Hou CD, Li D, Qiu JF *et al.* SeaHttp: A resource-oriented protocol to extend REST style for Web of Things. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 29(2): 205–215 Mar. 2014. DOI 10.1007/s11390-014-1423-z

SeaHttp: A Resource-Oriented Protocol to Extend REST Style for Web of Things

Chen-Da Hou[1,2] (侯陈达), Dong Li[1,*] (李　栋), Jie-Fan Qiu[1,2] (邱杰凡), Hai-Long Shi[1,2] (石海龙) and Li Cui[1] (崔　莉)

[1] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

[2] *University of Chinese Academy of Sciences, Beijing 100049, China*

E-mail: {houchenda, lidong, qiujiefan, shihailong, lcui}@ict.ac.cn

Received November 18, 2013; revised January 20, 2014.

Abstract Web of Things (WoT) makes it possible to connect tremendous embedded devices to web in Representational State Transfer (REST) style. Some lightweight RESTful protocols have been proposed for the WoT to replace the HTTP protocol running on embedded devices. However, they keep the principal characteristic of the REST style. In particular, they support one-to-one requests in the client-server mode by four standard RESTful methods (GET, PUT, POST, and DELETE). This characteristic is however inconsistent with the practical networks of embedded devices, which typically perform a group operation. In order to meet the requirement of group communication in the WoT, we propose a resource-oriented protocol called SeaHttp to extend the REST style by introducing two new methods, namely BRANCH and COMBINE respectively. SeaHttp supports parallel processing of group requests by means of splitting and merging them. In addition SeaHttp adds spatiotemporal attributes to the standard URI for naming a dynamic request group of physical resource. Experimental results show that SeaHttp can reduce average energy consumption of group communication in the WoT by 18.5%, compared with the Constrained Application Protocol (CoAP).

Keywords cloud-sea computing, Web of Things, resource-oriented, group communication, spatiotemporal attribute

1 Introduction

Internet of Things (IoT) consists of an enormously growing number of smart devices. It is estimated that over 50 billion of smart devices will be connected to the Internet by 2020[1]. The main interest in connecting these devices is to allow them to interact with each other using the existing web technology. It is expected that every device in the sensor and actuator network has a URI and can be queried for its readings. Besides that, the devices are abstracted as resources of the WoT, which can be reused in different applications. This new approach leads the IoT to a new developing period, named Web of Things (WoT)[2].

At present Representational State Transfer (REST)[3] architectural style is widely used for constructing web services in the Internet. An early work considering the application of REST principles in WoT system is called Constrained Application Protocol (CoAP)[4]. It seeks to apply the REST architectural styles and basic features of HTTP to constrained networks. At the same time, Embedded Binary HTTP (EBHTTP)① was proposed for developing embedded applications in REST architectural styles. These two RESTful protocols inherit the principal characteristic of HTTP, which is supporting one-to-one requests in client-server model.

However, in practical physically embedded devices typically operate in groups. For example, in an automated building all lights in a given room need to be turned on/off as a group. The group communication feature is extremely helpful to execute one request on a subset of multiple devices. Fig.1 illustrates the different communication costs for two protocols, one of which supports group communication while the other does not. We can see that group communication can lead to

Regular Paper

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06010403, the International Science and Technology Cooperation Program of China under Grant No. 2013DFA10690, the National Natural Science Foundation of China under Grant No. 61003293, and the Beijing Natural Science Foundation under Grant No. 4112054.

*Corresponding Author

① http://tools.ietf.org/search/draft-tolle-core-ebhttp-00, December 2013.

fewer hops. So we need a protocol which not only supports one-to-one communication but also one-to-many communication for practical use of the WoT. However none of the current RESTful protocols supports group communication efficiently.

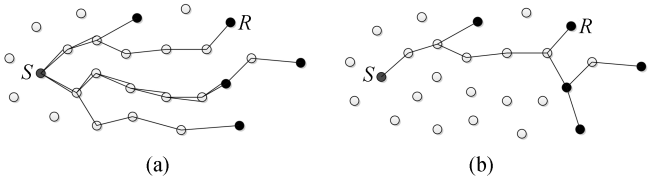


Fig.1. When the sending node produces separate requests to all destination nodes marked with solid circles (a), it causes more hops than utilizing an efficient group communication protocol (b).

Another problem in the WoT is how to identify physical resources, which are different from virtual resources. In the WoT, there are a large number of physical resources. Generally, physical resources have more constraints than virtual resources. They are located in a position, which may be changed with time, and they cannot be replicated. Consequently physical resources are usually defined by spatiotemporal attributes.

Based on the analysis of the abovementioned requirements and the problems, this paper proposes a protocol to extend the REST style by adding group communication for the WoT. The proposed protocol called SeaHttp focuses on supporting optimized communication with a group of resources across multiple embedded devices.

The contributions of this paper are as follows.

Firstly as an extended RESTful protocol, the proposed SeaHttp adds two original methods: BRANCH and COMBINE for group communication. These two new methods support group communication in the WoT while inheriting the merits of the REST principles. The function of the two new methods is to implement parallel processing of group requests in the network via splitting and merging. As a result, SeaHttp reduces the transmission of redundant data and saves energy.

Secondly, to name a group of physical resources in the WoT, SeaHttp adopts an improved URI mechanism by adding spatiotemporal attributes to the standard URI for naming a dynamic request group of physical resources. Its key feature is combining spatiotemporal attributes with the well-known URI to produce a dynamic address to name a group.

Finally we conduct a series of experiments to demonstrate the feasibility and performance of this protocol.

The rest of the paper is organized as follows. Section 2 introduces fundamentals and related work about group communication protocol in the WoT. Section 3 describes the main idea of the SeaHttp protocol. Design detail of the protocol is introduced in Section 4. Experimental scenarios and evaluation results are presented in Section 5. Section 6 concludes the paper.

2 Fundamentals and Related Work

In this section, we first summarize the important fundamentals of the WoT. The WoT stack is as Fig.2 shows. Second, we summarize related work on realizing group communication protocol in resource-constrained environments for the WoT.

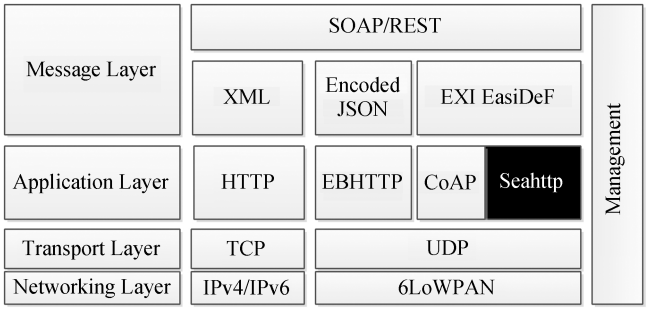


Fig.2. Web of Things stack (the element in the black is the contribution of this paper).

Initial researches have been carried out in order to offer IPv6 connectivity to smart objects based on IEEE 802.15.4. These researches are presented mainly under the 6LoWPAN[5][-6], GLoWBAL IPv6[7], and 6man② work.

Once, it has been enabled for smart objects to conduct end-to-end communication through Internet. It was considered necessary to define a horizontal access to the application layer in embedded devices. Analyzing the current developing status of the Internet, we see that web is the most widely used services medium. For that reason, some researchers have proposed to extend the web technology to the smart objects, which is referred to as the Web of Things[2]. Then, it is natural to extend HTTP to embedded devices, so that a binary version of HTTP called Embedded Binary HTTP (EBHTTP) was proposed for constructing RESTful web services of things. An application of EBHTTP for building RESTful web services has been considered in [8]. EBHTTP is a compressed version of the standard HTTP protocol, which is featured by binary-formatted and stateless encoding. It is used for resource constrained WSNs. The design of this protocol focuses on reducing the overhead of HTTP while maintaining the same HTTP semantics and communication paradigm. Though EBHTTP is a lightweight RESTful application-layer protocol, it does not take group

②Transmission of IPv6 over MS/TP networks, http://tools.ietf.org/html/draft-ietf-6man-6lobac-01, December 2013.

communication in the WoT into consideration. Even if we use EBHTTP to implement group communication it will induce extra energy consumption in resource-constrained devices due to redundant transmission.

Another related work on resource-oriented protocol is the Constrained Application Protocol (CoAP), which was proposed for building an efficient RESTful web application in resource constrained environments. CoAP provides a compact transfer capability on top of User Datagram Protocol (UDP), and realizes exactly a subset of HTTP methods, namely GET, PUT, POST and DELETE, which is necessary to offer RESTful web services in wireless sensor networks. The presenters of CoAP found the requirement of group communication in the WoT and drew up a draft③ to extend CoAP to support group communication in embedded devices on top of IP multicast. However, the group communication is different from the IP multicast. Firstly multicast deals with the same request for multiple devices while group communication handles both the same and the distinct request. Secondly group communication can support some computing operations such as sum, maximum, minimum and average of group while multicast cannot do that. Thus CoAP cannot solve the problem of group communication completely.

Based on CoAP Isam Ishaq *et al.* proposed an entity manager[9] to implement group communication in constrained environments. This work puts a group of resources together, which is called an entity. The entity manager is located in a smart gateway. The entity manager breaks down the request of entity and sends the individual requests to the respective objects using several unicast CoAP messages. Thus the work still has reluctant transmission and energy inefficiency in resource-constrained networks.

On the other hand, as multicast is very similar to group communication, there are some previous studies which propose several multicast protocols in wireless sensor networks. In [10] a protocol named BAM (Branch Aggregation Multicast) is presented. It supports single-hop link-layer multicast and multi-hop multicast in a way of doing branch aggregation. VLM2 (Very Lightweight Mobile Multicast)[11] is another multicast protocol for sensor nodes, which supports node mobility. VLM2 provides multicast from a base station to sensor nodes and unicast from sensor nodes to a base station. In [12] the authors propose an effective all-in-one solution for unicasting, anycasting, and multicasting in wireless sensor and mesh networks. RBMulticast in [13] is a stateless, receiving node based multicast protocol, which uses the geographic locations of nodes to reduce the cost of state maintenance. Furthermore, there are several multicast solutions for WSNs based on the geographical position of the sensor nodes[14-16]. Next the authors of [17] analyzed IP Multicast and showed that it is possible to use it in WSNs. An overlay multicast protocol[18] for wireless sensor networks presented by Gerald Wagenknecht, Markus Anwander, and Torsten Braun in 2012 is to design a protocol that supports multicast in WSNs in an efficient and energy-saving way. They used UDP as the transport protocol. It does not support any reliability since it is stateless, but it benefits from low complexity. All of these protocols do not support the Web of Things or the REST architectural style. Each of these protocols is only used in specific network and is not directly connected to the Web. Thus all of them are lack of interoperation, which is the advantage of the WoT.

3 Overview of SeaHttp

As mentioned above, a key challenge is to realize group communication for extending the REST style to embedded devices in the WoT. Thus this paper proposes a novel resource-oriented protocol, SeaHttp, for group communication using BRANCH and COMBINE methods in the WoT. In this section we introduce our assumptions and main idea of SeaHttp.

In this paper we make the following assumptions in SeaHttp design: 1) embedded devices maintain their own attribute information, including location, time on demand; 2) there is a unicast IP route protocol for end-to-end communication in the networking layer.

Resource-constrained networks of the WoT are different from the Internet and Web and RESTful principles are hardly applied to the WoT directly. Thus EBHTTP was proposed as a binary version of HTTP and CoAP was presented which provides protocol header compression, asynchronous transfer mode, and resource discovery mechanism for the WoT. EBHTTP and CoAP are RESTful protocols with four basic methods of GET, PUT, POST, and DELETE, which can cover the majority requirements of Web users on Internet. The four basic methods follow the principle of one-to-one request. However, resource-constrained devices in the WoT typically operate in groups. For example, in an environment monitoring system, observers usually request average temperature from a group of thermometers or need to control lights in a given room simultanously. The traditional RESTful protocols, EBHTTP and CoAP, cannot support such group communication directly. Even though Web users can write a program to implement group communication using CoAP API, there could be redundant transmission in resource-constrained devices. Thus we pro-

③Group communication for CoAP, http://tools.ietf.org/html/draft-ietf-core-groupcomm-18, December 2013.

pose two methods of COMBINE and BRANCH, which implement parallel group requests sent to multiple devices with a reduction of energy consumption.

The two new methods, BRANCH and COMBINE, are designed to merge and split requests in resource-constrained networks. To design the two methods, the first significant problem is to define node type in SeaHttp. There are four kinds of nodes, including source node, forwarding node, branching node, and receiving node. Branching nodes take part in the group communication to merge and split requests, forward requests, and maintain state information about receiving nodes and other branching nodes. Forwarding nodes have no information about the group communication state and just forward the data from one neighbor to the next one. An example of the network topology in SeaHttp is shown in Fig.3. Thus the branching nodes build up an application-layer overlay network upon the transport layer.

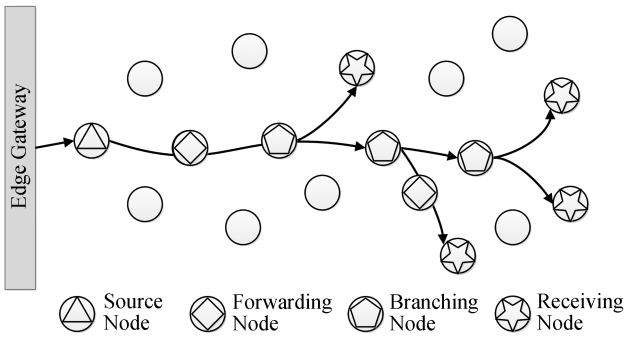


Fig.3. Roles of the nodes in a group communication scenario.

Another important problem which should be taken into consideration is how to name a group of resources across multiple devices dynamically. For naming a group of physical resources we propose an improved URI mechanism with a combination between the standard URI and the attributes information of physical resources. The attributes include space and time attribute labels, to define various requirements of Web users. Thus in our work, URI identifies not just virtual resources, but also real-world objects like sensors and actuators. With such improved URI SeaHttp achieves interoperability among the independent WoT systems.

4 Protocol Design

4.1 Packet Format

Fig.4 offers the SeaHttp packet format. The first 2-byte version ID (ver) is for packet switching in the protocol stack. ToS (type of service) indicates four kinds of packets in SeaHttp, which are “data”, “join”, “leave”, and “update” packets. The update packets are used in group management and periodic group list update. TTL (time to live) provides a maximum time, in hop number, that a packet should last in the network. “Method” indicates GET, PUT, POST, DELET, BRANCH or COMBINE. “Operator” can be SUM, AVEG (average), MAX, MIN, COMP (compare). “Message ID” is similar to message ID in CoAP. It is 16-bit unsigned integer in network byte order, which is used to match messages of type Acknowledgement/Reset, and for the detection of message duplication. The attribute label will be introduced in Subsection 4.2. The destination list maintains the destination nodes’ addresses. The maximum number of nodes included in the destination list allowed in SeaHttp is defined by the packet size. For example the maximum packet size is 128 bytes in the 802.15.4 standard. Thus the maximum number of nodes in the destination list is less than 50 when the data payload is not taken into consideration. Moreover SeaHttp uses sensor ID instead of IP address because the node IP of a group has the same prefix such as “fec0::”. As a result it could increase the maximum number of nodes in a group and decrease the overhead introduced by the destination list.

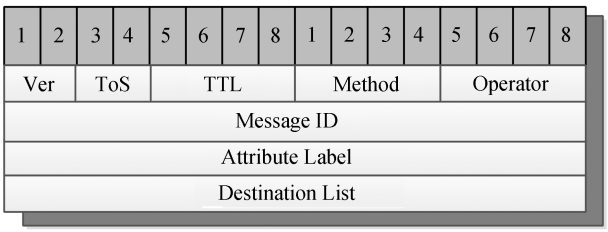


Fig.4. Packet header of the SeaHttp protocol.

4.2 Improved URI

This subsection introduces the improved URI design. The improved URI consists of a compact sequence of characters to identify physical resources. The structure of the improved URI has little difference with the standard URI. The improved URI is also organized hierarchically, with components listed in the order of decreasing significance from left to right. The detailed definition of the improved URI is shown in Fig.5.

```
SeaHttp-URI="SeaHttp:" "//" host [":"port]pathabempty
["?" query] ["⟨"location"⟩" "⟨"time"⟩"]
```

Fig.5. Definition of the improved URI.

The SeaHttp URI is improved from CoAP URI. It uses the ”seahttp” URI schemes for identifying and locating the physical resources. The next part is the host name and its port number. “path-abempty” refers to the path of resources. Then the key part is the spa-

tiotemporal attribute label, which is different from the CoAP URI. It identifies physical resources by their attribute information. Two examples of the improved URI are listed in Fig.6.

SeaHttp://www/homeautomation.com:5678/temperature/⟨RoomA⟩ ⟨Current⟩/
SeaHttp://www/homeautomation.com:5678/light?state=off/⟨RoomB⟩ ⟨22:00PM⟩/

Fig.6. Examples of improved URI.

The next step is to parse the spatiotemporal attribute label in the smart gateway. The gateway maintains the attribute information of all nodes of the network with a resource table. The resource table is created by the resource discovery which will be introduced in the next subsection. Thus the spatiotemporal attribute labels are transferred to a list of destination nodes.

In summary, the improved URI provides several benefits. It makes use of the feature of physical resources. Thus users do not need to specify one or several nodes to get information. Moreover the improved URI can reduce the coding complexity for system developers. They can use an improved URI instead of writing a long program to implement requirements. Finally the improved URI can produce group requests dynamically.

4.3 BRANCH and COMBINE

In order to illustrate the two new proposed methods, we implement resource discovery with BRANCH and COMBINE separately. Then SeaHttp supports resource discovery to manage resource information of all nodes in the gateway.

Firstly we give an example of BRANCH in the process of resource discovery. The discovery procedure is shown in Fig.7. Initially the sending node S creates a join message which includes all nodes in the network. Next, the sending node broadcasts the join message to its next hop neighbor. If all receiving nodes can be reached through one next hop neighbor, the next hop becomes a forwarding node F. Otherwise if the receiving nodes are reached by different next hop neighbors, it becomes a branching node B. The receiving node list is split accordingly and transmitted to the next respective hop neighbors. Then the branching node adds its address into the ID field of the join message. When a join message reaches a receiving node R it confirms the join by transmitting a join ACK message back to the last branching node, written in the branch ID of the join message and resource information such as location and resource type. The branching node takes following operations in sequence. First, it waits for the join ACK messages of all subordinate receiving nodes ($R1$ and $R2$). Then it combines these messages into one, and puts its own address as a branching node into the message and transmits it back to the sending node (or to the next branching node upstream on the path towards the sending node). The sending node knows all branching nodes and can later establish an overlay connection to them. Generally the sending node is gateway. Thus the gateway has resource information of all nodes in the network.

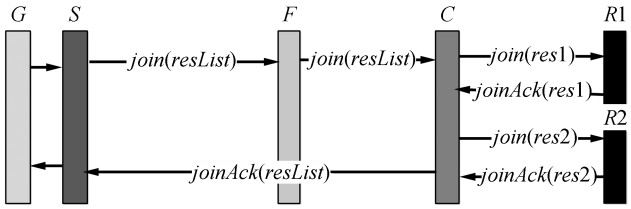


Fig.7. Resource discovery with BRANCH.

The discovery procedure with the COMBINE method is shown in Fig.8. Nodes can report their resource information actively. Each receiving node transmits a join message, which includes resource information, to the next upstream neighbor into the direction towards the sending node according to the routing table. The neighbor node collects all incoming join messages and becomes a branching node or a forwarding node. If the node is a branching node, it adds its own identity to the appropriate field in the join message to inform the sending node about its role as a branching node. Afterwards, the node transmits the message further to the sending node. The sending node collects all join messages and creates a join ACK message, which includes the receiving node list, the collected branch ID, and its own ID. The join ACK message is transmitted towards the receiving nodes. The branching node splits the join ACK message and transmits the messages to the according receiving nodes.

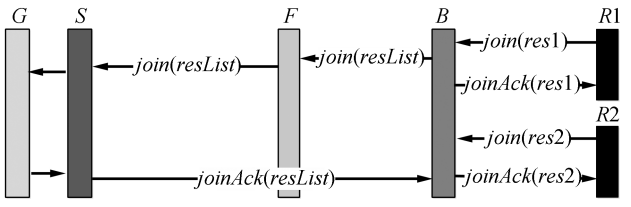


Fig.8. Resource discovery with COMBINE.

5 Performance Evaluation

We evaluate the performance of SeaHttp through a real implementation using TelosB④ motes. TelosB is

④Crossbow Technology Inc., TelosB datasheet. http://www.willow.co.uk/TelosB_Datasheet.pdf, December 2013.

a typical example of a low-cost wireless sensor used in constrained sensor networks. It features 16-bit RISC MCU at 8 MHz and 16 registers. The platform offers 10 KB of RAM, 48 KB of flash memory and 16 KB of EEPROM. It provides a CC2420 radio chip for wireless communication. One TelosB mote implements the 6LoWPAN border router connected to an embedded gateway running Linux. The gateway is an ARM11 board named Tiny6410⑤ using Samsung S3C6410A and 256M RAM.

To compare the performance of SeaHttp and CoAP, we use the implementation of CoAP in TinyOS[19], which is called CoapBlip⑥. Both SeaHttp and CoAP are built on top of the 6LoWPAN. We use Blip⑦ as an implementation of 6LoWPAN in TinyOS.

The paper evaluates the performance of SeaHttp from six aspects, including memory occupation, transmitted data size, latency, number of transmitted hops, energy consumption and user development code size.

5.1 Memory Occupation

Table 1 shows the amount of RAM and ROM memory allocated at compile time for SeaHttp and CoAP implementation. The occupation of ROM memory shows the complexity and weight of the code of each implementation. A code with a small memory footprint would allow adding further functionality or enrich protocol with more capabilities.

Table 1. Memory Occupation

Protocol	RAM (Byte)	ROM (Byte)
CoAP	7 112	43 454
SeaHttp	8 024	36 274

The experimental results show that CoapBlip has higher ROM memory footprint than the implementation of SeaHttp. The main reason is whether the code is optimized. CoapBlip is a C library, which is installed in the node along with the TinyOS component. The use of C libraries is normally too heavy for the memory in resource-constrained devices. SeaHttp reduces the ROM footprint to avoid the use of C libraries, and consists only of a code written in the TinyOS nesC language and optimized for this environment.

5.2 Transmitted Data Size

This subsection focuses on theoretical and experimental analysis of total transmitted data size in the network. Radio communication is a dominant factor determining the sensor energy consumption and the consumption is proportional to the amount of transmission. Generally we have to consider the energy for packet reception because the energy consumption to receive packet is nearly the same as that of to send. However, the number of receiving node changes according to the network topology. Therefore we just compare the effect only by the amount of packets to send.

First of all we give a theoretical analysis of transmitted data size. In this analysis, we do not consider the packet loss caused by link deterioration. The paper takes two boundaries of merging. At one extreme, network links to receiving nodes all merge at a single node (hereafter called the Max Merge case) through a single multi-hop link. To simplify the problem there is one sending node, $n-1$ forward nodes, one branching node and m receiving nodes as shown in Fig.9. At the other extreme, they do not merge in any node (hereafter called the Min Merge case) through different multi-hop links as Fig.10 shows. There are one sending node, $n \times m$ forwarding nodes and m receiving nodes in Fig.10.

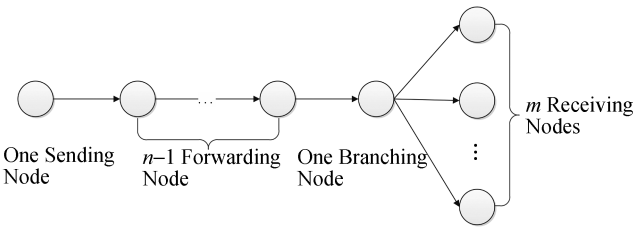


Fig.9. Topology of the Max Merge case.

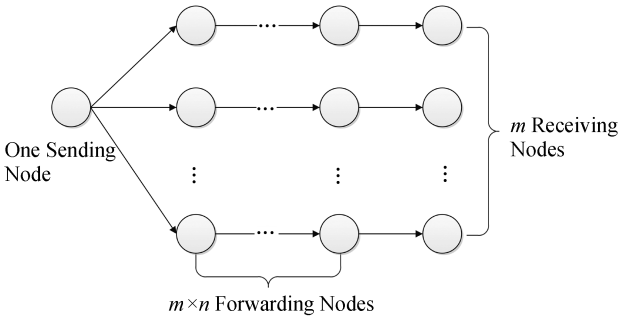


Fig.10. Topology of the Min Merge case.

Additionally the paper calculates the packet size of CoAP and SeaHttp. They take RPL as the underlying routing protocol. Hence, the SeaHttp packet size (*SeaHttpPkt*)is calculated as follows:

$$SeaHttpPkt = SeaHttpHeader + UdpHeader + SicsHeader + Payload, \quad (1)$$

⑤http://www.friendlyarm.net/products/tiny6410, December 2013.

⑥http://tinyos.stanford.edu/tinyos-wiki/index.php/CoAP, December 2013.

⑦http://tinyos.stanford.edu/tinyos-wiki/index.php/BLIP_2.0_Tutorial, December 2013.

where *SeaHttpHeadert* means the header length of SeaHttp packet, *UdpHeader* means the header length of UDP packet in TinyOS, *SicsHeader* means the header length of the 6LoWPAN packet in TinyOS, and *Payload* means the length of the data included in both the SeaHTTP and the CoAP packet.

Then CoAP packet size (*CoAPPkt*) is calculated as follows:

$$CoAPPkt = CoAPHeader + UdpHeader + SicsHeader + Payload, \quad (2)$$

where *CoAPHeader* means the header length of the CoAP packet.

Next the paper assumes that the ACK packet size is the same as the sending packet size of SeaHttp and CoAP. Thus the acknowledge packet size (*AckPkt*) is calculated as follows:

$$AckPkt = CoAPPkt \text{ or } AckPkt = SeaHttpPkt. \quad (3)$$

For the Max Merge case SeaHttp aggregates the multiple paths indicated in Fig.9 into one. Redundant radio communication occurs between the branching nodes and the receiving nodes. The source and n intermediate forwarding nodes send the SeaHttp packet and the receiving nodes return the acknowledge packet. The sending nodes send the normal packets to m receiving nodes and the m sinks return acknowledge packets. The total amount of transmission packets (*Total*) for SeaHttp is given by

$$Total = (n + m) \times (SeaHttpPkt + AckPkt) + SeaHttpPkt + AckPkt. \quad (4)$$

In contrast CoAP should send all requests one by one to m receiving nodes. Thus the total amount of transmission packets for CoAP is given by

$$Total = m \times (n + 1) \times (CoAPPkt + AckPkt) + m \times (CoAPPkt + AckPkt). \quad (5)$$

For the Min Merge case SeaHttp and CoAP almost work in the same way. Because there is no any branching node in the network, they do not aggregate the multiple paths. The only difference is that the sending node transmits m requests to m receiving nodes in CoAP while the sending node transmits only one request to m receiving nodes in SeaHttp. Thus the total amount of transmission packets for SeaHttp is given by

$$Total = m \times (n + 1) \times (SeaHttpPkt + AckPkt) + SeaHttpPkt + AckPkt. \quad (6)$$

The total amount of transmission packets for CoAP is given by

$$Total = m \times (n + 1) \times (CoAPPkt + AckPkt) + m \times (CoAPPkt + AckPkt). \quad (7)$$

We set (*SeaHttpPkt*, *CoAPPkt*, *AckPkt*) as (100, 70, 70) bytes referring to a real implementation on TelosB. The number of hops of disjoint paths, n, changes from 1 to 10. The number of receiving nodes, m, changes from 2 to 10. Fig.11 and Fig.12 show the results of the qualitative evaluation. Through the figures, we make two pairs of comparison. One is between SeaHttp model and CoAP model, and the other is between SeaHttp and CoAP. The former pair means the analytical results based on (6) and (7), while the latter pair means the experimental results of these two protocols. The gains of using SeaHttp are illustrated in Fig.11 for the Max Merge case, leading to a maximum reduction of 78% and a minimum reduction of 9% in total transmitted

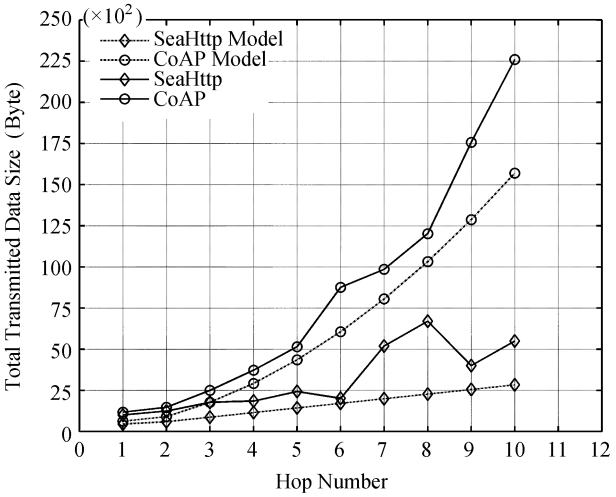


Fig.11. Comparison of total transmitted data size in the Max Merge case between CoAP and SeaHttp.

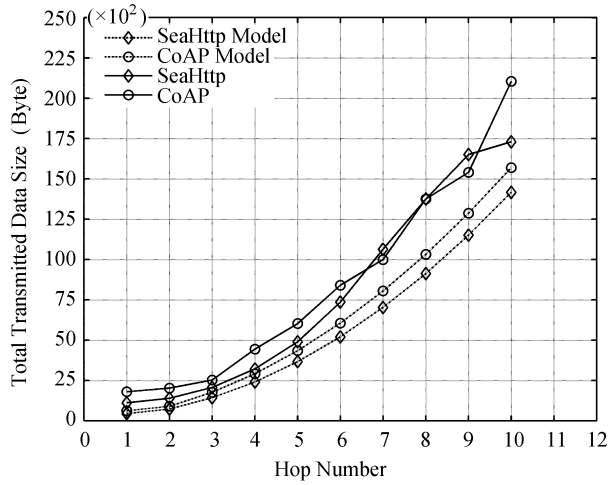


Fig.12. Comparison of total transmitted data size in the Min Merge case between CoAP and SeaHttp.

data size forwarded by intermediate nodes when compared with CoAP. The reduction is growing as the number of network nodes and hops grow. The Max Merge case has one shared data link to all receiving nodes, so CoAP has a large number of redundant transmitted data when it implements group communication. CoAP should send multiple requests to multiple receiving nodes one by one while SeaHttp can send one parallel request to multiple receiving nodes by splitting the request in branching nodes. Thus SeaHttp reduces redundant transmission compared with CoAP as Fig.11 shows.

The total transmitted data size of SeaHttp does not reduce obviously when compared with the total transmitted data size of CoAP in Fig.12 for the Min Merge case. In some test cases the total transmitted data size of SeaHttp is larger than that of CoAP because of the random number of retransmission packets. The main reason is that there is no any branching node in the network and no any shared transmission link to multiple receiving nodes. Thus there is no redundant transmitted data in both CoAP and SeaHttp and they achieve similar performance.

Moreover Fig.11 and Fig.12 show that the total transmitted data size in implementation is larger than calculated by the model. Because we do not consider the packet loss caused by link deterioration when we do theoretical analysis, the total data size is increasing in retransmission for reliable transmission when we do experiments.

5.3 Latency

The latency experienced by a sending node on sending a request to get a response from a receiving node is one of the most significant parameters used to evaluate the goodness of the protocol design. Low latency values can enhance user experience and benefit those real-time applications. We define the latency as the time elapsed from the moment the sender sends a request until the moment it receives the response.

Fig.13 shows the latency under different hop numbers. Each point on the graph represents the average latency value of 100 successful request/response transactions. As expected, the result shows that SeaHttp has better performance than CoAP.

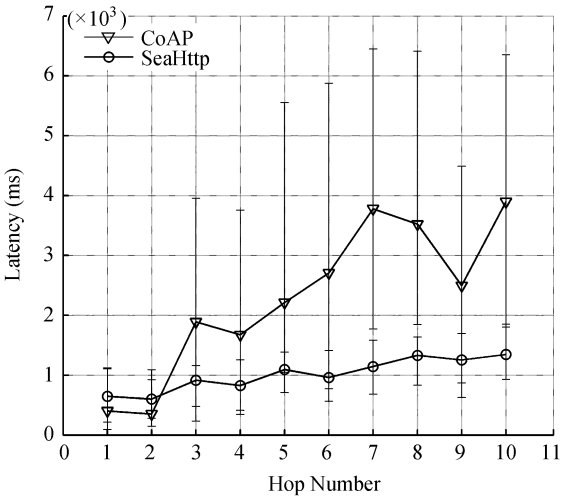


Fig.13. Latency.

5.4 Hop Count

In this subsection we design five experimental scenarios with different number of nodes and depths of tree, as shown in Fig.14. Each experimental scenario has two parameters. One is the number of nodes in the

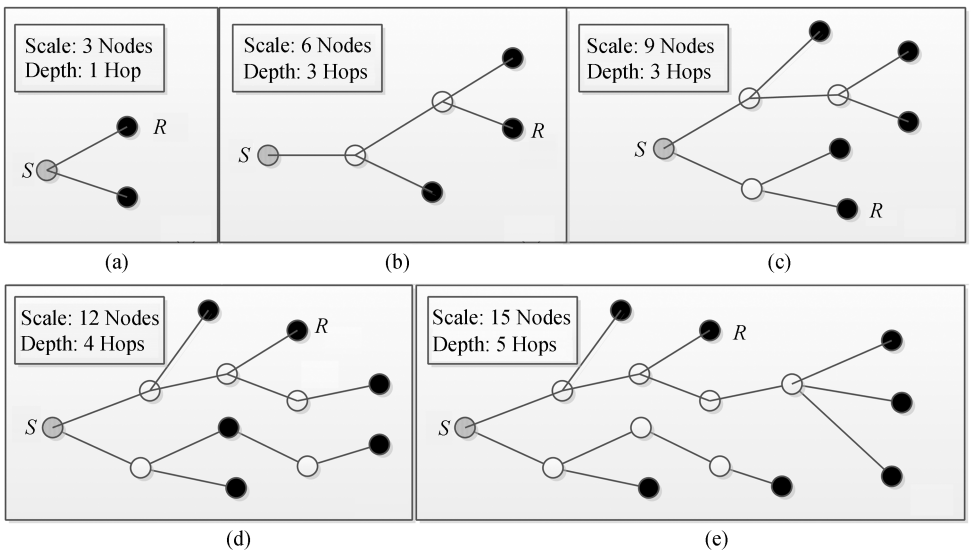


Fig.14. Topology of experimental scenarios.

whole network and the other is the tree depth in the network.

Since communication consumes more energy than calculation for a node, it is important to reduce the number of communication events between sensor nodes. Therefore, the protocol is better to route the packets directly and only to the nodes that really need to receive those packets. Such method can be achieved by using group communication. The gains of using group communication in the reduction of hop numbers are illustrated in Fig.15: an average reduction of 70.5% in the number of total hops forwarded by intermediate nodes when compared with CoAP.

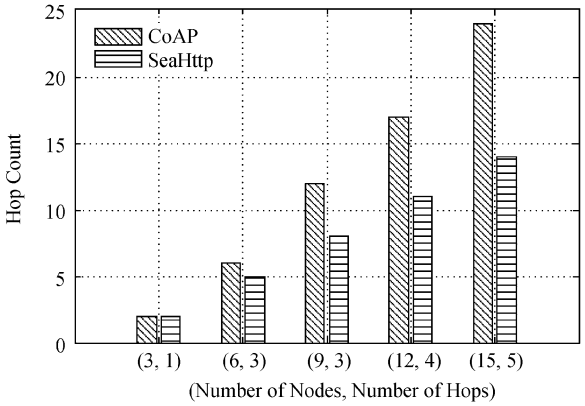


Fig.15. Total number of hops for transmission.

5.5 Energy Consumption

Fig.16 and Fig.17 show the energy consumption comparison between CoAP and SeaHttp. The paper evaluates the energy consumption of a node from four aspects, including CPU power (full power CPU time), LPM power (reduced power CPU time), LISTEN power (radio receive time), TRANSMIT power (radio transmit time). We also use five experimental scenarios mentioned in Subsection 5.5 as Fig.14 shows to test the energy consumption.

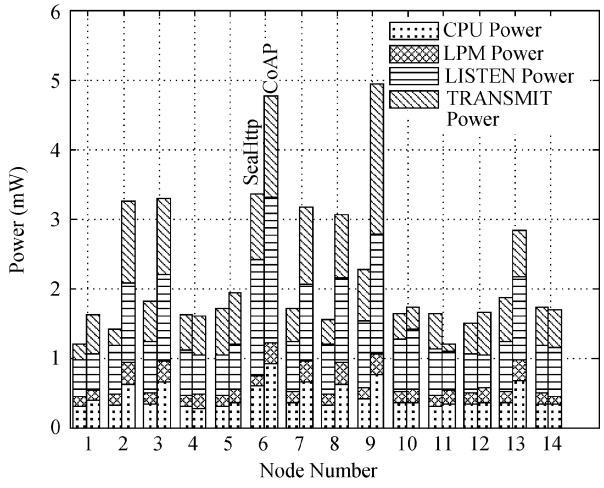


Fig.16. Energy consumption of each single node in the network.

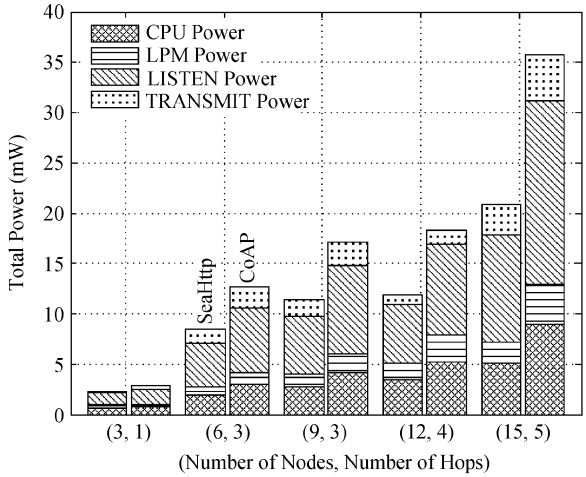


Fig.17. Total energy consumption of all nodes in the network.

Firstly Fig.16 shows the energy consumption results of each single node in the network. In Fig.16, No. 2, 3, 6, 7, 8, 9, 13 nodes are forwarding nodes and the other nodes are receiving nodes. Therefore it can be concluded that the difference of energy consumption is determined by the role of node (forwarding node or receiving node). The receiving nodes in SeaHttp and CoAP have little difference in the number of sending and receiving packets so their energy consumptions are very close to each other. However, the energy consumptions of the forwarding nodes are different with different numbers of forwarding packets. In addition CoAP should send multiple requests to multiple receiving nodes one by one while SeaHttp can send one parallel request to multiple receiving nodes by splitting the requests in branching nodes. SeaHttp reduces redundant forwarding data compared with CoAP. So the energy consumption of forwarding nodes in CoAP is 1.2∼2.1 times larger than that in SeaHttp in this experiment. Secondly Fig.17 shows the energy consumption results of all nodes in the network. The reduction of energy consumption is growing as the number of network nodes and hops grows. SeaHttp leads to an average reduction of 18.5% in the total energy consumption in the network compared with CoAP in this experiment. Because the energy consumption is proportional to the amount of transmission, the experimental results are in line with those of transmitted data experiments in Subsection 5.2.

5.6 User Development Code Size

Another advantage of SeaHttp is to reduce the user development code size for users' application. As Table

2 shows, the code size of SeaHttp is smaller than that of CoAP in all experimental scenarios. Even in one hop scenario, the code size of SeaHttp has a 30% reduction compared with that of CoAP. Thus this test proves the reduction of development cost using SeaHttp.

Table 2. User Development Code Size (Byte)

Protocol	(Number of Nodes, Number of Hops)				
	(3, 1)	(6, 3)	(9, 3)	(12, 4)	(15, 5)
CoAP	305	354	360	412	459
SeaHttp	197	210	215	228	256

6 Conclusions

Group communication among resource-constrained devices in the Web of Things is typical in practical applications. However, none of currently proposed protocols provide efficient support for group communication. This paper proposed a resource-oriented protocol called SeaHttp to solve the problem. Compared with the current RESTful protocols, SeaHttp is featured by two new methods, namely COMBINE and BRANCH, and the improved URI mechanism which adds spatiotemporal attributes to the standard URI. Through extensive experiments, we verified that SeaHttp can reduce the number of transmission hops and average transmission latency for each request. In addition experimental results show that SeaHttp outperforms CoAP in the case of group communication when multiple target leaf nodes are located in the same branch of the tree topology.

In future we will focus on the problem of low response rate induced by the low power mode of the resource-constrained nodes in the embedded networks of the WoT.

References

[1] Emerson B. M2M: The internet of 50 billion devices. *Win-Win Magazine* (*Huawei*), Jan. 2010, pp.19-22.

[2] Guinard D, Trifa V, Wilde E. A resource oriented architecture for the Web of Things. In *Proc. the 2010 Internet of Things*, Nov. 29-Dec. 1, 2010, pp.1-8.

[3] Fielding R T. Architectural styles and the design of network-based software architectures [Ph.D. Thesis]. Software Research Group, University of California, Irvine, 2000.

[4] Bormann C, Castellani A P, Shelby Z. CoAP: An application protocol for billions of tiny Internet nodes. *Internet Computing*, 2012, 16(2): 62-67.

[5] Montenegro G, Kushalnagar N, Hui J, Culler D. Transmission of IPv6 packets over IEEE 802.15.4 networks. *Internet Proposed Standard RFC*, 4944, 2007.

[6] Mulligan G. The 6LoWPAN architecture. In *Proc. the 4th Workshop on Embedded Networked Sensors*, June 2007, pp.78-82.

[7] Jara A J, Zamora M A, Gómez-Skarmeta A. Glowbal IP: An adaptive and transparent IPv6 integration in the Internet of Things. *Mobile Information Systems*, 2012, 8(3): 177-197.

[8] Dawson-Haggerty S, Jiang X F, Tolle G *et al.* sMAP: A simple measurement and actuation profile for physical information. In *Proc. the 8th ACM Conference on Embedded Networked Sensor Systems*, Nov. 2010, pp.197-210.

[9] Ishaq I, Hoebeke J, Van den Abeele F *et al.* Group communication in constrained environments using CoAP-based entities. In *Proc. the 2013 IEEE Int. Conf. Distributed Computing in Sensor Systems*, May 2013, pp.345-350.

[10] Okura A, Ihara T, Miura A. BAM: Branch aggregation multicast for wireless sensor networks. In *Proc. IEEE Int. Conf. Mobile Adhoc and Sensor Systems*, November 2005, pp.363-373.

[11] Sheth A, Shucker B, Han R. VLM②: A very lightweight mobile multicast system for wireless sensor networks. In *Proc. IEEE International Conference on Wireless Communications and Networking*, March 2003, pp.1936-1941.

[12] Flury R, Wattenhofer R. Routing, anycast, and multicast for mesh and sensor networks. In *Proc. the 26th IEEE Int. Conf. Computer Communications*, May 2007, pp.946-954.

[13] Feng C H, Heinzelman W B. RBMulticast: Receiver based multicast for wireless sensor networks. In *Proc. IEEE Int. Conf. Wireless Communications and Networking*, April 2009.

[14] Koutsonikolas D, Das S M, Hu Y C, Stojmenovic I. Hierarchical geographic multicast routing for wireless sensor networks. *Wireless Networks*, 2010, 16(2): 449-466.

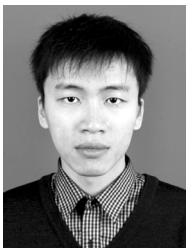
[15] Sanchez J A, Ruiz P M, Stojmenovic I. Energy-efficient geographic multicast routing for Sensor and Actuator Networks. *Computer Communications*, 2007, 30(13): 2519-2531.

[16] Lee J, Lee E, Park S, Oh S, Kim S H. Consecutive geographic multicasting protocol in large-scale wireless sensor networks. In *Proc. the 21st Int. Symp. Personal Indoor and Mobile Radio Communications*, September 2010, pp.2192-2197.

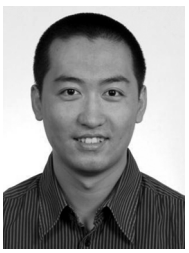
[17] Silva J S, Camilo T, Pinto P *et al.* Multicast and IP multicast support in wireless sensor networks. *Journal of Networks*, 2008, 3(3): 19-26.

[18] Wagenknecht G, Anwander M, Braun T. SNOMC: An overlay multicast protocol for wireless sensor networks. In *Proc. the 9th Annual Conference on Wireless On-Demand Network Systems and Services*, January 2012, pp.75-78.

[19] Levis P, Madden S, Polastre J *et al.* TinyOS: An operating system for sensor network. In *Ambient Intelligence*, Weber W, Rabaey J M, Aarts E (eds.), Springer Berlin Heidelberg, 2005, pp.115-148.



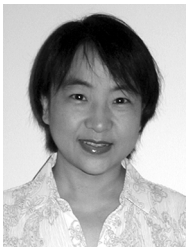
Chen-Da Hou is currently a Ph.D. candidate of ICT, CAS. He received the B.S. degree in computer science from Xidian University, Xi'an, in 2010. His main research interests include embedded system, wireless sensor network, Internet of Things, and Web of Things.



Dong Li is currently an associate professor of ICT, CAS. He received his Ph.D. degree in computer architecture from ICT, CAS, in 2009. His research focuses on wireless sensor networks, networked embedded system, and Internet of Things.

Jie-Fan Qiu is currently a Ph.D. candidate of Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing. He received his B.S. degree in electronic and information engineering from Henan Normal University, Xinxiang, in 2007, and his M.Sc degree in physical electronics from Zhejiang Normal University, Jinhua, in 2010. His main research interests include programming language, embedded operation system, sensor network, and Internet of Things.

Hai-Long Shi is currently a Ph.D. candidate of ICT, CAS. He received his B.S. degree in communication engineering and his M.Sc degree in communication and information system in 2008 and 2010 respectively, both from Wuhan University. His main research interests include embedded system, wireless sensor network, and Internet of Things.



Li Cui received her undergraduate degree from Tsinghua University, Beijing and the M.Sc. degree from the Institute of Semiconductors, CAS. She received the Ph.D. degree in bioelectronics from the University of Glasgow, UK, in 1999. She received "The 100 Talents Program of CAS" award in 2004 and is currently a professor and the leader of an interdisciplinary research group at ICT, CAS. Her current research is focused on sensor technology and wireless sensor networks.