

Lightweight Web-based Communication Interface Design For Web of Objects

Sanghong An, Sangmin Park, Hyeontaek Oh, Jinhong Yang, Hyojin Park, Junkyun Choi

Department of Electrical Engineering

KAIST

Daejeon, Republic of Korea

{ancom21c, deadpk, hyeontaek, sunupnet, gaiaphj1, jkchoi59}@kaist.ac.kr

Abstract—With recent advance of processors and communication technologies, the accessibility toward World-wide-web (WWW) gets better. Recently, there are many tries to integrate devices and things to the Internet. The purpose of Web of Objects is to enhance users life and experiences by providing rich web service with sensors and actuators. The difference from legacy device-connected service is that it follows the webs virtue loose coupling of service components. The loose coupling is important for the dynamic changes of service such that new device component is added on the service or legacy device gets apart from the service. For loose coupling of these components based on the web technologies, it requires lightweight service logic and bi-direct communication interfaces for limited devices like sensors. Each component works not only as server to deal with requests on it, but also as clients to send requests to other components. To support various devices with various capacity and protocols, the interfaces should be configured based on lightweight pure web technologies. In this paper, the lightweight web-based communication interface will be introduced. We designed, configured and implemented this interface for limited device. It is based on the RESTful APIs to react the dynamic changes of service configuration.

Index Terms—lightweight communication, web of objects, sensors on web, web intents

I. INTRODUCTION

With recent advance of processors and communication technologies, the accessibility toward World-wide-web (WWW) gets better. Recently, there are many tries to integrate devices and things to the Internet, such as Web of Objects (WoO). The purpose of Web of Objects is to enhance users life and experiences by providing rich web service with sensors and actuators. The difference from legacy device-connected service is that it follows the webs virtue loose coupling of service components. The loose coupling is important for the dynamic changes of service such that new device component is added on the service or legacy device gets apart from the service. The device-connected services like fire alarm service, finding missing child service, are composed of four web components sensor part that detects circumstance changes, actuator part that do something against the circumstance, controller part that process the data, and the service user client.

For loose coupling of these components based on the web technologies, it requires lightweight service logic and bi-direct communication interfaces for limited devices like sensor modules. Each component works not only as server to deal

with requests on it, but also as clients to send requests to other components. For example, the controller part could request working status and measurement value to the sensor part. In this case, the sensor part works as server, and the controller part works as client. If the sensor detects the urgent changes of the measurement value and need to report to the controller part, then sensor part works as client and the controller part works as server. To support various devices with various capacity and protocols, the interfaces should be configured based on lightweight pure web technologies.

There are some requirements for web of devices to realize kinds of explained works. The first requirement is light-weight. The network stacks, related software interfaces, and server/client implementation for web access should be light-weight to be operatable on contrained devices. CoRE(Constrained RESTful Environments) working group of IETF defined classes of constrained device as following: class-2 for devices that work well with HTTP, which have about 50KB RAM and 250KB code spaces, class-1 for devices that is not working with HTTP, TLS, XML, which have about 10KB RAM and 100KB code spaces, and class-0 for devices that are more constrained than class-1 devices [1]. For web environments that work on class-0 and class-1 devices, the architectures should be simple. The second requirement is real-time bidirectional web communication. In web environments for devices, it is needed to support various scenarios such as alarming and pre-processing for context-awaring. Sometimes techniques such as server pushing is required for device-to-device, device-to-web service, and device-to-user communication. Another requirement for web of devices is easy-to-mashup. Web for devices is a distributed computing environment that sensors and actuators are spreaded on the network. The application should be able to receive sensor data and control actuators with common web interfaces. Therefore, it is required that the web communication interface which is suitable for limited and distributed network and inter-operatable with legacy web environments.

In this paper, the lightweight web-based communication interface will be introduced. We designed and implemented this interface for limited device by extending web intents. It is based on the RESTful APIs to react the dynamic changes of service configuration.

II. RELATED WORKS

Recently, there are many researches to deploy web environments on the limited devices. The researches related to the requirements of web for devices are following.

A. Light-weight protocols

CoAP(Constrained Application Protocol) is a RESTful application protocol to meet M2M requirements, which is proposed by CoRE working group [2]. It supports asynchronous transactions, light-weight header and parsing, URI and content type, resource discovery. It uses a proxy for mapping HTTP and CoAP communication, and it is called stateless HTTP mapping. CoAP uses UDP as transport layer, while HTTP uses TCP. Therefore, the network with CoAP should support both unicast and multicast on the UDP environments. CoAP REST uses the HTTP-like basic methods of GET, POST, PUT, and DELETE. The representation of resources that used for HTTP request/response is URI, which is compressed for limited devices.

B. Bi-directional realtime communication

1) *Comet and WebSocket* : Comet, the server-side Ajax is a web application model with long-held HTTP request that allows a web server can push data to the browser, without any plug-ins [3]. The implementation of comet relies on basic features of browsers. The comet approach is different from the original web model that the browser requests a complete web page at a time. With comet technology, a server can push data to browsers that are accessing that server. Another approach is WebSocket [4]. The WebSocket protocol also makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time games. Compared to comet, this is made possible by providing a standardized way for the server to send content to the browser, and allowing for messages to be passed back and forth while keeping the connection open.

For real-time communication between browser and server, WebRTC(Web Real-Time Communication) standard is being defined by IETF and W3C collaboration [5]. It provides direct interactive rich communication using audio, video, collaboration, games between two peers' web browsers.

But many of these researches are about server to browser communication. It is required that the real-time communication methods for web of distributed devices.

C. Web Intents

Web intents is a framework for web-based inter-application communication and service discovery. [6] It is modelled after Intents system in Android, which includes message object that describes the action of application components. In Android, intent is used to data exchange between application components.

Web Intents also consists of a discovery mechanism and light-weight RPC system between web applications. It allows two web applications to communicate with each other without actually recognize each other. To use intent in action, the service provider needs to register the web intents on the web

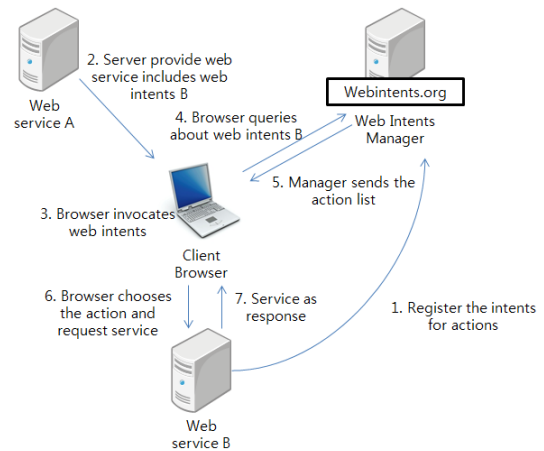


Fig. 1. Procedure of Web intents

intents manager, which can parse the intents and suggest the application to clients. The mash-up application maker can use tag of <intent> in his web document to register. To invoking an action, maker can create new web intent object and call the starting function with javascript. there are six intents is defined for now, such as share, edit, view, pick, subscribe, and save.

Web intents' philosophy is suitable for future web of objects. It makes easy to mash-up among web services and data exchanging with light-weight RPC system.

III. WEB INTENT EXTENSION FOR DEVICES:LIGHTWEIGHT WEB-BASED COMMUNICATION INTERFACE DESIGN

Web intents is originally designed for web browsers, which is a launcher of web applications. Figure 1 shows the procedure of web intents. But in web of objects environments, there is no browsers for devices. Each node can be a server or a client in web of objects environments. Each node in web of objects can be mapped a application component of Android intent model. Therefore it is needed to modify web intents model for devices.

A. Network Environments

The web of objects environment consists of sensor/actuator nodes, web service servers, and user clients. sensor and actuator nodes communicate to web service servers, user clients, and each other. Sensor node gathers its own data and reports when it meets its own rule. Actuator node gets receive control message from web service servers, user clients, or other sensor/actuator nodes and do something toward the world. Sensor and actuator nodes' architecture is described in Figure 3. It has IP-based network stack, and it is supposed to run HTTP, Javascript stacks on its processor.

Web service server mediates sensor/actuator nodes and user clients to provide proper service such as fire alarm/escape path notification service and pet care service. Web service could be composed of distributed computing of sensor nodes and servers. User clients uses web browsers to monitor nodes and use the web service.

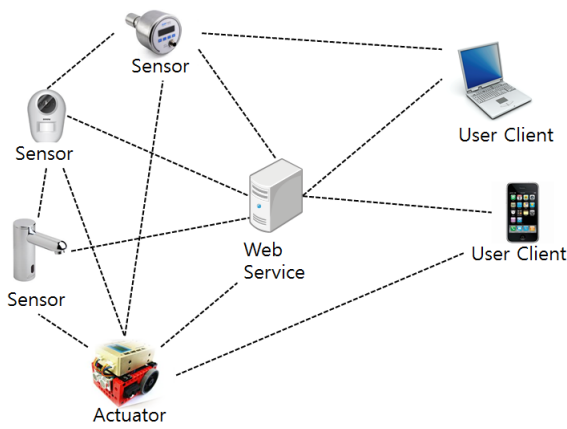


Fig. 2. Web of Objects Environments

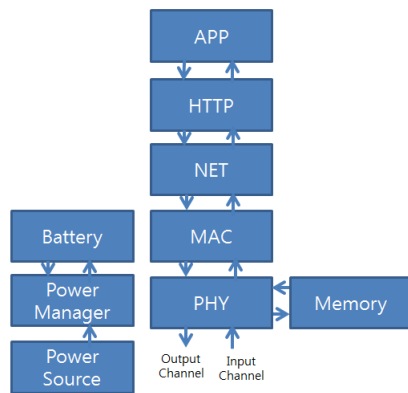


Fig. 3. Sensor/Actuator Node Structure

B. Web intents model for devices

Current model of web intents is for browsers and human, which means that human intelligence is working for choosing proper intent action for the first time launching. If there are several actions for one intent, then a user can choose proper action. However these series of works are very hard and resource-consuming job for limited devices. A device has no idea to choose the action. But in machine-to-machine distributed network model, it is not necessary to do one action for an intent. Web intents will be used to invoke another nodes' action and transfer data.

Therefore, in web intents model for devices, discovery and negotiation phase replaces the action decision phase for conventional model. If there are more than one actions for an intent, then the device starts to negotiate each action's owner(intent receiver) to communicate. The negotiation rule needs to be defined on intent receivers when it register its action to an intent. This negotiation phase seems to be inefficient if there are many actions for an intent. But once the negotiation is done, the next time of intent invocation will follow the previous rule.

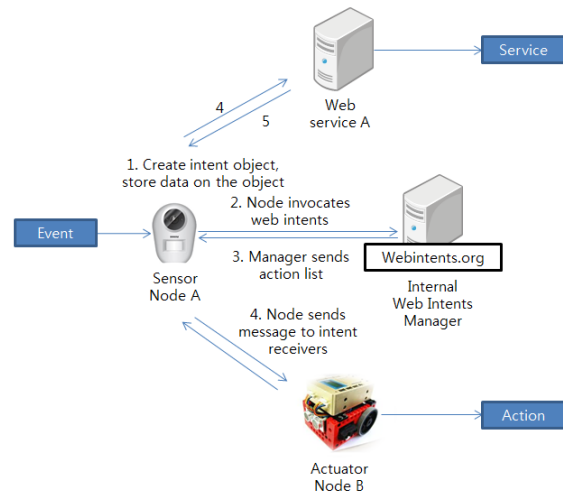


Fig. 4. Procedure of Web intents on device network

IV. APPLICATION SCENARIOS

With web intents model on device network, a new web service using devices can be composed easily and newly added device nodes can be configured for a web service fast. Here are some application scenarios using web intents for devices.

A. One sensor event, multiple action domains

In real application, a sensor event can invoke a series of actions. For example, in home automation environment, a 'user coming home event' can invoke 'turn on the lights' action, 'turn on TV and change to the favorite channel' action and 'turn on the air conditioner' action. These series of actions can be managed by a centralized server, but the web intents make it possible in distributed ways, as following:

- 1) A door sensor detects that a user comes back home.
- 2) The door sensor node creates web intent of 'come_home', store the timestamp, security information and others on the intent.
- 3) The door sensor node finds intent receiver in its own intent manager. If no intent receiver found, queries it to the home intent server.
- 4) Home intent server sends a list of intents that includes TV action of 'turn on TV', light actuator action of 'turn on lights', and air conditioner action of 'turn on'.
- 5) A door sensor node saves list of intents, and communicate to each node on the list to invoke actions.

B. Multiple sensor events, an integrated action

Also, there are some delicate actions that require more information from various sensors. For these actions, the service server may be need to communicate to sensor nodes chronically to gather required data. For example of fire alarm and escape path notification service, the service server should know where are the people to evacuate, which ways are safe to pass, where are crowded or not. With web intents, the service server can make sensor nodes to report its data for its own goal.

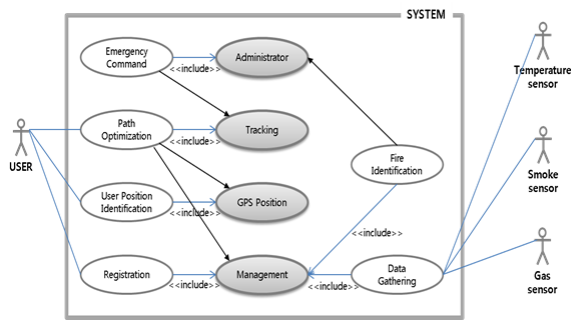


Fig. 5. Use case of Fire Alarm Scenario

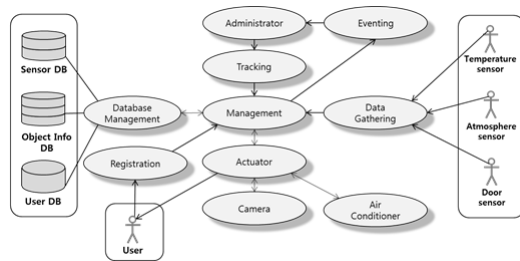


Fig. 6. Use case of Pet Care Service Scenario

- 1) A fire sensor detects a fire on sector A.
- 2) The fire sensor creates web intent of 'fire' and store the timestamp, fireplace information on the intent.
- 3) An escape path notification service server takes that intent, and plan to calculate escape path.
- 4) The server creates web intent of 'is_people_in_danger' and queries it to the intent server.
- 5) A motion sensor nodes takes the intent and reports that some people in danger on sector A.
- 6) A motion sensor nodes takes the intent and reports that people are crowded in path B.
- 7) With gathered data, the server calculate the path and notify to people in danger.

V. IMPLEMENTATION AND EVALUATIONS

A. Implementation

We implemented web intents module for device on AVR Raven with Contiki [7] with C. Web intents server is implemented by modifying web intent projects on Github [8]. The target scenario is fire alarm service and pet care service, which is described Figure 5 and 6.

B. Evaluation Factors

Since the web intents module will be deployed on limited nodes, it is important to optimize for the device. We suggest a list of evaluation factors for web intent module implementation on Table I. The comparison with conventional system is a future work.

TABLE I
PERFORMANCE EVALUATIONS FACTORS

Factors	Units
Processing power for intent creation	clocks
Processing power for intent parsing	clocks
Intent data size	Bytes
Network Bandwidth	Mb/s
User Buffer	KB
Average delay from invoke to action	ms

VI. CONCLUSION

With advance of technologies, many objects around us will be connected on the network. Machine-to-machine communication technology gets popular, and world-wide web becomes the platform of platforms. To enhance our life and experiences by providing rich web service with ubiquitous devices, whole things need to communicate via web - the web of objects. To realize these web of objects, the network environments should be light-weight for limited and constrained devices and network.

Web intents is a framework for web-based inter-application communication and service discovery, which includes light-weight RPC systems. It's philosophy is suitable to web of objects, therefore we extends it for limited devices. It is based on the RESTful APIs to react the dynamic changes of service configuration. The proposed web intents for devices is modified version to establish connection to intent receivers because it is hard to decide intent action for the device own.

We implemented the web intents module for AVR Raven toolkit. The nodes that deployed web intents module are applied to pet care service scenario and fire alarm scenario.

The remained work is optimizing the implementation for more limited devices of class-1. Compatibility with CoAP protocol is also another issue of remained work. Comparing with legacy devices need to be done. We expect that this study will be helpful to studies for a kind of machine-to-machine communication research and web of things.

ACKNOWLEDGMENT

This research was supported by Knowledge Economy Technology Innovation Programs - International Collaborative R&D Programs, and part of the EU ITEA-2 project 10028 Web-of-Objects (WoO) funded by MKE and supervised by KIAT.

REFERENCES

- [1] C. Bormann, *Guidance for Light-Weight Implementations of the Internet Protocol Suite*, 2nd ed, IETF LWIG Working Group, Aug. 2012
- [2] Z. Shelby, K. Hartke, and B. Frank, *Constrained Application Protocol (CoAP) draft-ietf-core-coap-11*, IETF CoRE Working Group, July 2012
- [3] R. Gravelle, *Comet Programming: Using Ajax to Simlate Server Push*, <http://www.webreference.com/programming/javascript/rg28/index.html>, Oct. 2010.
- [4] I. Fette and A. Melnikov, *The WebSocket protocol draft-ietf-hybi-thewebsocketprotocol-17*, 17th ed. IETF HyBi Working Group, Sep. 2011

- [5] H. Alvest, *Overview: Real Time Protocols for Brower-based Applications draft-ietf-rtcweb-overview-04*, 4th ed. IETF Network Working Group, June 2012
- [6] *Web Intents official web site*, <http://webintents.org/>
- [7] *AVR RAVEN Toolkit*, <http://www.atmel.com/tools/AVRRAVEN.aspx>
- [8] P. Kinlan, *Web intents implementation GitHub Project*, <https://github.com/PaulKinlan/WebIntents>
- [9] A. Alheraish, *Design and implementation of home automation system*, Consumer Electronics, IEEE Tran. Vol.50 Issue 4, pp 1087-1092, Nov 2004
- [10] L. Liu, M. Gaedke, and A. Koepfel, *M2M interface: a Web services-based framework for federated enterprise management*, Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on, July 2005