Internet of Things Cloud: Architecture and Implementation

The authors aim at providing insights about the architecture, implementation and performance of the IoT cloud. Several potential application scenarios of IoT cloud are studied, and an architecture is discussed regarding the functionality of each component. Moreover, the implementation details of the IoT cloud are presented along with the services that it offers.

Lu Hou, Shaohang Zhao, Xiong Xiong, Kan Zheng, Periklis Chatzimisios, M. Shamim Hossain, and Wei Xiang

Abstract

The Internet of Things (IoT), which enables common objects to be intelligent and interactive, is considered the next evolution of the Internet. Its pervasiveness and ability to collect and analyze data that can be converted into information have motivated a plethora of IoT applications. For the successful deployment and management of these applications, cloud computing techniques are indispensable since they provide high computational capabilities as well as large storage capacity. This article aims at providing insights about the architecture, implementation, and performance of the IoT cloud. Several potential application scenarios of IoT cloud are studied, and an architecture is discussed regarding the functionality of each component. Moreover, the implementation details of the IoT cloud are presented along with the services that it offers. The main contributions of this article lie in the combination of the Hypertext Transfer Protocol (HTTP) and Message Queuing Telemetry Transport (MQTT) servers to offer IoT services in the architecture of the IoT cloud with various techniques to guarantee high performance. Finally, experimental results are given in order to demonstrate the service capabilities of the IoT cloud under certain conditions.

COMMUNICATIONS STANDARDS

Introduction

With the development of wireless communication technologies, pervasive objects can be interactive and are connected to the Internet. These inter-connected objects with built-in computing, communications, and sensing capabilities constitute the Internet of Things (IoT). In particular, it is estimated that by 2020 the number of IoT devices will be close to 50 billion, while the global population will reach 7.6 billion [1]. These devices can generate huge amounts of data, which usually come in different formats and meanings [2–4]. However, IoT devices usually have very limited capabilities due to their small physical size and energy consumption. Therefore, an IoT cloud is imperative to support the requirements of millions of IoT devices and provide various new and exciting IoT applications for the end-users.

Recently, a plethora of novel ideas has been proposed for the IoT cloud. M2M communications or MTC, which enable direct communication among IoT devices, have attracted significant attention. A standard M2M service layer platform, which is called oneM2M, has been established and developed for the standardization of deployment of IoT services [5]. A detailed discussion of MTC can be found in [6]. Subscription control, congestion, and overload control are described in detail, as well as a new solution to the latter issue. As for the IoT itself, the authors in [7] and [8] provide a comprehensive survey and discuss the feasibility as well as enabling technologies for the IoT cloud. Meanwhile, inspired by the IoT cloud, certain research on related applications is conducted in [9–11]. Moreover, in order to efficiently manage IoT workloads, several IoT cloud platforms have been proposed. For example, the authors in [12] proposed a platform dubbed servIoTicy with data stream processing capabilities for the IoT cloud, and they carried out a performance evaluation of the proposed platform. However, the relevant literature on the architectural design and implementation details for the IoT cloud are scarce to date. An architecture that can support millions of concurrent IoT devices as well as diverse IoT applications is highly desirable.

Against the above background, this article aims at proposing an IoT cloud architecture based on both the Hypertext Transfer Protocol (HTTP) and Message Queuing Telemetry Transport (MQTT) protocols, and other relevant techniques to guarantee high performance. First, certain application scenarios and requirements of the IoT cloud are presented. A generic IoT system is then proposed, and we discuss the supporting IoT infrastructure. Moreover, the implementation details of the proposed IoT cloud architecture are discussed, followed by a presentation of how the IoT system is built and how a message broker for MQTT servers is built by employing the Redis cluster database. Finally, a number of experiments is conducted to evaluate the performance of the IoT cloud, in terms of the average response time/average transmission latency, throughput, and CPU utilization.

The rest of this article is organized as follows. Section II provides typical application scenarios for the IoT cloud and the corresponding performance requirements. The proposed IoT cloud architecture is then described. We discuss in detail the overall implementation of the proposed IoT cloud, while we then report our experimental results. Finally, we conclude this article and provide an outlook for the future.

Typical Applications of the IoT Cloud

The use of an IoT cloud has a significant impact on the performance of the supported IoT applications. Thus, several typical application scenarios of the IoT cloud are discussed in this Section.

Smart Buildings: Smart buildings can adapt to internal and external environmental changes without human intervention in order to provide comfort to the occupants, while taking into consideration financial and energy requirements. Ubiquitous devices can monitor the entire building at all times, generating large amounts of data, which can then offer notification services in case of emergency incidents or critical situations after

Lu Hou, Shaohang Zhao, Xiong Xiong and Kan Zheng are with Beijing University of Posts and Telecommunications.

Periklis Chatzimisios is with the Alexander Technological Educational Institute of Thessaloniki (ATEITHE).

M. Shamim Hossain is with King Saud University.

Wei Xiang is with James Cook University.

This work is funded by Deanship of Scientific Research at King Saud University, Riyadh, Saudi Arabia for funding this work through the research group project no. RGP-228.

Digital Object Identifier: 10.1109/MCOM.2016.1600398CM

Scenario	Typical use case	Device type	User population	Energy consumption	Maintenance cost	Throughput	Tolerable latency	Mobility	Reliability	Security & privacy
Smart building	Water metering	Sensors, meters	Large	Low	Low	Low	High	Fixed	Medium	Low
	Residential monitoring	Sensors	Few	Low	Low	Low	Low	Fixed	High	High
Smart home/ office	Home automation	Intelligent appliances, sensors	Few	High	Low	Low	High	Low	Medium	High
	Smart meeting	Laptops, videos	Medium	High	Low	High	Medium	Fixed	Low	High
Intelligent transportation	Traffic monitoring	Sensors, cameras	Large	High	Low	High	High	Fixed	Low	Low
	Driving assistance	Sensors, vehicles	Few	Medium	Medium	Low	Low	High	High	High
Smart healthcare	Patient monitoring	Sensors, medical equipment	Few	Low	High	Low	Medium	Fixed	High	High
	Vital signal alert	Sensors, medical equipment	Few	Low	High	Low	Low	Medium	High	High

Table 1. Features of typical applications using IoT cloud.

a proper analysis of the gathered data [13].

Smart Home/Office: A smart home establishes a future home environment, where embedded sensors and intelligent appliances are self-configured and can be controlled remotely through the Internet. It enables a variety of monitoring and intelligent control applications that are responsible for the control and management of home resources. More intelligently, an IoT device can be controlled by the IoT cloud to adjust its operation. As a result, a comfortable living environment can be created for all occupants. On the other hand, smart offices aim mainly at easing workloads and improving the productivity of employees at work. With a proper IoT cloud, workers in different organizations or areas can all access office-related services in a convenient and efficient manner.

Intelligent Transportation: Intelligent transportation brings comfort to people traveling within cities or rural areas. Vehicles can be smart if they are equipped with a large number of sensors that monitor the status of the surrounding environment. All sensed data can be collected and uploaded to the IoT cloud. With real-time data processing, the IoT cloud can provide useful assistance to the driver, such as the provision of emergency warnings or optimal path planning, as well as knowledge of road/traffic conditions or traffic accident notifications. The IoT cloud also offers warnings for pedestrians when there are potential life or injury threats by analyzing the data gathered from vehicles, the infrastructure, or even pedestrians.

Smart Healthcare: Smart healthcare applications decrease patients' dependence on care givers and reduce their healthcare costs through efficient use of medical equipment and sensors. Measurements of various biological information such as pulse and blood pressure can be taken by the patients themselves via their smartphones that contain special on-body and near-body sensors. All the measured data are then transmitted to the IoT cloud, where the early detection of life-threatening emergency situations is possible through continuous monitoring and analysis of the data received from the patient being monitored.

For the purpose of better illustration, Table 1 summarizes the features of several typical application scenarios using the IoT cloud.

Architecture of the IoT Systems

In order to support millions of IoT devices, we propose an IoT cloud architecture based on hardware support of the IoT infrastructure. By using virtualization, hardware resources can be well utilized. Consequently, both HTTP and MQTT servers are introduced as the application servers of the IoT cloud. The HTTP servers can provide services for end-users and devices, while the MQTT servers ensure a large number of device connections and real-time communication among devices. Furthermore, some other key components such as the supporting databases are also presented for the sake of functionality, availability, and performance.

IoT Infrastructure

IoT infrastructure is a fundamental component of the entire IoT system since it can sense and perform actions from/to the environment as well as sending information to the IoT cloud. The IoT infrastructure consists of all IoT devices and the supporting access networks. The former is deployed in the application environment, whereas the latter provides communications between IoT devices and the cloud. IoT devices mainly include sensors, actuators, intelligent appliances, etc., and may generate huge amounts of data that are transmitted to the IoT cloud through reliable and

As the hardware resources of the physical machine cannot be fully utilized, there is a significant waste of resources, as well as the problem of low scalability of servers. To tackle these problems, the virtualization technique is used to provide feasible solutions that aim at improving resource utilization for the IoT cloud.

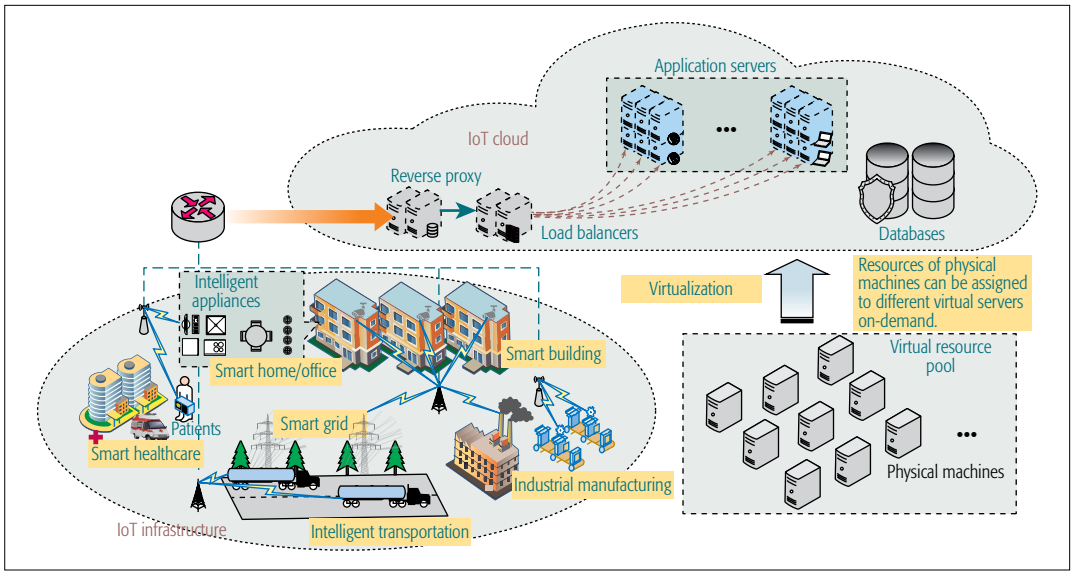

Figure 1. Illustration of the architecture of IoT systems.

efficient access networks. Additionally, control messages may be transferred to IoT devices from the IoT cloud via the same access networks.

IoT Cloud

As illustrated in Fig. 1, the IoT cloud consists of several key components, each of which is composed of multiple servers that perform different tasks. The servers are established as virtual machines (VMs) utilizing virtualization technology. They are independent from each other even if they run on the same physical machine. With these VMs, load balancers/reverse proxy servers, databases, and application servers can be configured. The functionalities of each component are described as follows.

Virtual Resource Pool: As the hardware resources of the physical machine (such as the CPU, memory, and network connectivity) cannot be fully utilized, there is a significant waste of resources, as well as the problem of low scalability of servers. To tackle these problems, the virtualization technique is used to provide feasible solutions that aim at improving resource utilization for the IoT cloud. By means of virtualization, hypervisor software runs on the physical machine as an abstract layer to manage all resources and also to provide an operating environment for various independent guest operating systems (OSs) (known as VMs) that enable dynamic resource allocation. Furthermore, the IoT cloud services can be deployed on VMs instead of directly on physical machines, which helps reduce the usage of physical machines, and thus can deliver high performance at low cost.

In particular, by employing the virtualization technique, a virtual resource pool can be established on several physical machines that contain all the hardware resources and can assign them to different VMs on demand. In this way, all other servers can obtain a proper amount of resources, in accordance with their demands.

Application Servers: Application servers are often considered to be the most important component of the IoT cloud since they are responsible for offering business services to customers. They need to provide facilities and an appropriate environment to run multiple applications based on certain application protocols [14]. The application servers in the traditional cloud are usually based on HTTP. HTTP servers work in a request-response manner through Transmission Control Protocol (TCP) connections with clients. When connections are established, an HTTP server can listen to certain ports for requests from clients and send appropriate responses to the received requests.

However, HTTP is not well suited for the IoT cloud since IoT devices are constrained by their computing, communication, and energy resources. Consequently, another type of application protocol is more appealing for the IoT cloud, i.e., the MQTT protocol [14]. MQTT is designed for resource-constrained IoT devices as a lightweight messaging transportation protocol that operates via a topic-based publish-subscribe mode. This means that when a client publishes a message on a particular topic, all the clients that have subscribed to the same topic can receive this message. A key component that completes the transfer process is regarded as the broker [15], by which one-to-many connections are enabled.

Database: According to various application requirements for data storage, relational and non-relational databases, also known as Structured Query Language (SQL) and NoSQL databases, are optional in the IoT cloud. SQL is designed as a type of programming language for relational databases that can store data in the form of two-dimensional tables. However, the performance of the SQL databases is the main bottleneck for the deployment of real-time IoT applications. Consequently, NoSQL databases are used to provide real-time and high-efficiency services for data storage. These databases allow data to be stored directly in memory or hard disks, and thus the input/output (I/O) speed is significantly improved.

Reverse Proxy and Load Balancing: Due to the large number of IoT devices and users, application servers are required to handle millions of concurrent requests or transfer a massive num-

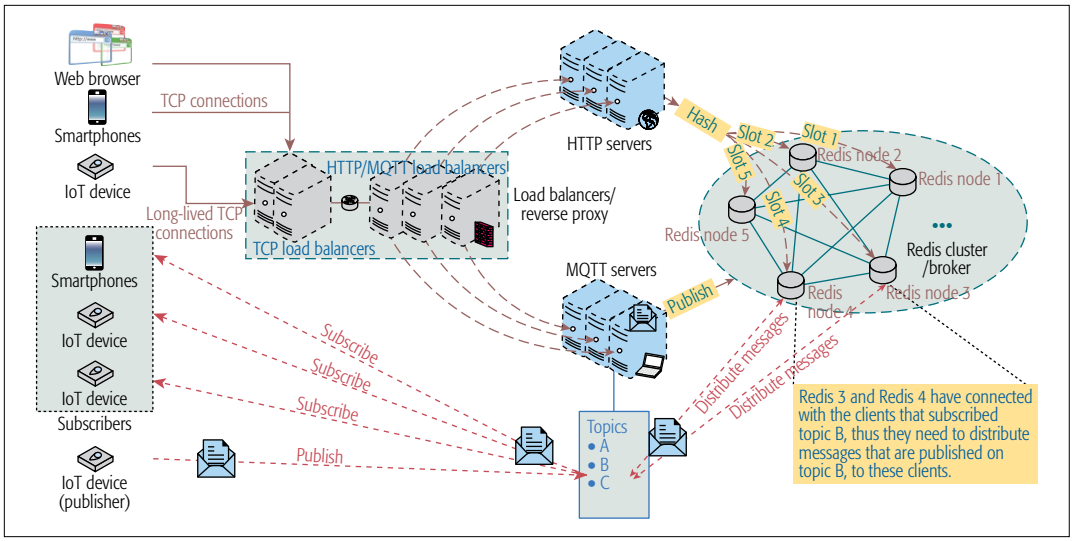

Figure 2. Illustration of the IoT cloud implementation.

ber of messages. These requests or messages are processed without scheduling if load balancing is not enforced. As a result, some servers may be heavily congested due to excessive burdens, and it is possible that new requests or messages sent to the congested servers will be rejected or discarded. Meanwhile, other servers may be idle, although spare resources may actually be enough to process these requests, resulting in a significant waste of resources. Therefore, load balancing is imperative to evenly distribute the workload across multiple backend servers, and to achieve full utilization of all available resources.

Implementation and Services of the IoT Cloud

Aiming at connecting millions of devices and end-users, the proposed IoT cloud is developed with details that are described in this Section. The services that the IoT cloud can provide are also discussed.

Implementation

A virtualization OS such as VMWare vSphere[1] can be used to establish a resource pool with a number of VMs and can directly handle the CPU and memory resources of the physical machines. A server in the IoT cloud can be implemented as one VM. The implementations of different types of servers are described in detail as follows.

Application Servers: The IoT cloud includes the HTTP and MQTT servers that can both be developed using Node.js, which is typically used for developing server applications due to its capability in high concurrency. It runs in the form of an asynchronous event loop that performs all I/O operations with a single thread asynchronously. As a result, application servers are capable of handling a large number of concurrent connections.

HTTP Servers: They apply a flexible web application framework, i.e., Express, in order to work. In this way, the web and mobile applications are easily deployed on an HTTP server, which interacts with clients through a request-response cycle. The HTTP servers offer three different methods, i.e., GET, POST and DELETE for clients to make requests. Clients can obtain resources from the HTTP servers through a GET request. Clients can also send information to the HTTP servers through a POST request. Moreover, a DELETE request enables clients to delete certain resources in an HTTP server. After receiving a request, the HTTP server tries to process the request and send a response back to the clients.

MQTT Servers: They are deployed for instant communication between the IoT devices and end-users by utilizing the MQTT protocol that is a broker-based protocol for publishing/subscribing message transportation. Its publication and subscription are organized based on the notion of "topic", and all packets are published through the broker. As for publication, a topic should be uniquely defined, while for subscription, an MQTT client can subscribe multiple topics at once. The MQTT servers are implemented based on an open-source library in Node.js, i.e., MQTT-connection. In order to enhance real-time performance of the MQTT servers, they have to maintain long-lived TCP connections with clients or devices. Furthermore, the MQTT servers use three levels of quality of service (QoS) to ensure reliability. QoS level 0 means that recipients do not send any acknowledgment to publishers, and all messages are published only once. By contrast, QoS level 1 requires acknowledgments. As for QoS level 2, a handshake mechanism is used to ensure that messages can be successfully delivered to all subscribers. In accordance with the various business requirements, the corresponding QoS level can be configured.

Multiple application servers can constitute a cluster, which allows for simultaneously scaling server programs across multiple parallel processors. The parallel multithreaded machine (PM2) helps form a cluster of multiple HTTP servers. On the other hand, the MQTT server has to act as a cluster by operating in a master-slave mode. In this mode, the MQTT server operates at the master node, which can initiate other servers as the slave nodes. Each MQTT server (master or slave) runs on an individual CPU core.

Database Cluster and Broker: The IoT cloud uses Redis[2] (a NoSQL database) to store data.

The performance of the SQL databases is the main bottleneck for the deployment of real-time IoT applications. Consequently, NoSQL databases are used to provide real-time and high-efficiency services for data storage. These databases allow data to be stored directly in memory or hard disks and, thus, the input/output speed is significantly improved.

[1] http://www.vmware.com/products/vsphere/

[2] http://redis.io/

The IoT cloud should meet the requirements of different IoT applications. By using the services provided by the IoT cloud, diverse applications can be offered to both end-users, developers or managers. These services can be accessed by web browsers or smartphones anytime and anywhere.

By storing all key-value data in the memory, Redis can significantly increase I/O speed. In order to improve the reliability of the database, a Redis cluster with more than one Redis node can be configured in the IoT cloud. Thus, the users can enjoy continuous data services even when one or more Redis nodes are out of order. The Redis cluster is fully connected such that each Redis node is connected with all the others through TCP connections. After forming the Redis cluster, slot share should be configured before the cluster can work properly. The data stored in the Redis cluster are first hashed, e.g., taking the CRC16 of modulo 16384 of the data as the hash slot. By checking in which interval the hash slot is located, the data are then stored in the corresponding Redis node, as depicted in Fig. 2. Since the hash slots of incoming data are uniformly distributed, the load of each Redis node is inherently balanced. From the users' viewpoint, there is no difference in accessing the database, whether it is a single Redis node or a cluster.

On the other hand, Redis works well with MQTT servers, since it can work as a message broker. The load of the MQTT server can then be largely moved to the Redis cluster so that higher concurrency can be achieved. Clients can publish messages on some topics to the MQTT server at first, which then transfers the payloads of the messages directly to the Redis cluster. These payloads can only be received by one Redis node. However, this node may not be connected with the clients that subscribe to these topics. Therefore, this Redis node would have to share the payloads with all the other nodes. Other nodes connected with the correct subscribers can send the payloads to those subscribers. In this manner, a satisfactory performance of the message broker can be ensured.

To further guarantee database reliability, advanced techniques such as hot standby and transaction logging are needed for the Redis cluster. Each Redis node has its backup, which runs in a standby server in the event of malfunction. Furthermore, transaction logging can record the history of the database, including hostile attacks for the convenience of recovery.

Load Balancers: In order to support a large number of requests and messages from massive IoT devices and users, a load balancer is necessary for the IoT cloud. HAProxy[3] is a proper option. As an open-source lightweight load balancer, HAProxy can offer efficient TCP-based and HTTP-based load balancing for the IoT cloud and runs in an event-driven, single-thread model supporting high concurrency. Through proper configurations, the following two types of load balancing are deployed for the IoT cloud with HAProxy.

HTTP Load Balancers: The default load balancing method, i.e., weighted round robin, is configured in HAProxy to distribute requests for the HTTP servers. HAProxy can check the uniform resource locator (URL) of a request on a binding port, and then distribute the request to the object HTTP server according to the URL or some predefined rules. In this way, the workload of the HTTP servers can be efficiently balanced.

MQTT Load Balancers: There are no functions of load balancing for the MQTT servers present in HAProxy. As a result, TCP load balancing is deployed to distribute loads for multiple MQTT servers. TCP load balancing can transfer TCP packages to the matching backend server in accordance with a set of predefined rules. This is realized by the Network Address Translation (NAT) protocol and some load balancing methods. According to the NAT protocol, HAProxy provides a virtual IP and port for outer networks to visit, and listen to a certain port for TCP packages. When it receives any TCP package, HAProxy can change the destination IP address and the port number of the package before forwarding the concerned package to the target backend MQTT server. Different from HTTP servers, the MQTT servers need to keep long-lived connections with clients, and thus HAProxy needs to maintain these connections. Therefore, the least connections policy is chosen as the load balancing method for the MQTT servers in HAProxy. It helps HAProxy select the MQTT server with the least number of active connections as the target server.

Services of the IoT Cloud

The IoT cloud should meet the requirements of different IoT applications. By using the services provided by the IoT cloud, diverse applications can be offered to both end-users, developers, or managers. These services can be accessed by web browsers or smartphones anytime and anywhere. IoT cloud services can be mainly divided into the following three categories.

Web Applications: The IoT cloud can provide services through web applications by deploying web pages in HTTP servers. These HTML pages are developed using the Hyper Text Markup Language (HTML) and cascading style sheets (CSS) for static web pages, as well as JavaScript that defines the actions a page should take toward different events. Web applications in the IoT cloud are mainly developed for managers to supervise the devices they own. By managing the IoT cloud interface, managers can monitor the detailed information of the devices, e.g., the on-off state, the media access control (MAC) address, or the timing tasks. They can also control devices directly with the permission of device owners for convenient debug.

Mobile Applications: The smartphone is becoming an indispensable communications tool in daily life. With the pervasiveness of smartphones, people browse the Internet on their phone using a large variety of smartphone application programs (APPs). Android and iOS are the two dominant OSs for smartphones. For the convenience of end-users, mobile APPs based upon the IoT cloud are developed to meet the requirements of both the Android and iOS platforms. On the other hand, there are some APPs that are developed by third parties (such as Facebook or WeChat) that are very popular. In order to provide IoT cloud services to end-users, several interfaces based on these APPs are defined to allow users to access the corresponding services. For instance, end-users can control their smart-home devices using APPs that are designed by manufacturers or by WeChat directly.

Software Development Kits: In order to enhance its applications and services, the IoT cloud provides two software development kits (SDKs) to third party developers, i.e., Android and

[3] https://www.haproxy.org/

iOS SDKs. The SDKs consist of several application programming interfaces (APIs) by which the complex functions of the IoT cloud can be easily used. Every API encapsulates a number of underlying operations, and provides easy access for developers. The SDKs make it easier to develop APPs so as to unleash the full potential of the IoT cloud.

Experimental Results and Analysis

In this Section, our experiments are carried out under various conditions in order to evaluate the performance of the proposed IoT cloud. We focus only on the performance of MQTT and HTTP servers because both of them have a significant impact on the perceived quality of the services of the IoT cloud under consideration. Because of the different mechanisms of MQTT and HTTP servers, different metrics are used to evaluate their respective performance.

Experimental Configuration

The MQTT and HTTP servers are deployed on two separate VMs. The number of CPU cores used by either the MQTT or HTTP server can be configured from one to four. For testing purposes, we use Node.js on another VM to simulate the interactions between the clients and servers. Furthermore, a Redis cluster with eight nodes is implemented on four VMs, acting as both the database and broker.

Results

HTTP Server: To measure the performance of the HTTP server, a certain number of clients is generated, each of which establishes a connection with the HTTP server. The number of clients can be between 1,000 to 20,000 as deemed appropriate. All the clients send only the GET requests to the servers to fetch a web page, whose size is about 1K bytes. If the HTTP server handles the request successfully, it sends the web page back to the client. Once the client receives the response, it continues to send a same new request. Otherwise, if no response is received by the client within ten seconds, the request is regarded as a failure. During a total of 180 seconds, the number of requests that are successfully handled by the HTTP servers is counted so as to compute the throughput performance of the HTTP servers. Meanwhile, the response time of each request is also collected and averaged.

Figure 3a shows that the response time becomes largely linear with the number of clients. Since the clients continuously send requests, the HTTP server runs at its full capacity all the time. Thus, throughput and CPU utilization remain at high levels, as can be seen in Fig. 3b and Fig. 3c. However, many requests cannot be handled on time and are discarded when the number of clients is larger than 10,000 or 20,000 for the HTTP server with one or two CPU cores, respectively. Correspondingly, the throughput performance of the HTTP server with one or two CPU cores rapidly deteriorates when the number of clients is above 10,000 or 20,000, respectively. In addition, due to the asynchronism of the HTTP server, it can still handle more requests with full CPU utilization, as shown in Fig. 3c.

MQTT Server: We consider 2,000 to 40,000 publishers when evaluating the performance of the MQTT servers. Each publisher subscribes an individual topic and publishes messages that contain only the timestamp to that topic every 10 seconds. During the period of the experiment, i.e., 120 seconds, we collect the information on the total number of messages that have been successfully delivered to the publishers as well as the transmission latency of the messages.

As shown in Fig. 4, one MQTT server can pro-

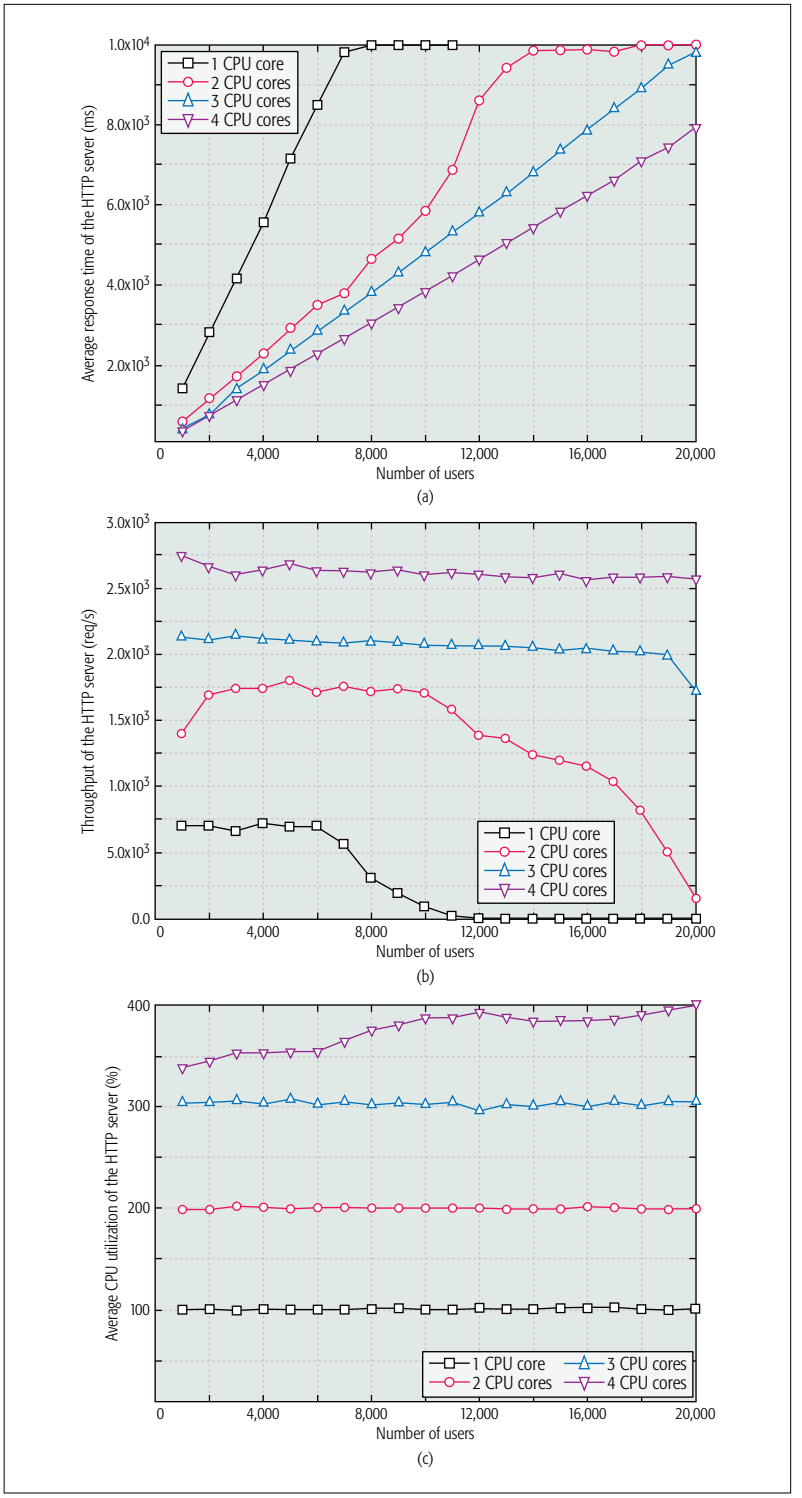

Figure 3. Performance of the HTTP server: a) Average response time of the HTTP server; b) throughput of the HTTP server; c) average CPU utilization of the HTTP server.

vide services up to 40,000 clients with four CPU cores in ten-second intervals of publishing at most. The average transmission latency becomes larger with the increase of the number of users, and lower when the number of CPU cores increases. The throughput of the MQTT server becomes floored when the number of clients becomes larger than some predefined threshold, depending on the number of CPU cores, i.e., 10,000, 22,000, 34,000, and 40,000 for one, two, three, and four cores, respectively. It is evident that an excessive number of packages render the MQTT server unable to transfer packages in time. A similar trend for CPU utilization can be observed in Fig. 4c.

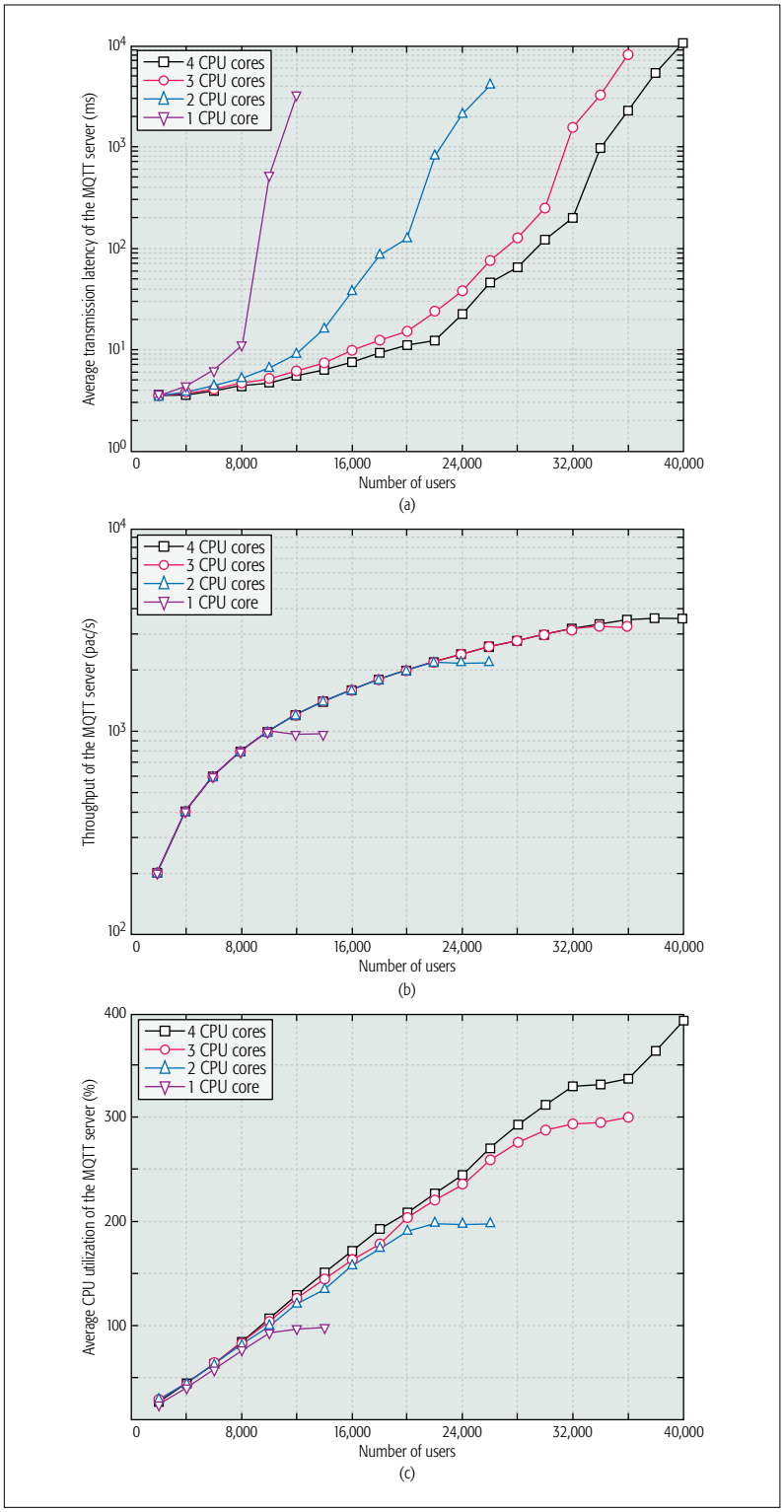

Figure 4. Performance of the MQTT server: a) average transmission latency of the MQTT server; b) throughput of the MQTT server; c) average CPU utilization of the MQTT server.

Conclusion and Outlook

IoT is a new technological paradigm enabling ubiquitous things or objects to interact with each other and to access the Internet. By integrating cloud computing and IoT techniques, new valuable and reliable services can be provided to many users. This article mainly proposed an IoT cloud architecture and its corresponding implementation. The key point of the architecture is the combination of the HTTP and MQTT servers, as well as the implementation of the message broker. Several experiments were conducted with the objective of evaluating the performance of the application servers in the proposed IoT cloud. The performance results demonstrated the significant impact of the number of clients and CPU cores on the average transmission latency/response time, throughput, and CPU utilization for the HTTP and MQTT servers, respectively. They also study and discuss the performance of the IoT cloud that was implemented.

Much more research is still needed to address many other challenges of the IoT cloud in the near future. First, there is a lack of well-defined standards to unify varying architectures and interfaces of the IoT cloud. Moreover, advanced data analytics are necessary to make full use of big data that IoT brings about. Finally, security and privacy challenges should also be taken into account in designing the IoT cloud.

References

[1] E. Dave, "White Paper: The Internet of Things, How the Next Evolution of the Internet is Changing Everything," Cisco Internet Business Solutions Group (IBSG). Accessed on: Apr. 2011, http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[2] K. Zheng *et al.*, "Big Data Driven Optimization for Mobile Networks Towards 5G," *IEEE Network*, vol. 30, no. 1, Jan. 2016, pp. 44–51.

[3] Y. Zhang, S. He, and J. Chen, "Data Gathering Optimization by Dynamic Sensing and Routing in Rechargeable Sensor Networks," *IEEE/ACM Trans. Net.*, no. 99, June 2015, pp. 1–15.

[4] N. Kumar *et al.*, "Performance Analysis of Bayesian Coalition Game-Based Energy-Aware Virtual Machine Migration in Vehicular Mobile Cloud," *IEEE Network*, vol. 29, no. 2, Apr. 2015, pp. 62–69.

[5] J. Swetina *et al.*, "Toward a Standardized Common M2M Service Layer Platform: Introduction to oneM2M," *IEEE Wireless Commun.*, vol. 21, no. 3, June 2016, pp. 20–26.

[6] T. Taleb and A. Kunz, "Machine Type Communications in 3GPP Networks: Potential, Challenges, and Solutions," *IEEE Commun. Mag.*, vol. 50, no. 3, Mar. 2012, pp. 178–84.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Comput. Net.*, vol. 54, no. 15, Oct. 2010, pp. 2787–805.

[8] A. Al-Fuqaha *et al.*, "Internet of Things: A Survey on Enabling Technologies, Protocols and Applications," *IEEE Commun. Surveys Tut.*, vol. 17, no. 4, Nov. 2015, pp. 2347–76.

[9] H. Zhang *et al.*, "Optimal DoS Attack Scheduling in Wireless Networked Control System," *IEEE Trans. Control Syst. Tech.*, vol. 24, no. 3, Aug. 2015, pp. 843–52.

[10] L. Lei *et al.*, "Delay-Optimal Dynamic Mode Selection and Resource Allocation in Device-to-Device Communications — Part II: Practical Algorithm," *IEEE Trans. Vehic. Tech.*, vol. 65, no. 5, June 2015, pp. 3491–505.

[11] K. Zheng *et al.*, "Soft-Defined Heterogeneous Vehicular

Network: Architecture and Challenges," *IEEE Network*, vol. 30, no. 4, July 2016, pp. 72–80.
[12] J. L. Perez and D. Carrera, "Performance Charaterization of the servIoTicy API: An IoT-as-a-Service Data Management Platform," *Proc. IEEE Int'l. Conf. Big Data Comput. Service and Appl.*, Redwood City, Calif., Mar. 2015, pp. 62–71.
[13] A. Zanella *et al.*, "Internet of Things for Smart Cities," *IEEE Internet Things J.*, vol. 1, no. 1, Feb. 2014, pp. 22–32.
[14] V. Karagiannis *et al.*, "A Survey on Application Layer Protocols for the Internet of Things," *Trans. IoT and Cloud Comput.*, vol. 3, no. 1, Jan. 2015, pp. 11–17.
[15] M. Collina, G.E. Corazza, and A. Vanelli-Coralli, "Introducing the QEST Broker: Scaling the IoT by Bridging MQTT and REST," *Proc. IEEE Int. Symp. Personal Indoor and Mobile Radio Commun.*, Sydney, Australia, Sept. 2012, pp. 36–41.

Biographies

Lu Hou received his B.S. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications (BUPT), China, in 2014. He is now a Ph.D. candidate in the Intelligent Computing and Communication (IC^2) Lab, Key Lab of Universal Wireless Communications, Ministry of Education, BUPT. His research interests are mainly focused on resource allocation and security in mobile cloud computing.

Shaohang Zhao received his B.S. degree from Beijing University of Posts and Telecommunications (BUPT), China, in 2014. He is now studying for a master's degree at the Intelligent Computing and Communication (IC^2) lab. His research mainly concentrates on Internet of Things networks.

Xiong Xiong received his B.S. degree from Beijing University of Posts and Telecommunications (BUPT), China, in 2013. Since then he has been working toward a Ph.D. degree at BUPT. His research interests include M2M networks and software defined radio.

Kan Zheng [S'02, M'06, SM'09] (zkan@bupt.edu.cn) is a professor at Beijing University of Posts and Telecommunications (BUPT), China. His current research interests are in the field of wireless communications, with an emphasis on performance analysis and optimization of heterogeneous networks and 5G networks. He has published more than 200 papers in IEEE conferences and transactions.

Periklis Chazimisios [S'02, M'05, SM'12] is an associate professor with the Department of Informatics at the Alexander TEI of Thessaloniki (ATEITHE), and he is the director of the Computing Systems, Security and Networks (CSSN) Research Lab. He is an author/editor of eight books and more than 100 peer-reviewed papers on performance evaluation and standardization of mobile/wireless communications, Internet of Things, and big data. He received his Ph.D. from Bournemouth University, UK in 2005, and his B.Sc. from ATEITHE, Greece in 2000.

M. Shamim Hossain [SM'09] is an associate professor at King Saud University, Riyadh, KSA. Dr. Hossain received his Ph.D. in electrical and computer engineering from the University of Ottawa, Canada. His research interests include serious games, cloud and multimedia for healthcare, resource provisioning for big data processing on media clouds, and biologically inspired approaches to multimedia and software systems. Dr. Shamim is a senior member of IEEE, and a member of ACM and ACM SIGMM.

Wei Xiang [S'00, M'04, SM'10] received the B.Eng. and M.Eng. degrees, both in electronic engineering, from the University of Electronic Science and Technology of China, Chengdu, China, in 1997 and 2000, respectively, and the Ph.D. degree in telecommunications engineering from the University of South Australia, Adelaide, Australia, in 2004. He is currently foundation professor and head of Discipline Electronic Systems and Internet of Things Engineering in the College of Science and Engineering at James Cook University, Cairns, Australia.