Received July 28, 2017, accepted August 8, 2017, date of publication August 21, 2017, date of current version September 6, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2742020

A Lightweight XMPP Publish/Subscribe Scheme for Resource-Constrained IoT Devices

HENG WANG, (Member, IEEE), DAIJIN XIONG, PING WANG, AND YUQIANG LIU
Key Laboratory of Industrial Internet of Things and Networked Control, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Corresponding author: Heng Wang (withwh@gmail.com)

This work was supported in part by the National High Technology Research and Development Program of China under Grant 2015AA043801 and in part by the Fundamental and Advanced Research Program of Chongqing under Grant cstc2015jcyjA40023.

ABSTRACT The development of wireless technology like Internet Protocol version 6 over Low power Wireless Personal Area Networks, which defines IP communication for resource-constrained networks enables communication on the lower layers, while the diverging and incompatible application layer protocols lead to barriers in the way of information transferring between heterogeneous networks. Fortunately, extensible messaging and presence protocol (XMPP) is a preferential protocol to solve the problem of interoperability between heterogeneous networks. Hence, it is attractive to extend XMPP to the Internet of Things (IoT). However, XMPP protocol is initially designed for the Internet where the equipment is rich in resources, considering the characteristics of the IoT, the XMPP protocol needs to be optimized to meet the requirements of IoT, *e.g.*, downsized protocol, publish/subscribe service and sleeping mechanism. In this paper, by staying true to the IoT visions, we propose a lightweight XMPP publish/subscribe scheme for resource-constrained IoT devices to perform data exchange either periodically or upon any value change. According to the subscriber's needs, the publisher can adjust the data information published to the server. Inherit the merits of XEP-0060, the server maintains and manages the publish/subscribe relationships with multiple subscribers, as well as distributes the received data sent by publisher to all subscribers, yielding less complexity of publication and energy efficiency. In addition, the proposed scheme strongly supports publish/subscribe architecture of the sleeping clients that actually prolongs the lifetime of battery-powered devices. Thus, based on the standardized XMPP protocol, we present a down-sized and trimmed XMPP to implement this scheme. Through the experimental results, we demonstrate our work on optimizing and improving XMPP publish/subscribe scheme for resource-constrained IoT devices and show the simplicity and efficiency of this scheme.

INDEX TERMS Internet of Things, protocols, wireless sensor networks, XMPP, publish/subscribe mechanism.

I. INTRODUCTION

The Internet of Things (IoT) is bringing a massive surge of connected things. Billions of interconnected devices are integrated into the global Internet to facilitate an interaction with the physical world. With the rapid development of the IoT technologies, the IoT can be applied to a lot of areas, such as environmental sensing, intelligent shopping systems, energy management, transportation systems, medical care, smart home, industrial controlling and so on [1]–[4].

In order to bring IoT to the reality, organizations and alliances provide lots of solutions, which can be divided into two categories, *i.e.*, Non-Internet Protocol (IP) based and IP-Based. Rather than the Non-based solutions, the IP-based solutions may be the tomorrow of IoT networks. Indeed, Internet Engineering Task Force (IETF) set up several working groups [5], such as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), Routing over Lossy and Low-power Networks (RoLL) and Constrained Restful Environment (CoRE), to carry out the research of low power IPv6 network and standardize communication protocols for resource-constrained devices such as 6LoWPAN [6], Routing Protocol for Low power and Lossy Networks (RPL) [7], and Constrained Application Protocol (CoAP) [8], etc. However, due to the different technology selections and commercial profits of the solution-providers, the large-scale commercial applications of IoT have been hindered by the existing problems of non-unified standards system. Heterogeneous applications using different communication protocols and data models are congested in the IoT domain [9]. Separated application layer protocols like CoAP, Message Queuing Telemetry Transport (MQTT) and Web Application Messaging Protocol (WAMP) will bring issues of interoperability among heterogeneous networks to large-scale IoT deployments [8], [10], [11].

To address the above problems, we prefer to implement XMPP protocol in the IoT to achieve instant messaging, as well as to solve the problem of interoperability between heterogeneous networks. XMPP protocol is a real-time information transmission specification based on IP technology and Extensible Markup Language (XML) that supports publish/subscribe messaging systems. As an extensible application layer protocol for near real-time instant messaging [12], XMPP has been widely deployed and applied in the Internet. For example, chat, entertainment and other applications that require real-time communications. In particular, Google Talk[1] was an instant messaging service that used XMPP to provide real-time extensible messaging and presence events, including offline messaging and voice mailing [13]. Facebook also supported the XMPP protocol in its chat tool Facebook Chat in 2008 [14]. Moreover, XMPP has advantages in areas, *e.g.*, openness, flexibility, extensibility, interoperability, and so on [15]. Note that it is a promising protocol that is well suited for using within the area of IoT [16].

However, the XMPP protocol is initially designed for resource-rich equipment in the Internet that is unsuitable for the computing- and energy- constrained devices. In addition, the application of XMPP protocol in the Internet mainly focuses on data interaction, while the application of XMPP protocol in the IoT cares more about data collection. Moreover, the existing publish/subscribe scheme of XMPP Extension Protocols (XEPs) for the Internet applications mainly focus on the event-driven publication. While in the IoT, except for the publication that is triggered by the change of any value, by considering the need of periodic data transmission in industrial networks, the publish/subscribe scheme should be optimized to support efficient periodic publication, which will benefit the WSNs. Furthermore, there is no explicit sleeping mechanism in the existing XMPP publish/subscribe scheme, whereas the sensor devices in the IoT are mostly battery powered and set to sleep periodically or conditionally, so that improved XMPP publish/subscribe scheme should be provided to support sleeping nodes to save massive energy. A lot of work need to be done to bring XMPP to IoT, and more efforts should be made to implement efficient XMPP publish/subscribe scheme into the IoT. Therefore, it is necessary to optimize and improve the publish/subscribe scheme of the XMPP protocol.

In this paper, in considering the characteristics and application requirements of the IoT, we propose a lightweight XMPP publish/subscribe scheme for resource-constrained IoT devices based on the XMPP series protocol standards. The key point for publication and subscription is the server to which publishers send data and from which subscribers receive data publication. We add a publish/subscribe relationship maintenance and management functional module in the XMPP server to support the event-driven publication and periodic data publication. Our contributions and key results are summarized as follows.

1) We present a lightweight XMPP publish/subscribe scheme that achieves less complexity of publication and energy efficiency for resource-constrained IoT devices. On the one hand, this scheme inherits the merits of the XEP-0060 that the server maintains and manages the publish/subscribe relationships with multiple subscribers and broadcasts data to all authorized subscribers. On the other hand, in our proposed scheme, the subscription request of the subscriber needs to be processed by the publisher, which could adjust the data information published to the server according to subscriber's needs rather than publishing all its information, *e.g.*, a sensor node using object descriptions doesn't have to publish all its objects and attributes. In addition, a lightweight and small resource overhead XMPP is provided and a downsized XMPP client only with memory footprint about 43484 bytes FLASH and 8197 bytes RAM is implemented.

2) The proposed scheme supports publish/subscribe function of the sleeping sensor nodes, which are set to sleeping mode periodically or conditionally, so that the lifetime of IoT devices can be significantly extended by using our mechanism.

3) In addition to support for event-driven publication, the scheme also supports the periodic data publication by considering the need of industrial network for periodic data transmission.

The remainder of this paper is organized as follows. In Section II, the related works are provided and the overview of XMPP protocol is given. In Section III, the proposed XMPP publish/subscribe scheme is described. The experimental platform is presented in Section IV and performance evaluation results are shown in Section V. At last, concluding remarks and future work are given in Section VI.

II. RELATED WORK

A. GENERAL XMPP

XMPP is an open protocol based on XML for instant messaging and presence awareness and approved by IETF [17]. It was originally named Jabber [18], which was developed in 1999. The important XMPP RFC documents include XMPP Core, XMPP Instant Messaging and Presence, XMPP Address Format and End-to-End Signing and Object Encryption for the XMPP, corresponding documents are RFC 6120, RFC 6121, RFC 6122 and RFC 3923 [19]–[22]. As a widely used protocol in the field of instant messaging, XMPP offers the following key advantages:

[1]Google Talk for Windows was ceased to work and replaced by Google Hangouts in 2015.

Openness– extensive application of XMPP protocol in the Internet benefits from its openness. Many companies have launched dozens of open source servers and clients that support different application platforms in the XMPP official website. Thus, if the XMPP protocol is used as an application layer protocol for IoT, it can not only achieve a unified real-time application transmission specification within IoT, but also make the field network equipment and the Internet equipment follow a unified communication standard.

Extensibility– XSF (The XMPP Standards Foundation) developed a series of XMPP extension protocols based on the core specifications. The current hundreds of XMPP extensions cover multi-user chat, privacy lists, publish/subscribe, chat status notifications and etc. in instant messaging, which can meet the potential diverse needs of IoT when applying XMPP to IoT.

Flexibility– XMPP protocol is based on the XML, which is a self-description markup language that provides language- and platform- independent data forms. It allows different organizations define their own markup language according to industry needs. Inheriting the power of XML, XMPP is helpful to address the data exchange issues between heterogeneous networks.

B. XMPP FOR IoT

Researchers have been trying to bring IoT vision to reality for many years by using a wide range of technologies, like WSNs, which are the most promising IoT enabling technologies, since they sense and interact with the real world. Also a IP-based solution to enable sensor networks to be integrated into the internet is provided by the development of 6LoWPAN, an adaptation layer for low-power lossy networks designed by IETF. Nevertheless, when it comes to the IoT application layer standards, which are not uniform, many alternative protocols like CoAP, MQTT and XMPP are provided to manage resources in resource limited networks. CoAP is a synchronous request/response application layer protocol that was designed for the needs of constrained devices by the CoRE IETF group. However, CoAP itself is based on the request/response scheme, a scheme that intrinsically supports the topic-based publish/subscribe functionality may be more suitable for resource-constrained applications by considering the needs of IoT [23]. MQTT is a M2M connectivity protocol that was designed as an extremely lightweight publish/subscribe messaging transport by IBM. It is an ISO standard for use on top of the TCP/IP protocol [24]. Whereas we choose XMPP to make IoT a reality because of its some advantages over the above two protocols. On the one hand, in contrast to CoAP's request/response mode, XMPP inherently supports the publish/subscribe architecture that is more suitable for the IoT. On the other hand, with regard to the relatively new MQTT, XMPP is an already established protocol that is widely deployed all over the Internet [25]. Also with XMPP's good openness and scalability, it can be used to implement the interoperability between wide varieties of instant messaging systems [26]. Due to the advantages of XMPP, we prefer a thin but effective XMPP protocol to provide a solution as it is a communication protocol suitable for IoT.

XMPP is highly extensible and allows the XEPs. To facilitate the applications of XMPP to IoT, a lot of XMPP-IoT standardization effort has been made by XSF, which develops sufficient and valuable XEPs. Especially, an Internet of Things Special Interest Group (IoT SIG) within the XSF was proposed to be formed in 2016 [27]. For the IoT domain, Peter Waher proposed several valuable XEPs that define many different architectures as well as use cases, paving the way for the the application of XMPP in the IoT, for example, XEP-0347 defines the installation and configuration of Things, as well as how to discovery Things by JID, location, etc [28]. Moreover, XEP-0045 defines an XMPP protocol extension for multi-user text chat, whereby multiple XMPP users can exchange messages in the context of a room or channel [29]. XEP-0030 specifies an XMPP protocol extension for discovering two kinds of information about other XMPP entities, namely the identity and capabilities of an entity and the items associated with an entity [30]. Furthermore, publish/subscribe is defined in the XEP-0060 associated with two subset protocols XEP-0163 and XEP-0248 [31]–[33]. XEP-0060 defines an XMPP protocol extension for generic publish/subscribe functionality where subscribers can subscribe to content nodes according to their interest and receive information from a particular node or a list of nodes. The XEP-0163 provides a publish/subscribe subset that enables a XMPP user's JID to function as a virtual publish/subscribe service, greatly easing the discovery of account owner's syndicated data and event notifications. The XEP-0248 species the nature of collection nodes that is useful to organize the publish/subscribe nodes to build relationships between nodes so that it simplifies the subscription process of the subscribers.

To the best of our knowledge, there are also some works on the application of XMPP for IoT. For example, Finland Tampere University released a function reduced XMPP protocol stack that allows Contiki devices to achieve point-to-point communication with XMPP instant messaging software in an unnamed way [34], [35]. In addition, the work by Adrian Hornsby and Eloi Bail presents a light, open source, uXMPP messaging and presence implementation for Contiki operating system enables rich web-service like applications to be implemented even in constrained devices such as WSN nodes [36]. However the uXMPP cannot fully support service discovery, publish/subscribe and semantic annotation. Also the work of XMPPClient for mbed provided rudimentary but lightweight XMPP implementations that have limitations, *e.g.*, cannot support IQ stanzas and complex presence stanzas, as well as handle error streams and stanzas [37]. Chatty Things presents an optimized uXMPP that supports for essential extensions and optimizes for small RAM/ROM footprints. Also it realizes a XMPP-compliant message reduction without using complex and costly techniques like XML compression [38]. Furthermore, in order to implement the publish/subscribe functionality in the IoT, a horizontally layered and XMPP publish/subscribe scheme based architecture is proposed for mobile participatory sensing in [39]. In this communication system, the smartphones with rich resources can publish events autonomously to the publish/subscribe service as publishers or subscribe to event notifications and event contents as subscribers. Similarly, in [40], a service platform with XMPP publish/subscribe functionality is implemented based on Mobilis platform [41], the service adds a consumer to the list of interested clients when the consumer registers at appropriate service for data published by data sources. The above initial experiments of using XMPP for the IoT just showed the feasibility of running minimized XMPP on resource-constrained devices so that further studies are needed. Moreover, in the existing works, the publish/subscribe functionality is mainly implemented on smart objects, so it is necessary to implement this functionality on the resource-constrained devices to discovery and integrate them to the Internet, as well as facilitate the seamless interaction with them.

C. XMPP COMMUNICATION SYSTEM

To better illustrate our XMPP publish/subscribe scheme, we would like to give a brief introduction of XMPP communication network system, where there are three roles, namely client, server, and gateway. Each communication entity (such as a client or server) has a unique address called the Jabber identity or JID. A valid JID consists of a set of aligned elements, including localparts, domainparts, and resourceparts. Where the local part is separated from the domain part by the "@" symbol, while the domain part and the resource part are separated by a "/" symbol. The complete format of a JID is: localpart@domainpart/resourcepart. The entities communicate with each other using two open-ended XML streams that consist of three types of stanzas including <presence/>, <message/> and <iq/> (info/query). In the communication system, the client communicates with the other servers and clients in the network over XMPP by exchanging the XML stanza. The responsibilities of server are: 1) to save the messages of clients, 2) to manage connections between entities, 3) to route the messages between clients. The gateway is used to communicate with different instant messaging protocols. The XMPP communication can be two-way communication between any of the three. Clients need to connect to a server over TCP/IP communication for XML streams transmission, and servers also connected over TCP/IP communication for inter domain communication. The XMPP communication system architecture is shown in Fig. 1. We notice that XMPP client can directly communicate with an XMPP server using XMPP technology, however, communication between XMPP client and other non-XMPP client, and between XMPP server and non-XMPP server can only be achieved through the gateway. In particular, it is worth to address that the XMPP clients can not directly communicate with each other, the communication must be processed by the

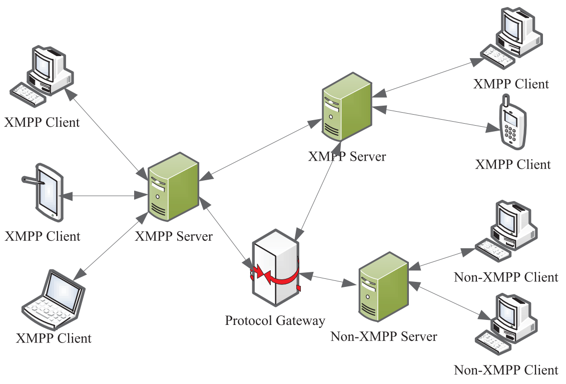


FIGURE 1. XMPP communication system architecture.

server to achieve the information exchange between them. To ensure the security of communications, the communications between XMPP client and XMPP server, and between XMPP server and XMPP server adopt Simple Authentication and Security Layer (SASL) and encrypted with Transport Layer Security (TLS) to authenticate its account and encrypt data, respectively [42], [43].

III. PROPOSED XMPP PUBLISH/SUBSCRIBE SCHEME

In this section, based on the original XMPP standards, we present a lightweight XMPP publish/subscribe scheme by considering the visions of IoT.

A. COMMUNICATION SYSTEM STRUCTURE

The publish/subscribe service of the original XMPP protocol concentrates on applications in the Internet, which is too heavy to run in the resource- and energy- constrained devices. Driven by this, we optimize and improve the original XMPP publish/subscribe extension protocol according to the requirements of IoT. The proposed XMPP-based publish/subscribe system of the IoT devices is still composed of publishers, subscribers, and XMPP servers. In the system, the data exchange is performed either periodically or upon any value change. Subscribers only need to send the interested items to the server to request the services, the publisher will objectify the underlying information to provide periodic and event-driven data publication services and publish the objectified messages when the subscription rules are satisfied. As shown in Fig. 2, the server not only serves as a data transfer channel, but also manages and maintains the publish/subscribe relationship, as it pushes messages to all subscribed subscribers after receiving messages from publishers and provides proxy sleep service when the publishers sleep. In the XMPP communication system, a XMPP client can act as a subscriber or as a publisher or as both the publisher and the subscriber at the same time.

In this scheme, inheriting the advantages of XEP-0060, the server will undertake the publish/subscribe relationship management functions as well as collect and send the interested data for several subscribers at an individual defined rate. Hence, the implementation of XMPP on the client's side

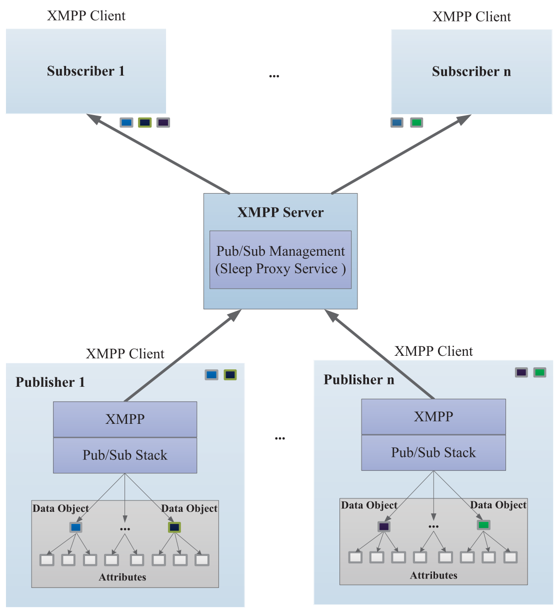


FIGURE 2. Publish/subscribe communication system structure.

is becoming very simple; most of the system complexities reside on the server's side. In addition, since the server manages the subscription relationship between the publisher and the subscriber within a distributed architecture, publishers are allowed to send messages without knowing whether the subscribers exist or not. Consequently, the subscribed publisher only needs to send data to the server and the data will be sent by the server to a subscriber or a list of subscribers. These mechanisms are the basis of publish/subscribe scheme and very suitable for resource-constrained networks. Moreover, the published messages have abundant content in the Internet, while the content of the messages published in the IoT is relatively simple. Thus, in our scheme, the publisher can adjust the published data information based on subscriber's needs to reduce the amount of published data, saving energy for resource-constrained devices.

To implement the above-mentioned methods and functions those are achieved with the XMPP server as mediator and subscription connection manager. Firstly, we propose the resource discovery function to find and connect things together such that the resources of publishers can be retrieved. Secondly, we present the specific procedures of subscription/unsubscription for sensor nodes working both in the active mode and the sleep mode. Then, a multiple publications function is achieved with an extended Pub/Sub management module built-in XMPP server. Last, objects and publish/subscribe XML grammar are designed to implement the lightweight XMPP publish/subscribe scheme.

B. RESOURCES DISCOVERY

Before a subscriber subscribes to a topic or event, it must first discover the published resources for a specific node. Owing to the variability of published resources of IoT devices, we think that the resources discovery information requests should be responded by the publishers themselves, rather than uniformly managed by the server, as shown in Fig. 3. When a subscriber requests a publisher's publish/subscribe resource, the server acts as a transport channel and the publisher responds the request. Here, we suggest that nodes should provide the attributes information including resource name, read and write attributes, subscription attribute, publication cycle, publication cycle configurable attribute and so on. Those attributes will facilitate the acquisition of the subscription service items and configuration of attributes for subscribers.

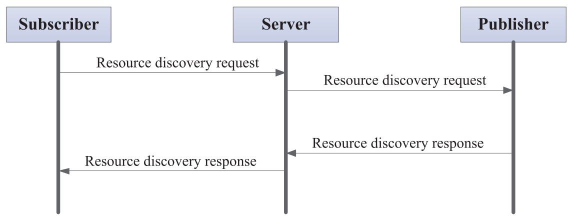

Subscriber
Server
Publisher
Resource discovery request
Resource discovery request
Resource discovery response
Resource discovery response

FIGURE 3. Resource discovery procedures.

C. PUBLISH/SUBSCRIBE PROCEDURES DESIGN

In this scheme, on the one hand, it inherits the merits of XEP-0060 in which the server maintains and manages the publish/subscribe relationship between entities and stores the list of authorized subscribers, since this design meets the characteristics of the resource-constrained IoT scenarios and reduces the complexities on the client's side to save overhead. On the other hand, in the XEP-0060, when a subscriber has subscribed to a node, by configuring the subscription options and filtering the content, the subscriber can receive its desired notifications or payloads sent by the node. This functionality means that the information published to the node by publisher includes not only the information that one subscriber needs, but also some additional information that other subscribers may need. However, unlike the abundant content in the published messages in the Internet, the content of the messages published in the IoT is relatively simple. Considering the practical application scenarios where resource-constrained devices existed, the publisher, which is an entity that is allowed to publish objectified information to the established node, having no need to publish all the information, *e.g.*, sensor nodes using object descriptions don't have to publish all their objects and attributes. Thus, in our proposed scheme, the subscription request of the subscriber needs to be processed by the publisher, which could adjust the published data information according to subscriber's needs. This can be achieved due to the fact that the publisher can learn the subscriber's interested information (*e.g.*, the publication type, subscribed events or attributes) and only publish the interested information rather than all the information when the publisher processes the subscription request from a subscriber that is forwarded by the server. This design yields less energy consumption and is suitable for scenarios, *e.g.*, multiple subscribers are interested in the same topic,

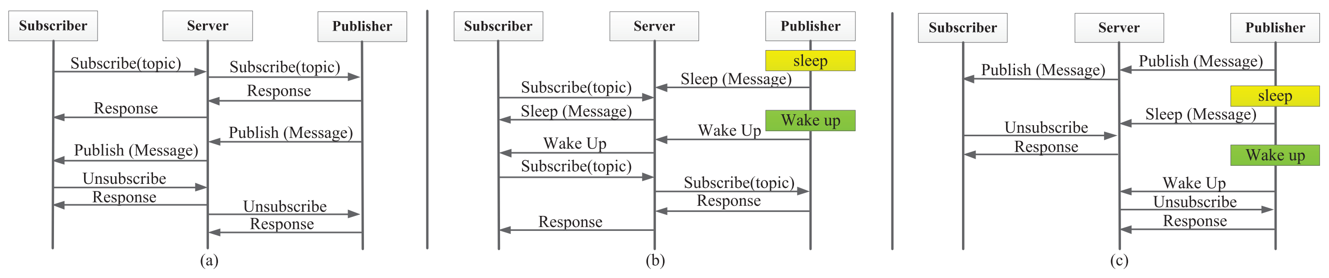


FIGURE 4. **The specific procedures of the proposed XMPP publish/subscribe scheme. (a) The subscriber subscribes/unsubscribes to the active publisher. (b) The subscriber subscribes to the sleeping publisher. (c) The subscriber unsubscribes from the sleeping publisher.**

small and medium size networks. Of course, this advantage is no longer obvious when the needs of subscribers are very different. However, multiple subscribers tend to be interested in the same information in practical application, *e.g.*, multiple subscribers are interested in the temperature data collected by a temperature sensor and need to be periodically notified of the temperature value.

In sensor networks and IoT applications, some nodes work all the time or have no need to support hibernation. On the other hand, some sensor nodes will be idle for a long time if there is no sensing task, so that those nodes are put into sleep to conserve energy. In our method, the sensor nodes act as publishers by default, and the state of the IoT nodes is divided into the active mode and the sleep mode. In the active mode, the nodes can handle any required task and receive or send data. On the contrary, the sleep mode means the nodes conditionally or regularly go to sleep to save as much energy as possible. In this sleeping period, the nodes do not deal any task. Only after waking up that the nodes can work properly and send or receive data. Therefore, the design of XMPP publish/subscribe scheme is different according to the state of the sensor network nodes, as depicted in Fig. 4. The specific design of the publish/subscribe scheme is shown as follows.

1) SUBSCRIPTION AND UNSUBSCRIPTION PROCEDURES FOR ACTIVE NODES

Fig. 4(a) shows the specific procedures in which a subscriber initiates a subscription/unsubscription to an active publisher. First, when a subscriber wishes to subscribe to a publisher, a subscription request is sent to the server and forwarded to the publisher by the server. Second, the subscription connection is established only after the publisher successfully validates the request and sends a subscription response to the server, which will repost the subscription response to the subscriber. Third, the publisher will publish information to the server periodically, and the server broadcasts the message to all the authorized subscribers according to the subscription relationships. Then, when the subscriber wishes to unsubscribe from a publisher, an unsubscription request is sent to the server by the subscriber, and the server will directly response to the unsubscription request and cancel the subscription connection. Finally, after cancelling the connection, the server forwards the unsubscription request to the publisher, which responds to the server an unsubscribe response after the unsubscription request is successfully processed by the publisher.

Throughout the entire process, the publisher does not need to know which subscribers are interested in its data. It just needs to be aware of the publication type, subscribed events or attributes information in the subscription message and sends its data to the server, which will then take care of the data distribution to the subscribers. In the scheme, the server acts as mediator and manages the subscription relationships between publishers and subscribers. Besides, the publishers can adjust the published objects and values based on subscriber's demands. So the scheme could greatly reduce the amount of published data for XMPP publishers and benefit resource-constrained devices.

2) SUBSCRIPTION AND UNSUBSCRIPTION PROCEDURES FOR SLEEPING NODES

In our proposed scheme, the publisher needs to process the subscription request and provide corresponding services according to the subscriber's subscribed objects and attributes. However, if the publisher is sleeping, it cannot process the subscription request, as a result, the subscriber who initiates the subscription/unsubscription request will not be able to get a real time response and the reason why the request failed, which will cause the subscriber to get stuck in a "confused" state. For this reason, by having the publisher send sleep message (carrying the duration of sleep) and wake up message to the server, the server can be aware of the working state of the publisher so that a sleep proxy service is able to be implemented for the sleeping publisher. We explicitly design a publish/subscribe scheme that can support the sleeping sensor devices strongly. It is convenient and effective to maintain and manage the subscription relationships between subscribers and XMPP publishers with sleep function.

Fig. 4(b) depicts the interaction flows in which the subscriber subscribes to the sleeping publisher for retrieving the real-time information of the publisher. Before the publisher goes to sleep, a sleeping message (carrying the duration of sleep) is sent to the server to declare that the publisher is in hibernation. When a subscriber initiates a subscription request to the sleeping XMPP client, a subscription request is sent to the server. The server directly responses to the subscriber with a sleep message carrying the publisher's wake-up time to indicate that the publisher is currently sleeping which is unavailable to be subscribed and when to wake up. Only after the publisher sends a wake-up message to the server and declares it stops hibernation, the subscriber can subscribe to the sleeping publisher just like the publisher in active mode.

Fig. 4(c) describes the exact procedures in which a subscriber unsubscribes from the sleeping publisher. Before the publisher goes to sleep, also a sleeping message with the duration of sleep is sent to the server to declare that the publisher is in hibernation. When a subscriber unsubscribes a connection, a request is made directly to the XMPP server, which will acknowledge the unsubscription request and cancel the subscription connection regardless of whether the publisher is in sleeping state. After the publisher wakes up, the server forwards the unsubscription request to the publisher, which responds to the server an unsubscribe response after the publisher successfully processed it.

D. MULTIPLE PUBLICATIONS

Fig. 5 illustrates how the publishers publish the information to multiple subscribers via the XMPP server with the Pub/Sub management service. In the case where the subscribers subscribe to the interested messages, first, the publishers publish the subscribed messages to the server without knowing who subscribed to the messages. Then, the server forwards the messages to all subscribers. Indeed, this multiple publication service can be implemented due to the fact the server maintains and manages the publication/subscription relationships between the publishers and subscribers, as well as finds corresponding entities during the publish/subscribe process. Therefore, when the publishers' messages are published to the server, the server accesses the subscription relationship list and pushes the messages to one or a large number of subscribers in turn. This function is implemented in XEP-0060, and in order to yield less complexity of publication and energy efficiency of resource limited XMPP publishers, we inherit this mechanism so that a single published message may be distributed to multiple subscribers.

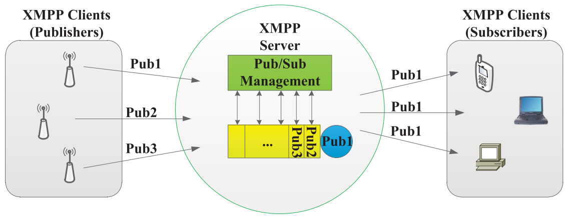


FIGURE 5. The publishers publish the message to multiple subscribers.

E. PUBLISH/SUBSCRIBE OBJECT AND XML GRAMMAR DESIGN

As a system element, the object model is a language and platform independent description and each object has a unique tag. Object definition not only promotes the modularization of the system, but also facilitates the reusability of the system elements. Therefore, we introduce the concept of object so that the implementation of publish/subscribe service can be carried out in the way of manipulating objects. For example, we can easily access the node publish/subscribe to the various attributes of objects in resource discovery and adjust the parameter configuration of the publish/subscribe service. In this publish/subscribe scheme, each device should have a Data Object (DO) to put the same or a certain kind of data into the same module to achieve easily access and use. An object encapsulates the attributes and methods. The attribute represents a property of the object. The method represents the interface function and the access model of the object. Especially for the data object, we need to use the corresponding method to access it. Take the node data services object design as an example, data services object is used to retrieve local data for clients, the communication patterns used to access data can be read, write and publish/subscribe. As shown in Table 1, the object attributes including OutValue, Unit and HighLimit, etc. In order to implement the publish/subscribe service, other objects such as node sleep management object, server publish/subscribe relationship management object and pubsublist structure object are designed, which are similar as the node data services object.

TABLE 1. Node data services object.

Attributes	Methods	Data type	Description
OutValue	Read/PubSub	Float|Int	Output data
Unit	Read/Write	String	Output data unit
Action	Read/Write	Enum	Start/stop sampling (Value: start|stop)
SamplePeriod	Read/Write	Int	Sampling period (ms)
HighLimit	Read/Write	Float|Int	High limit of output data
LowLimit	Read/Write	Float|Int	Low limit of output data
PubSubType	Read/Write	Enum	Data type of Publish/Subscribe (*e.g.*, period|highLimit|lowLimit), The type of Pub/Sub service can be periodic publication or event-driven publication

The improved publish/subscribe grammar for the objects is implemented based on the grammar of the XMPP protocol data service, we extend the <item/> element to support publish/subscribe functionality. The <item/> element may have the following attributes: **ObjectName**, **ObjectID**, **AttrName** and **Type**. Where the **ObjectName** and **ObjectID** denote the class name and attribute instance ID of the data service object. The **AttrName** represents the name of the attribute to which the subscriber subscribes. The **Type** shows the subscription method of the subscriber is a periodic subscription or an event-driven subscription. The above designed attributes are consistent with the characteristics of the data in sensor network.

IV. EXPERIMENT PLATFORM ESTABLISHMENT

In this section, in order to validate the presented publish/subscribe scheme, we develop the test environment and implement the XMPP publish/subscribe scheme on 6LoWPAN platform.

A. XMPP PUBLISH/SUBSCRIBE SERVICE APPLICATION SCENARIO

Fig. 6 illustrates the XMPP publish/subscribe service application scenario, where solid lines denote publish/subscribe routes, and dotted lines denote network connection. In this paper, the XMPP-based publish/subscribe scheme supports publish/subscribe function between the different WSN clients, and between the WSN client and Internet client (such as Personal Computer (PC), mobile phone). In the network, the sensor nodes act as resource-constrained XMPP devices. The 6LoWPAN gateway is not only responsible for the sensor network data collection and broadcasting, but also running XMPP server to support the management of XMPP publish/subscribe function. Moreover, the 6LoWPAN gateway can form a field local area network such that the data can be processed in the field without getting through the Internet. Furthermore, we can set the server in the cloud to support the cloud publishing, and the cloud XMPP server will manage the XMPP subscription connection.

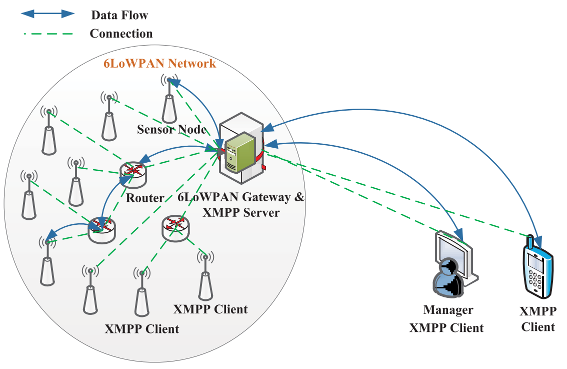


FIGURE 6. XMPP publish/subscribe service application scenario. Solid lines denote publish/subscribe routes, and dotted lines denote network connection.

B. EXPERIMENTAL PLATFORM

For the test of the lightweight XMPP publish/subscribe scheme, the experimental platform is deployed as shown in Fig. 7, including 9 kinds of sensor nodes act as publishers, a 6LoWPAN gateway running XMPP sever and a mobile phone acts as subscriber. In this network, the 6LoWPAN gateway and sensor devices are connected wirelessly over IEEE 802.15.4 using channel 26. The mobile phone is connected to the gateway through the WiFi channel and retrieves sensor resources via the 6LoWPAN gateway.

C. SENSOR NODES

The sensor nodes are equipped with STM32L152 Microcontroller Unit (MCU) with the ARM Cortex-M3 32-bit core operating at 32 MHz frequency, 16 KB RAM, and 128 KB FLASH, also equipped with radio module UZ/CY2420 to support IEEE 802.15.4-compliant radio transceiver. Afterwards, all sensor nodes run an XMPP software client on top of a 6LoWPAN implementation, through which they can publish sensed data or receive control commands.

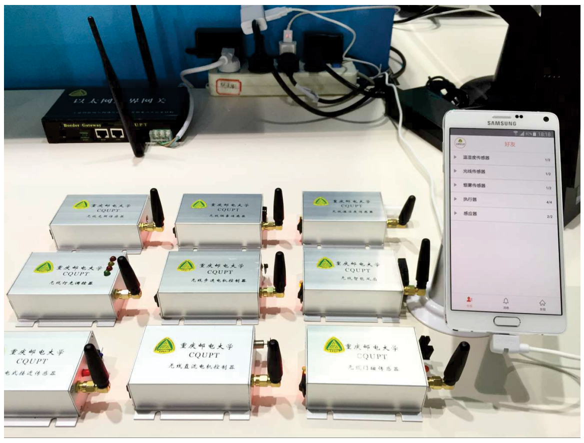


FIGURE 7. Experimental platform for XMPP publish/subscribe scheme test.

D. XMPP SERVER DEVELOPMENT

At present, some organizations and the companies have developed the XMPP server on Linux platform in PC. However, the highly resource occupied XMPP server with multi service functions is not suitable for running on embedded devices. So that it is essential to develop a sensor network XMPP server which is able to run on the gateway or Internet. We developed the gateway that supports OpenWrt based on RT5350, the gateway not only just collects and forwards data, but also runs XMPP sever. Fig. 8 shows the XMPP server software architecture.

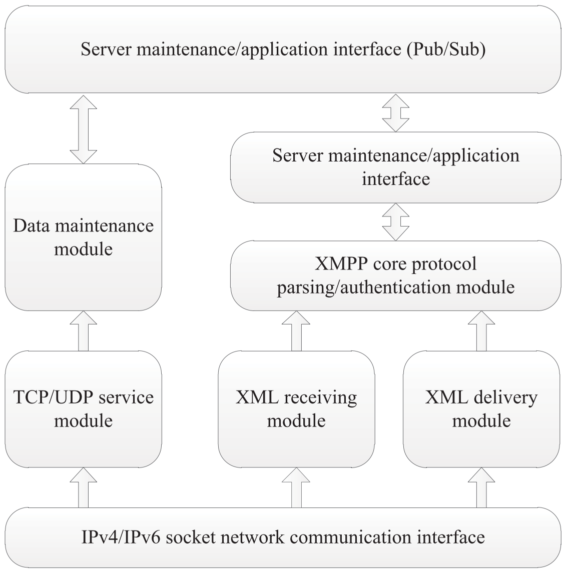

Server maintenance/application interface (Pub/Sub)
Server maintenance/application interface
Data maintenance module
XMPP core protocol parsing/authentication module
TCP/UDP service module
XML receiving module
XML delivery module
IPv4/IPv6 socket network communication interface

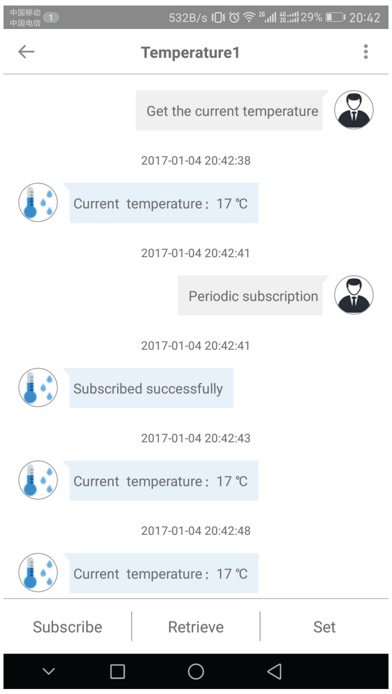
FIGURE 8. XMPP server software architecture.

V. PERFORMANCE EVALUATION

In this section, we present evaluation results to illustrate the performance of the proposed XMPP publish/subscribe scheme for resource-constrained IoT devices.

A. FUNCTION IMPLEMENTATION

Based on the deployed network, we carry out a series of functional tests for XMPP core protocol, including the verification of login process, getting the roster group, getting the online roster, stanzas exchanging and so on, as well as the UDP transmission, read and write the current sensed data service, read and write subscription cycle service, publish/subscribe service and so on. Specially, for the functional tests of XMPP publish/subscribe service, the sensor temperature node is selected as the test node. We test whether the presented scheme supports the publish/subscribe function for the temperature information collected by the sensor node. Fig. 9 presents the test results, from which one can see that the subscribed temperature node will periodically send the collected temperature data to the mobile client per 5 seconds after the node is successfully subscribed. Correspondingly, the mobile client is able to unsubscribe from the temperature node at any time. Moreover, other read and write function tests are similar with the test of publish/subscribe function. The results of those tests not only verify the basic services of the slim and trimmed XMPP protocol, but also verify the proposed XMPP publish/subscribe function.


532B/s 29% 20:42

Temperature1

Get the current temperature

2017-01-04 20:42:38

Current temperature : 17 ℃

2017-01-04 20:42:41

Periodic subscription

2017-01-04 20:42:41

Subscribed successfully

2017-01-04 20:42:43

Current temperature : 17 ℃

2017-01-04 20:42:48

Current temperature : 17 ℃

Subscribe | Retrieve | Set

FIGURE 9. Functional test for the XMPP publish/subscribe function.

B. CODE VOLUME EVALUATION

Since the resource-constrained device in IoT has small memory size and limited computing resources. We should analyze which memory footprint is reachable for a minimized XMPP client that supports a tiny embedded hardware platform equipped with 6LoWPAN stack. The overall memory footprint of the protocol stack is 43484 bytes FLASH, 8197 bytes RAM. Fig. 10 shows the memory footprint rate of XMPP for STM32L152. It can be seen that the memory footprint of the XMPP extension module is relatively low whether its FLASH or RAM. For the sensor nodes that are equipped with STM32L152 core chip with 16 KB RAM and 128 KB FLASH, the protocol stack only occupied 33% FLASH, 50% RAM of the chip resources, noting that the protocol stack can be implemented on devices with only limited resources to satisfy the needs of IoT.

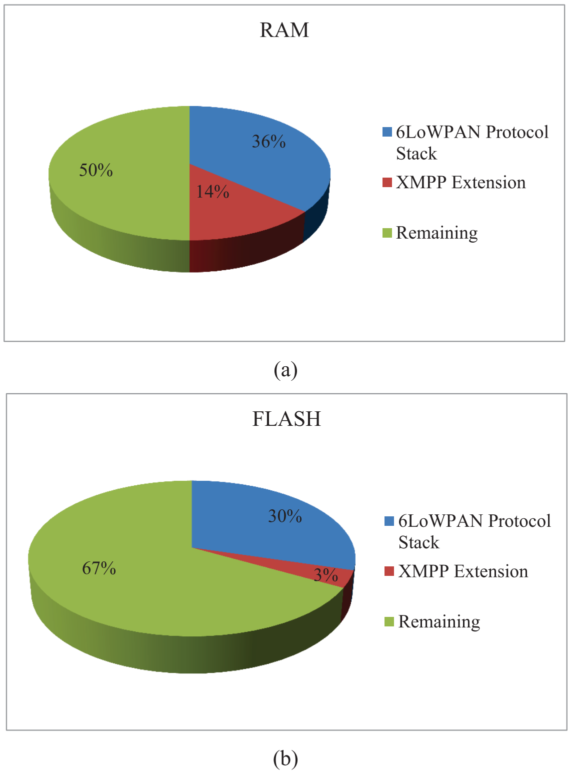


FIGURE 10. Memory footprint rate of XMPP for STM32L152. (a) RAM. (b) FLASH.

C. FUNCTIONAL RESPONSE TIME EVALUATION

Owing to the limited processing capacity of embedded devices and the instable wireless data transmission, it is vitally important to test the response time of the protocol stack to illustrate its impact on the network when applying XMPP protocol to the IoT, so that the feasibility of the application in IoT can be verified. In IoT application scenarios, IoT devices interact with the server frequently. If a functional response time is too long, it will have a dramatic impact on the performance of the network. Therefore, for the resource-constrained nodes, several key test results of the Node-PC communication response time are tabulated in Table 2. The tested communication response time is the time for the PC to access the node through the gateway without interference

TABLE 2. Node-PC communication interaction time.

Test item	Resource discovery	Object attributes acquisition	Object attributes configuration	Subscription	Unsubscription	Information publication	Sleep proxy service
Time	0.66 s	0.48 s	0.52 s	0.54 s	0.22 s	0.50 s	0.22 s

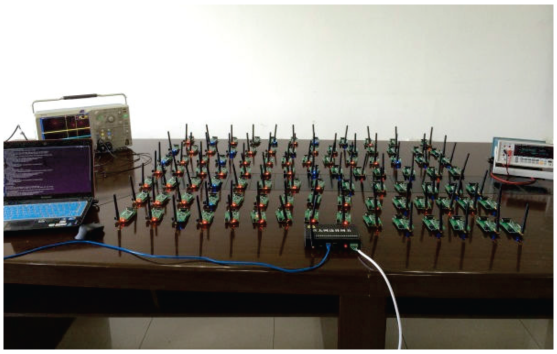


FIGURE 11. 6LoWPAN-based experimental network of resource-constrained IoT devices.

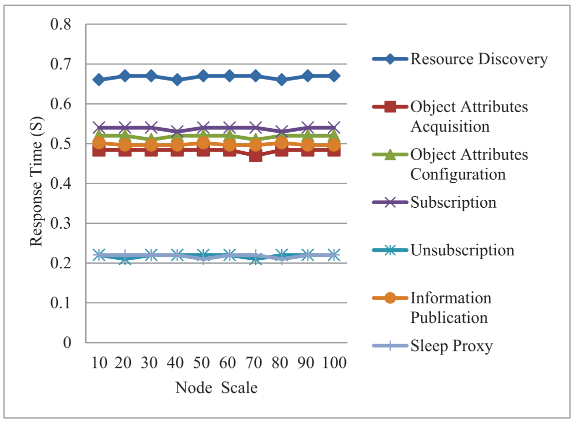


FIGURE 12. Functional response time with the expansion of network scale.

disturbances. From Table 2, it is easy to be seen that the requests can be responded in a second. For instance, the response time of subscription and unsubscription are 0.54 s and 0.22 s respectively. Besides, it is interesting to observe that the response time of unsubscription and server's sleep proxy is less than the response time of other functions. This is attributed to the fact that the server directly response to the requests of subscribers without receiving the permission of publishers. Specifically, in order to further evaluate the performance of XMPP protocol with the expansion of network scale, we have set up different scale of the sensor network that is shown in Fig. 11 to carry out the tests. The tested nodes are temperature and humidity sensor nodes, which are about 10 cm apart. Through the test, we get 10 sets of data for each function with different network scale, as shown in Fig. 12. Note that in the given topology and network setup during the test, the interaction time has nothing to do with the expansion of network scale within the throughput of the network. Therefore, the proposed XMPP protocol with publish/subscribe function can achieve near real-time communication between the entities.

D. ENERGY CONSUMPTION EVALUATION

The sensor nodes in IoT are usually battery powered and limited by resources. Obviously, energy consumption is a noteworthy factor that affects the lifetime of the network. In this paper, the XMPP publish/subscribe scheme has two benefits for energy saving. Firstly, similar to XEP-0060, the server will maintain and manage the publish/subscribe relationships between publishers and subscribers, so that the publishers only need to care about whether their data is subscribed and publish each data to the server, the server will broadcast the data to all the authorized subscribers. What's more, in our scheme, the publisher can adjust the published data information based on subscriber's needs to decrease the amount of published data, which can reduce the power consumption of resource-constrained nodes. Secondly, sensor devices are generally used for data acquisition and distribution, without any doubt, most sensor devices should support the sleep mode to work long hours. Table 3 indicates that the power consumption of the Radio Frequency (RF) transceiver and processor is reduced greatly when the experimental node in the sleep mode. For example, the current of STM32L152 MCU in Low-power sleep mode is 4 uA compared to 9 uA in Low-power run mode. However, to our best knowledge, the current XMPP protocol used in the IoT hasn't implement the publish/subscribe function for sleeping nodes yet. Thus, we improve the XMPP publish/subscribe messaging system to support the sensor nodes to work in the sleep mode. Table 4 presents the average power of temperature and humidity sensor nodes in different modes within 15 seconds. The deep sleep mode indicates that the nodes do nothing. The acquisition mode namely the nodes collect data once in a cycle. The transmit mode is that the nodes do not collect data in one cycle but send data once. Normal work mode means that the nodes collect and send data once in the cycle, while the rest of the time standby. The sleep mode, which is, the nodes become asleep for a long time and wake up to work for a short time. We can see that from Table 4, compared to 136 mW average power in normal work mode, the average power is only 0.457 mW when the node operating in the sleep mode. Note that the power consumption is significantly reduced in the sleep mode. For instance, we flexibly set the devices to work for a total of 2 hours per day in normal work mode to support the publish/subscribe management of the devices, such as subscribing to a device or configuring a subscription cycle, while the rest 22 hours the devices work in the sleep mode. Under this setting, the nodes can be well managed and the publish/subscribe functionality can be implemented. In contrast to the original scheme, where the nodes always

TABLE 3. RF Transceiver and processor performance parameters.

UZ/CY2420 RF Transceiver	Transmission power range: -40 dBm - 3 dBm	TX: 17.5 mA; RX: 16 mA; Low-power sleep mode: 3 uA
	2.4 GHz IEEE 802.15.4-2006	
	Receiving sensitivity: -95 dBm	
STM32L152 MCU	Ultra-low-power platform	Low-power sleep mode: 4 uA; Low-power run mode: 9 uA
	ARM 32-bit Cortex-M3 CPU	
	Reset and supply management	

TABLE 4. Power consumption of temperature and humidity sensor nodes in different mode.

Working mode	Deep sleep mode	Acquisition mode	Transmit mode	Normal work mode	Sleep mode
Sleep	√				√
Capture		√		√	√
Transmit			√	√	√
Power	0.404 mW	38 mW	82 mW	136 mW	0.457 mW

work in normal work mode to provide publish/subscribe service, the work time of the resource-constrained devices can be extended more than 10 times, as shown in Fig. 13.

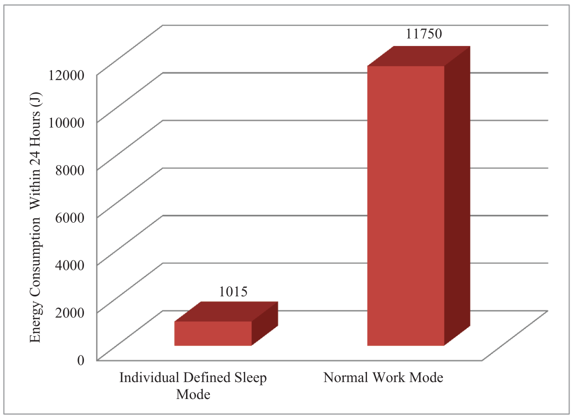


FIGURE 13. The power consumption of the publish/subscribe service between the individual defined sleep mode and the normal work mode.

E. PROTOCOL COMPARISON

In the application layer of the IoT, the established and standardized protocols are not unified. There are many protocols, such as CoAP, MQTT, AMQP, RESTFUL Services, and so on, and each of these protocols performs well in specific scenarios and environments. Thus, for all the IoT applications, it is not feasible to have a single solution [44]. Table 5 presents a brief comparison among the common IoT application protocols, and we can note that the improved XMPP protocol with publish/subscribe functionality for IoT is superior to other application layer protocols.

The XEP-0060 specifies a full-featured and complete extension of the publish/subscribe functionality to broadcast events and it can be applied to many application scenarios. However, the full publish/subscribe protocol is often thought to be very complicated. Thus, it has not been widely implemented [32]. What's more, it is obviously that the full protocol without improvement is unsuitable for implementation in a resource-constrained scenario. Comparing with the publish/subscribe scheme in the XEP-0060, it is worthy to note that our proposed scheme has the following advantages: 1) It is specifically optimized for the field of IoT and completely lightweight. 2) The publisher can adjust the published data information according to subscriber's needs without sending all the objects and attributes information to the node automatically or periodically, so that massive energy can be saved for resource-constrained devices. 3) It can explicitly support the sleeping publisher to achieve more effective interaction between subscriber and publisher with sleep function. 4) Our downsized protocol can not only achieve the event-driven publication, but also implement the periodical publication functionality. On the other hand, further improvements need to be made. As mentioned in the future work, for example, although authentication is used in our system to ensure the security of accounts, there are no corresponding measures to ensure the security of end-to-end data transmission. Besides, the XMPP publish/subscribe functionality in the large-scale sensor networks has not yet implemented.

TABLE 5. Comparison among the IoT application protocols.

Protocols	TCP	UDP	XML	Event-driven Publish/Subscribe	Periodical Publish/Subscribe	Request/Response	Explicit Sleep Function
CoAP		√				√	
MQTT	√			√			
RESTFUL Services	√		√			√	
AMQP	√			√			
Standard XMPP	√		√	√		√	
Proposed XMPP	√	√	√	√	√	√	√

To the best of our knowledge, [39] and [40] also implemented XMPP publish/subscribe functionality. [39] has focused on implementing the XMPP publish/subscribe functionality on smartphones with rich resources and the event-driven publication so that an XMPP publish/subscribe based participating sensing architecture was provided. Compared with [39], our work focused on proposing a lightweight XMPP publish/subscribe scheme for resource-constrained IoT nodes, which might even go to sleep periodically or conditionally. Besides, to satisfy the periodic data transmission in industrial network, we have optimized the scheme to support both the event-driven publication and periodic publication. The achieved experimental results and performance evaluations showed the efficiency of our scheme for resource-constrained IoT networks. In [40], to evaluate their first implementation, the authors measured the event propagation latency between one data generator and 20 data consumers, which both were deployed on an Apple MacBook Pro with 2.3 GHz quad-core processor and 8 GB RAM. Besides, the XMPP server was deployed on a virtual machine with 2.4 GHz single core processor and 2 GB RAM. Compared with the experimental system in [40], we have concentrated on resource limited scenarios and implemented the XMPP publishers on sensor nodes that equipped with STM32L152 MCU with ARM Cortex-M3 core operating at 32 MHz frequency, 16KB RAM, and 128 KB FLASH, as well as implemented the XMPP sever on RT5350-based gateway. In addition, [40] achieved the nearly instant delivery that only averagely 108 ms delay between data generator and data consumers when the Apple MacBook Pro was deployed almost 10 hops away from the server. In our work, the interaction between publisher and subscriber can be done in a second, for example, 480 ms for a subscriber to retrieve attributes of a publisher. However, the longer interaction time is still acceptable due to the fact that resource-limited devices have smaller memory size and slower processing speed. This is a tradeoff between the response time and the processing capacity of limited device. Thus, our downsized XMPP protocol with proposed lightweight publish/subscribe scheme is promising to benefit the IoT development.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a lightweight XMPP publish/subscribe scheme for resource-constrained IoT devices. We optimized and improved the existing publish/subscribe scheme of the XMPP protocol to support the event-driven publication and periodic data publication in IoT. In this scheme, inheriting the advantages of XEP-0060, the server maintains and manages the publish/subscribe relationships with multiple subscribers and forwards data to clients, so that most of the system complexities reside on the server's side. Besides, considering the practical application in resource-constrained networks, the publisher does not periodically or conditionally publish all objects and its attributes, but triggers the services based on the publication type, events and attributes subscribed by subscribers to save energy for resource-constrained devices. In addition, the publish/subscribe function of the sleeping clients is supported to extend the lifetime of the IoT devices. Furthermore, the achieved test results and performance evaluations validate our work on improving XMPP publish/subscribe scheme in IoT.

In the future work, to make this XMPP publish/subscribe scheme more suitable for the IoT and achieve the reliable and efficient data transmission, we believe that the following research issues need to be given consideration with higher priorities: 1) In this paper, we have implemented a XMPP publish/subscribe service based on UDP for resource-constrained devices. Nevertheless, XMPP protocol does not support Quality of Service (QoS), packet loss will be caused under the condition of poor network quality, so how to implement the QoS module to ensure the reliability of the network and improve the stability of the system will be our concern. 2) Considering the characteristics and requirements of IoT, where communication not only requirements data value transmission, but also requirements semantic information exchange, efforts can be made to recommend a unified data description format by working with standardization organization such as OPC Unified Architecture. 3) We have conducted a preliminary experimental test in the XMPP communication system, where hundreds of devices are connected. Whereas IoT networks can be extremely large with thousands or millions of devices, thus the implementation of XMPP protocol in large-scale sensor networks will be the issue that we are going to study. 4) Security is an important feature of the IoT protocols and XMPP protocol. How to implement XMPP-compliant security on resource-constrained devices is our future direction.

REFERENCES

[1] S. Mitchell, N. Villa, M. Stewart-Weeks, and A. Lange, *The Internet of Everything for Cities: Connecting People, Process, Data, and Things to Improve the Livability of Cities and Communities*. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/everything-for-cities.pdf

[2] D. Kyriazis, T. Varvarigou, D. White, A. Rossi, and J. Cooper, "Sustainable smart city IoT applications: Heat and electricity management & Eco-conscious cruise control for public transportation," in *Proc. IEEE 14th Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Madrid, Spain, Jun. 2013, pp. 1–5.

[3] J.-S. Hwang and Y. H. Choe, "Smart cities smart cities seoul: A case study," ITU-T Technol. Watch Rep., Feb. 2013.

[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[5] *The Internet Engineering Task Force (IETF)*. Accessed on Jun. 10, 2017. [Online]. Available: https://www.ietf.org/about/

[6] G. Montenegro, N. Kushalnagar, J. W. Hui, and D. E. Culler, *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks*, document RFC 4944, IETF, Sep. 2007.

[7] T. Winter, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, document RFC 6550, IETF, Mar. 2012.

[8] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, document RFC 7252, IETF, Jun. 2014.

[9] S. Bandyopadhyay and A. Bhattacharyya, "Lightweight Internet protocols for Web enablement of sensors using constrained gateway devices," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, San Diego, CA, USA, Jan. 2013, pp. 334–340.

[10] IBM. *Message Queue Telemetry Transport*. Accessed Jun. 10, 2017. [Online]. Available: http: //mqtt.org/

[11] S. Yegulalp. *Review: WAMP Stacks for Web Developers*. Accessed on Jun. 10, 2017. [Online]. Available: http://www.infoworld.com/article/2616867/web-development/review–wamp-stacks-for-web-developers.html

[12] A. Hornsby and R. Walsh, "From instant messaging to cloud computing, an XMPP review," in *Proc. 14th IEEE Int. Symp. Consum. Electron. (ISCE)*, Braunschweig, Germany, Jun. 2010, pp. 1–6.

[13] B. Sat and B. W. Wah, "Analysis and evaluation of the skype and Google-talk voip systems," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Toronto, ON, Canada, Jul. 2006, pp. 2153–2156.

[14] M. Grossman, "Facebook chat," *Accessworld, Technol. Consum. Vis. Impairments*, vol. 11, no. 2, p. 8, May 2010.

[15] XMPP. *An Overview of XMPP*. Accessed on Jun. 10, 2017. [Online]. Available: https://xmpp.org/about/technology-overview.html

[16] XMPP. *IoT*. Accessed on Jun. 10, 2017. [Online]. Available: https://xmpp.org/uses/internet-of-things.html

[17] M.-K. Liao and Y.-C. Chen, "An XMPP-based XML representation middleware to build universal service-oriented gateway in M2M environment," in *Proc. IEEE Int. Conf. Internet Things (IThings), Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Soc. Comput. (CPSCom)*, Taipei, Taiwan, Sep. 2014, pp. 193–200.

[18] *Jabber Inc*. Accessed on Jun. 10, 2017. [Online]. Available: http://www.cisco.com/c/en/us/about/corporate-strategy-office/acquisitions/jabberinc.html

[19] P. Saint-Andre, *Extensible Messaging and Presence Protocol (XMPP): Core*, document RFC 6120, IETF, Oct. 2015.

[20] P. Saint-Andre, *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*, document RFC 6121, IETF, Oct. 2015.

[21] P. Saint-Andre, *Extensible Messaging and Presence Protocol (XMPP): Address Format*, document RFC 6122, IETF, Oct. 2015.

[22] P. Saint-Andre, *Extensible Messaging and Presence Protocol (XMPP): End-to-End Signing and Object Encryption*, document RFC 3923, IETF, Oct. 2004.

[23] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the Internet of Things," *Trans. IoT Cloud Comput.*, vol. 3, no. 1, pp. 11–17, Jan. 2015.

[24] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for wireless sensor networks," in *Proc. 3rd Int. Conf. Commun. Syst. Softw. Middleware Workshops (COMSWARE)*, Bengaluru, India, Jan. 2008, pp. 791–798.

[25] M. Kirsche and R. Klauck, "Unify to bridge gaps: Bringing XMPP into the Internet of Things," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, Lugano, Switzerland, Mar. 2012, pp. 455–458.

[26] B. Xuefu and Y. Ming, "Design and implementation of Web instant message system based on XMPP," in *Proc. IEEE 3rd Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Beijing, China, Jun. 2012, pp. 83–88.

[27] P. Saint-Andre and R. Strid, *Internet of Things Special Interest Group (IoT SIG)*, document XEP-0381, XMPP Standards Foundation, Nov. 2016.

[28] P. Waher and R. Klauck, *Internet of Things—Discovery*, document XEP-0347, XMPP Standards Foundation, Aug. 2016.

[29] P. Saint-Andre, *Multi-User Chat*, document XEP-0045, XMPP Standards Foundation, Dec. 2016.

[30] J. Hildebrand, P. Millard, R. Eatmon, and P. Saint-Andre, *Service Discovery*, document XEP-0030, XMPP Standards Foundation, Oct. 2016.

[31] P. Millard, P. Saint-Andre, and R. Meijer, *Publish-Subscribe*, document XEP-0060, XMPP Standards Foundation, Dec. 2016.

[32] P. Saint-Andre and K. Smith, *Personal Eventing Protocol*, document XEP-0163, XMPP Standards Foundation, Jul. 2010.

[33] P. Saint-Andre, R. Meijer, and B. Cully, *Pubsub Collection Nodes*, document XEP-0248, XMPP Standards Foundation, Sep. 2010.

[34] *The Contiki OS*. Accessed on Jun. 10, 2017. [Online]. Available: http://www.contiki-os.org/

[35] A. Hornsby, "XMPP message-based MVC architecture for event-driven real-time interactive applications," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, Jan. 2011, pp. 617–618.

[36] A. Hornsby and E. Bail, "μXMPP: Lightweight implementation for low power operating system Contiki," in *Proc. Int. Conf. Ultra Modern Telecommun. Workshops (ICUMT)*, Saint Petersburg, Russia, Oct. 2009, pp. 1–5.

[37] M. Schulz. *Mbed Cookbook—XMPPClient*. Accessed on Jun. 10, 2017. [Online]. Available: http://mbed.org/cookbook/XMPPClient

[38] R. Klauck and M. Kirsche, "Chatty things - Making the Internet of Things readily usable for the masses with XMPP," in *Proc. 8th Int. Conf. Collaborative Comput. Netw., Appl. Worksharing (CollaborateCom)*, Pittsburgh, PA, USA, Oct. 2012, pp. 60–69.

[39] R. L. Szabo and K. Farkas, "A publish-subscribe scheme based open architecture for crowd-sourcing," in *Proc. 19th EUNICE Workshop Adv. Commun. Netw. (EUNICE)*, Chemnitz, Germany, Aug. 2013, pp. 287–291.

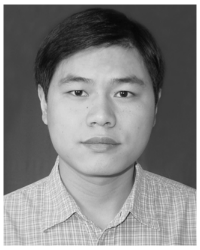
[40] S. Bendel, T. Springer, D. Schuster, A. Schill, R. Ackermann, and M. Ameling, "A service infrastructure for the Internet of Things based on XMPP," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, San Diego, CA, USA, Mar. 2013, pp. 385–388.

[41] *Mobilis Platform*. Accessed on Jun. 10, 2017. [Online]. Available: https://github.com/mobilis/ mobilis/wiki/About-mobilis

[42] T. Dierks, *The Transport Layer Security (TLS) Protocol Version 1.2*, document RFC 5246, IETF, Aug. 2008.

[43] A. Melnikov and K. D. Zeilenga, *Simple Authentication and Security Layer (SASL)*, document RFC 4422, IETF, Jun. 2006.

[44] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.



HENG WANG (S'10–M'11) received the B.S., M.S., and Ph.D. degrees in communication engineering from Chongqing University, Chongqing, China, in 2003, 2006, and 2010, respectively. He is currently a Full Professor with the College of Automation, Chongqing University of Posts and Telecommunications. His research interests include wireless sensor networks, industrial Internet of Things, and cooperative communications.

DAIJIN XIONG received the B.S. degree in automation from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2015. He is currently pursuing the M.S. degree with the Chongqing University of Posts and Telecommunications. His research interests include wireless communication systems and protocols, wireless sensor networks, and industrial Internet of Things.

PING WANG received the B.S. and M.S. degrees from Chongqing University, Chongqing, China, in 1983 and 1988, respectively, the Ph.D. degree from the Southwest Jiaotong University, Chengdu, China, in 1994. He is currently a Full Professor with the College of Automation, Chongqing University of Posts and Telecommunications. His research interests include industrial Internet of Things, wireless sensor networks, and wireless industrial networks.

YUQIANG LIU received the B.S. and M.S. degrees in control science and engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2014 and 2017, respectively. His research interests include wireless communication systems and protocols, wireless sensor networks, and industrial Internet of Things.