordinate the computational and physical parts of a system. The benefits of using service-oriented architecture will provide users a unified access method to interact with physical and software elements, and facilitate to build flexible systems while preserving efficiency and scalability. Currently the device services are atomic services and our immediate goal is to develop composite services to make more complex control.

6. ACKNOWLEDGMENTS

This work is supported by NSFC of China (Grant No.61203360), NSFC of Zhejiang Province, China (Grant No. LY13F020014, Y14F010027, Q14F010007).

7. REFERENCES

[1] Bandyopadhyay S., Sengupta M., Maiti S., and Dutta S. 2011. Role of Middleware for Internet Of Things. International Journal of Computer Science & Engineering Survey (IJCSES), 2, 3, 94–105.

[2] Scott de D., Randy C., Kevin E. K., Bill M., and Jeffrey R. 2006. SODA: Service-Oriented Device Architecture. IEEE Pervasive Computing, 5,3, 94-96.

[3] Badii A., Crouch M., and Lallah, C. 2010. A context-awareness framework for intelligent networked embedded systems. In *Proceedings of* Third International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services(CENTRIC), 105 – 110.

[4] Dat Dac H., Hye-Y. P. C.K.K. 2012. Service-Oriented Middleware Architectures for Cyber-Physical Systems. International Journal of Computer Science and Network Security, 12, 1, 79–87.

[5] Madden S. R., Franklin M. J., Hellerstein J. M., and Hong W. 2005. TinyDB: An acquisitional query processing system for sensor networks. ACM Transactions on Database Systems (TODS), 30, 1,122–173.

[6] Domingues J., Damaso A., Nascimento R., and Rosa N. 2011. An Energy-Aware Middleware for Integrating Wireless Sensor Networks and the Internet. International Journal of Distributed Sensor Networks, 2011, 1–19.

[7] Sang O. P., Jong H. P.,Young S. J. 2013. An Efficient Dynamic Integration Middleware for Cyber-Physical Systems in Mobile Environments. Mobile Netw Appl ,18, 110–115.

[8] OASIS Standard. 2009. Devices Profile forWeb Services (DPWS). Available at http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01.