计算机科学
COMPUTER SCIENCE

CARINA:一种高效的解决IoT互操作性的应用层协议转换方案

王丽娜, 赖坤豪, 杨康

引用本文

王丽娜, 赖坤豪, 杨康. CARINA:一种高效的解决IoT互操作性的应用层协议转换方案[J]. 计算机科学, 2024, 51(2): 278-285.

WANG Lina, LAI Kunhao, YANG Kang. CARINA:An Efficient Application Layer Protocol Conversion Approach for IoT Interoperability [J]. Computer Science, 2024, 51(2): 278-285.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230100108

CARINA:一种高效的解决 IoT 互操作性的应用层协议转换方案

王丽娜　赖坤豪　杨　康
武汉大学国家网络安全学院空天信息安全与可信计算教育部重点实验室　武汉 430000

摘　要　为了解决物联网设备众多、协议众多,以及协议架构和应用场景不同引发的物联网设备互操作性问题,针对应用层使用广泛的 HTTP 等 4 种协议,提出了一种基于协议包解析和关键方法映射的高效可扩展的应用层协议转换方案。考虑到 4 种协议的基础架构、消息格式、通信模式以及应用场景具有较大差异,该方案通过对协议原始数据包进行解析和关键信息提取,然后统一以键值对的形式进行信息存储,解决了不同协议信息存储的统一性问题。通过构造关键方法映射表,将不同协议的方法进行映射,实现了不同协议之间的互联。实验结果表明,基于所提方案实现的协议转换系统能很好地完成 4 种协议之间的消息转换。相比同类型的 Ponte 方法,在相同实验条件下,所提方案的转换速度都优于 Ponte,甚至在某些情况下表现出了将近 10 倍的速度差距,同时支持多出一倍的转换类型。实验结果验证了所提方法在可扩展性和转换时间等效率方面相比同类型的协议转换算法具有显著提升。

关键词:物联网;应用层协议;协议转换;互操作性
中图分类号　TP393

CARINA:An Efficient Application Layer Protocol Conversion Approach for IoT Interoperability

WANG Lina,LAI Kunhao and YANG Kang
Key Laboratory of Aerospace Information Security and Trusted Computing,Ministry of Education,School of Cyber Science and Engineering,Wuhan University,Wuhan 430000,China

Abstract　To solve the interoperability problems caused by numerous IoT devices and protocols with varying architectures and application scenarios,this paper proposes an efficient and scalable application layer protocol conversion approach. This approach uses protocol packet parsing and key method mapping for widely used HTTP and other three protocols. Considering the significant differences in the underlying architecture,message format,communication mode,and application scenario of the four protocols,the proposed approach solves the uniformity of information storage for different protocols by parsing the original data packets of the protocols and extracting key information,and storing the information in the form of key-value pairs. By constructing the key method mapping table,the methods of different protocols are mapped,realizing the interconnection between different protocols. Experimental results show that the proposed approach performs well in message conversion between the four protocols. It demonstrates a significantly improved conversion speed compared to the Ponte method of a comparable type,with a nearly 10-fold difference observed in some cases when subjected to the same test conditions. Furthermore,it supports twice as many conversion types as Ponte. Experimental results show that the proposed method outperforms state-of-the-art methods in terms of scalability and efficiency.

Keywords　Internet of things,Application layer protocol,Protocol conversion,Interoperability

1　引言

随着越来越多的物理设备以前所未有的速度连接到互联网,物联网(Internet of Things,IoT)[1]的理念逐渐变为现实。现代物联网将互联网的基础设施作为信息传递的载体,实现了各种各样的物联网场景。大多数物联网设备由于资源受限、通信可靠性低,需要有效的通信协议来提高效率,而设备和场景的多样化导致了通信协议的多样化。在应用层,就有很多不同的协议来支持不同的应用场景。例如,HTTP 基于传统的 Web 服务,适用于大多数的应用场景;消息队列遥测传输协议(Message Queuing Telemetry Transport,MQTT)被设计适用于网络代价昂贵、带宽低、不可靠的环境;受限应用协议(Constrained Application Protocol,CoAP)被设计用于内存空间和计算能力有限的资源受限设备。此外,还有高级消息队列协议(Advanced Message Queuing Protocol,AMQP)、数据分发服务(Data Distribution Service,DDS)等适用于其他不同场景的协议。

然而,由于庞大的设备数量和众多的通信协议,物联网的

到稿日期:2023-01-19　返修日期:2023-05-29
基金项目:国家自然科学基金(61876134);国家科技支撑计划(2020YFB1805400)
This work was supported by the National Natural Science Foundation of China(61876134) and National Key Technology Research and Development Program of the Ministry of Science and Technology of China(2020YFB1805400).
通信作者:王丽娜(lnwang@whu.edu.cn)

环境非常复杂,不同的设备可能无法相互通信,出现了互操作性问题。物联网互操作性的缺乏,意味着数据无法在不同的设备和系统之间进行有效交换,而数据在不同设备和系统之间进行有效交换是大规模部署物联网的基石。在现有的物联网生态环境中,物联网设备的互操作性对于构建可扩展、可适应和无缝物联网设备网络变得越来越重要[2]。

为了解决互操作性问题,一对一和多对多的协议转换方法相继被提出。文献[3]提出了从 HTTP 到 CoAP 的转换,使得基于 CoAP 协议的设备可以接收来自 HTTP 协议设备的消息。文献[4]研究了将 CoAP,MQTT,DDS 和 Websockets 消息转换为HTTP 消息的方法。然而,一对一的方法很难扩展新的协议,随着支持协议数量的增多,实现难度呈几何倍数的上升[2,5-6]。此外,尽管存在多对多的协议转换方法,但这些方法存在效率低的问题,不能尽可能快地给予响应[4,7-8]。虽然先前的工作覆盖了大部分常见的物联网应用层协议,但 AMQP 消息队列协议几乎没有被提到过。由于消息队列协议几乎是必不可少的一个应用层协议,因此研究 AMQP 协议与其他协议的交互也至关重要。

综上所述,本文选取了物联网最常见的 HTTP,CoAP 和 MQTT 协议,以及未被提到过的 AMQP 协议,从可扩展性和效率出发,提出了一种高效的、可扩展的协议转换方案 CARINA,以解决 HTTP,CoAP,MQTT 和 AMQP 的互操作性问题。具体地,CARINA 通过对不同数据的原始数据包进行解析并以键值对的形式存储相关信息,根据不同协议关键方法的映射关系进行一一对应,封装目标协议的回复包,来达到互操作性的目的。实验验证了该方案能达到较优的转换效果,相比同类型的 Ponte 方法,其转换速度在最好的情况下能快 10 倍以上,最坏情况下也能快 12%左右,且支持多出一倍的转换类型。此外,通过探究数据量对转换时间的影响,证明了随着数据量的增大,所提方案所需转换时间并不会有很大变化。例如,在数据量增加 100 倍的情况下,转换时间也只是增加几微秒。

具体来说,本文的贡献如下:

1)实现了一个高效的、可扩展的协议转换方案 CARINA,解决了 HTTP,CoAP,MQTT 和 AMQP 这 4 种应用层协议之间的互操作性问题。

2)CARINA 具有很好的扩展性和效率。总结了同属于请求响应的 HTTP 协议和 CoAP 协议的共性,使得相同通信模式新协议的添加只需要按照规定的接口实现就可以与已有的协议进行交互。同样地,从底层数据包出发对不同协议进行解析和封装,达到了较高的效率。

3)所提方案的转换速度优于同类型的方案 Ponte,甚至某些情况下可以快 10 倍左右,并且转换时间基本不受数据量大小的影响。

2 背景知识和相关工作

本章系统性地描述和分析了本文所使用的 IoT 应用层协议和物联网应用层协议的转换方法。

2.1 应用层协议

物联网网络使用各种各样的应用层协议,每种协议都有不同的特点。本小节介绍物联网网络中使用的主要应用协议的特点和格式。由于 HTTP 协议的使用范围较广,大部分开发者对其比较熟悉,因此这里主要介绍 CoAP,MQTT 和 AMQP 这 3 种协议。

1)CoAP:一种请求回复模式的协议,由 IETF 标准化。CoAP 旨在使功耗低、计算和通信能力弱的微型设备能够利用 REST 格式进行交互。

CoAP 在 HTTP 功能之上定义了一个基于 REST 的 Web 传输协议,其客户端与服务器之间的连接是无状态的。CoAP 与 HTTP 一样,利用 GET,PUT,POST 和 DELETE 等方法来实现创建、检索、更新和删除操作。由于 CoAP 是基于 REST 设计的,因此在 REST-CoAP 代理中这两种协议之间的转换很简单[1]。

CoAP 使用简单且小型的格式对消息进行编码。CoAP 数据包的消息格式如图 1 所示[9]。第一行是 4 个固定字节的消息头(Header)。消息头(Header)中的字段如下:Ver 是 CoAP 的版本,T 是事务的类型,OC 是 Option count,Code 代表请求方法(1-10)或响应码(40-255)。例如,GET,POST,PUT 和 DELETE 的代码分别为 1,2,3 和 4。标头中的 Message ID 是用于匹配响应的唯一标识符。在一个字节的分隔符之后,是负载(Payload)。我们主要关注的是代码(Code)和负载(Payload)字段。同时,CoAP 使用 4 种类型的消息,即可确认、不可确认、重置和确认,其可靠性是通过混合可确认和不可确认的消息来实现的。

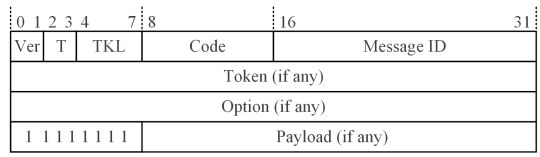

图 1 CoAP 的消息格式

Fig. 1 Format of CoAP message

2)MQTT:一个发布订阅模式的协议,在 2013 年由 OASIS 标准化。MQTT 旨在将嵌入式设备和网络与应用程序和中间件连接起来,适用于使用不可靠或低带宽链路的资源受限设备。

MQTT 大致由 3 个组件组成:订阅者(Subscriber)、发布者(Publisher)和代理(Broker)。可以将设备注册为特定主题的订阅者,以便在发布者发布该主题时由代理通知它。发布者充当对应数据的生成者。之后,发布者通过代理将信息传输给订阅者[10]。MQTT 协议的整体功能图如图 2 所示。

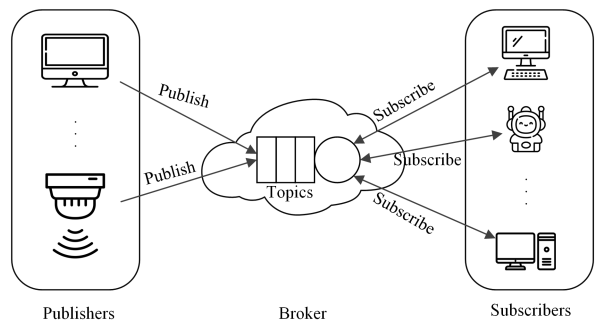

图 2 MQTT 协议的通信模型

Fig. 2 Communication model of MQTT

图3给出了MQTT协议[11]使用的消息格式。在MQTT协议中,一个MQTT数据包由固定头(Fixed Header)、可变头(Variable Header)和消息体(Payload)3部分构成。固定头的第一个字段为消息类型,表示CONECT(1),CONNACK(2),PUBLISH(3),SUBSCRIBE(8)等多种消息。可变头存在于部分MQTT数据包中,数据包类型决定了可变头是否存在及其具体内容。消息体(Payload)字段也存在于部分MQTT数据包中,表示客户端收到的具体内容。

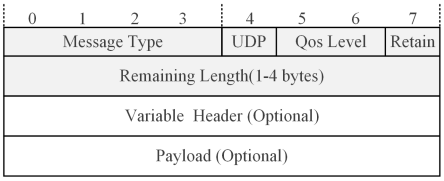

图3 MQTT的消息格式

Fig. 3 MQTT message format

3)AMQP:一种消息队列模式的消息,是物联网的开放标准应用层协议,专注于面向消息的环境[12]。基于此协议的客户端与消息中间件可传递消息,并不受产品、开发语言等条件的限制。

AMQP的协议模型如图4所示,大体上是由3个组件构成:发布者(Publisher),消费者(Consumer),代理(Broker)。代理中包含两个主要组件:交换机(Exchange)和消息队列(Queue)。发布者发布消息,经由交换机;交换机根据路由规则将收到的消息分发给与该交换机绑定的队列;最后AMQP代理会将消息投递给订阅了此队列的消费者,或者消费者按照需求自行获取。

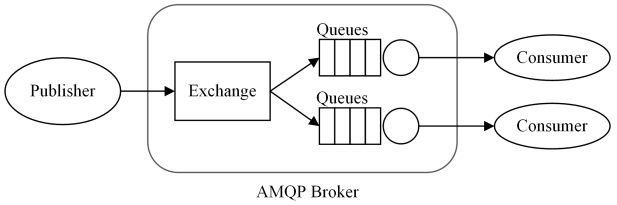

图4 AMQP协议的通信模型

Fig. 4 Communication model of AMQP

AMQP拥有众多的适用于各种流行语言和框架的客户端。其中一部分严格遵循AMQP规范,提供AMQP方法的实现。另一部分提供了额外的技术,方便使用的方法和抽象实现。其中较为知名的是RabbitMQ[13]。本文使用了RabbitMQ的客户端与其他协议进行交互。

2.2 相关工作

已有工作可以被分为两类:一对一的协议转换和多对多的协议转换。一对一的协议转换就是在特定的两个协议之间进行转换。文献[6]提出了一种MQTT和XMPP的协议转换方法。文献[14]提出了一种MQTT和REST之间的转换方法。文献[15]进行了HTTP与CoAP的协议转换代理的研究。文献[16]提出了一种HTTP2和CoAP之间的转换方法。文献[17]提出了一个将MQTT消息转换为HTTP消息的应用层网关。上述工作都是特定两种协议之间的转换。一对一的转换局限性较大,只适用于特定的两种协议。如果方法的扩展性不好,对新协议的支持可能需要次方级别的工作量。因此,需要一种可扩展性好、支持更多协议的转换方法来解决互操作性问题。

多对多的协议转换连接了两种以上的协议,包括单向的转换和双向的转换。多对多的转换中最为重要的尝试是文献[7]提出的Ponte,其解决了MQTT,CoAP和HTTP之间的互操作性问题。它提供了统一的开放接口,供开发者实现三者之间的自动转换。Ponte是在Eclipse物联网项目[18]下开发的,同时包含其他子项目用于帮助消费者简化IoT解决方案。但其使用了中间件,导致转换效率受到影响;并且由于是联合其他子项目的共同方案,需要复杂的环境配置。文献[8]提出了一种MQTT,CoAP,XMPP和SMTP互相转换的方法。文献[19]提出了一个初步的网关体系结构——GoThings,其支持不同协议之间的互联。Mauro等将他们先前的工作扩展到了4个协议,提出了名为MiddleBridge的应用层网关[4],其可以将CoAP,MQTT,DDS和Websockets消息转换为HTTP消息。文献[20]提出了IoT-CPCS,将MQTT,CoAP和Modbus-TCP协议包提取出统一的JSON格式的关键信息,并存储在云平台构建的虚拟服务器中。这些方法各有特点,且适用于不同的场景。这些方案的部署场景相对简单,且大部分只能部署在网关处,不能部署在资源有限的设备端。

考虑到上述工作的局限性,本文提出的方案也是支持多对多的协议转换,在3个较常用的协议(HTTP,CoAP和MQTT)之上增加了之前没有被提及的AMQP协议。相比Ponte中间件的使用,CARINA直接对原始协议数据包进行解析,不仅节省了中转时间,提高了转换效率,更让其可以部署在资源受限的物联网设备端。与此同时,HTTP和CoAP都属于请求响应模式的通信协议,本文总结了该类通信模式的共性,使得使用者能够容易地对相同模式的协议进行扩展。

3 框架设计

本文提出了CARINA——一个应用层协议转换方案,通过对协议原始数据包进行解析和关键信息提取,依据不同协议关键方法的映射表,将不同协议的方法进行映射,封装目标协议的响应包,实现了HTTP,CoAP,MQTT和AMQP 4种协议的通信。

图5给出了CARINA的基本框架。一个单个的节点包含6个组件:4个服务端(Servers),1个数据处理单元(Data Processing Unit),1个数据存储单元(Data Storage Unit) Redis[21]。各组件主要的功能如下。

1)服务端(Servers):用来监听4种协议的消息,对原始数据包进行解析并对回复包进行封装。

2)数据处理单元(Data Processing Unit):对解析获得的消息进行不同协议的业务处理,并对不同协议的关键方法和关键字进行映射关联。

3)数据存储单元(Data Storage Unit):用键值对的形式存储资源的最近更新值。Redis作为一个基于内存的数据库,有专门的方式存储键值对信息,同时利用Redis的发布订阅

功能能很好地实现了资源更新的通知。因此,选用 Redis 作为数据存储单元的具体实现。

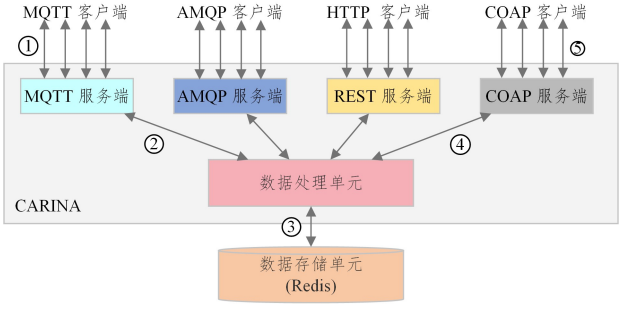

图 5 CARINA 的基本框架

Fig. 5 Basic framework of CARINA

接下来介绍这些组件之间的相互作用。CARINA 充当一个多协议代理,其主要职责是接收并响应客户端的更新与查询,通知 MQTT 订阅者特定主题的更新,通知 AMQP 订阅者特定消息队列的更新。完整的工作流程大致可以描述如下:

1)4 个协议的服务端对接收到的客户端数据包进行解析和信息提取,获取到资源处理方法(如 HTTP 的 PUT 方法)、要更新的资源和要更新的新值。

2)数据处理单元根据解析提取到的消息进行业务处理,例如给某种资源添加订阅,对某个资源值进行更新。最主要的是,对不同协议的方法进行映射关联,使得资源可以在不同的协议之间进行互操作。

3)将资源的最新值存储在 Redis 中进行记录,并发布特定 Redis 键的更新给数据处理单元。

4)数据处理单元根据 Redis 键的更新和不同协议方法的映射关系,向对应的服务器提供数据包封装所需要的信息(处理方法,以及资源和资源值)。

5)服务器按照协议对应的格式对数据包进行封装,然后发送。

特别地,考虑到不同协议的特性和共性,本文按照协议背后的通信模式来统一进行处理。对于请求响应模式的协议 HTTP 和 CoAP 来说,只需要进行资源的更新和获取,获取资源时直接从 CARINA 的数据存储单元处返回对应的资源值即可,所以从其他模式到该种模式的响应是比较迅速的。对于发布订阅模式的协议 MQTT,需要提前对资源进行订阅,当资源进行更新时,Redis 也会发布对应的更新,相对应的服务端就会向订阅客户端发送对应的订阅资源更新消息。

与其他 3 种协议相比,消息队列模式的协议 AMQP 更为复杂,数据包种类更多,因此 CARINA 使用中间件 RabbitMQ 来代替数据包的解析和封装。CARINA 建立一个超级客户端来监听消息队列的变化,及时获取到消息队列的变化,然后向其他协议的客户端传递消息,进行数据包的封装和发送。图 6 详细描述了上述过程。

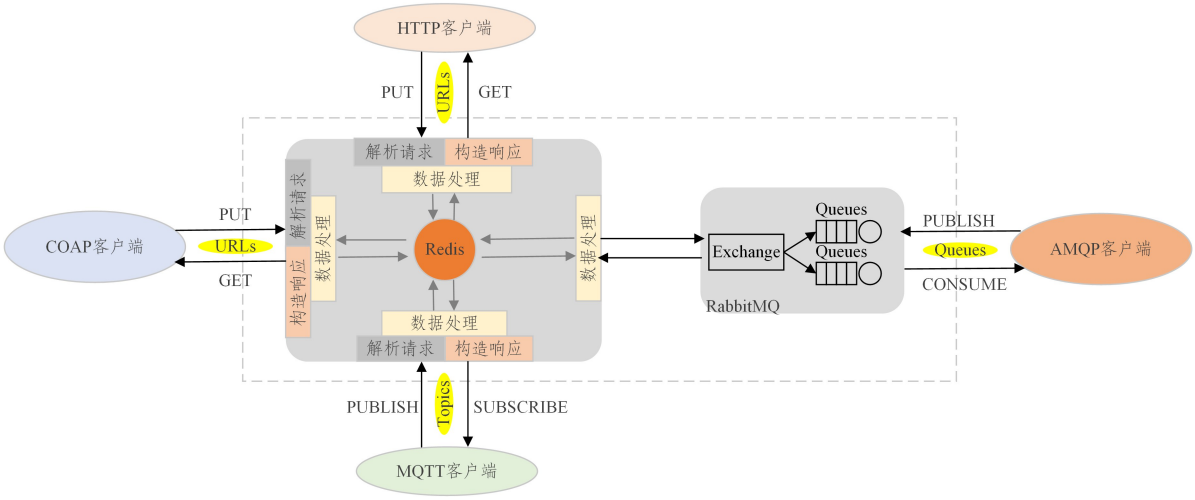

图 6 CARINA 的交互过程

Fig. 6 Interaction process of CARINA

4 实现

不同协议能够相互通信的关键,是一种协议的数据包能转换成另一种协议的数据包。但是不同的协议使用不同的方法去获取和更新资源,因此最大的挑战就是如何确保不同协议的设备一致性地操作同一种资源。为了实现从一种协议到另一种协议的通信,我们将两种协议的资源名进行匹配。当一种协议的设备更新资源时,CARINA 会及时通知另一种协议的设备。表 1 列出了 4 种协议的资源表示方法,以及获取和更新资源的方式。

表 1 不同协议的资源名和操作方法名

Table 1 Resource names and operation methods of different protocols

协议	资源名	更新资源	获取资源
HTTP	URLs	PUT	GET
CoAP	URLs	PUT	GET
MQTT	Topic	PUBLISH	SUBSCRIBE
AMQP	Queue	PUBLISH	CONSUME

HTTP 和 CoAP 背后的通信模式都是请求通信模式,且它们都是基于 REST 风格的协议,使用 URL 来表示资源,都使用 GET 和 PUT 方法去获取和更新资源。区别在于,CoAP

协议的数据包更小，更适用于资源受限的设备。因此统一将HTTP协议和CoAP表示为REST协议，将CoAP资源和HTTP资源都称为REST资源。

MQTT协议是发布订阅模式的协议，设备需要提前订阅资源，当资源发生变化时，代理会通知客户端该资源的值发生了变化。

AMQP是一个消息队列协议，包含发布者和消费者。AMQP协议的资源存储在各个队列中，通过交换机转发，因此AMQP协议使用队列名来表示和定位资源，队列里面的内容表示资源的值。客户端会持续监听感兴趣的队列并等待消息的到达，当发布者向队列中发布消息时，客户端就会收到并处理消息。

4.1 REST协议和MQTT协议的转换

REST协议和MQTT协议之间的转换主要是REST资源和MQTT资源的互相表示。这两种协议进行相互转换的前提是REST的URL和MQTT的Topic一致，表示对同一种资源的修改。

将REST资源表示为MQTT资源，也就是让MQTT类型的设备能获取到REST类型设备的变化。在REST类型的设备利用PUT或者POST修改一个新值后，CARINA会将新值存储起来，然后封装MQTT协议对应的PUBLISH包，向订阅这个主题的设备都发送对该资源的修改。例如，如果资源的名字是*light_brightness*，表示灯的亮度。该主题事先被一些MQTT设备所订阅，当HTTP客户端或CoAP客户端使用PUT请求修改对应的URL：*/light_brightness*为85时，所有订阅该主题的设备都会收到灯的亮度变为85的消息。

将MQTT资源表示为REST资源，就是让REST类型的设备能获取到MQTT设备发布的值。由于REST协议通过GET请求获取资源是客户端主动的行为，因此CARINA会将MQTT客户端发布的最新值存储起来，在REST客户端通过GET数据包请求时返回最新值。

4.2 REST协议和AMQP协议的转换

对于REST和AMQP的交互，将REST协议中的URL与AMQP协议中的队列名对应，具体的细节如下。

将REST资源表示为AMQP资源时，即表示使得AMQP的设备能访问到REST设备的修改。这里需要AMQP客户端提前对某个队列名进行监听，当REST客户端发出修改资源的PUT请求时，CARINA获取了请求的内容之后会构造AMQP发布(PUBLISH)包，向AMQP对应的队列发布消息，这样监听在该队列的客户端就能获取到相应的变化，进而消耗处理资源。

将AMQP资源表示为REST资源，与MQTT到REST的转换类似。AMQP协议发布资源的方式是发布者向队列中发布资源；CARINA内部需要模拟AMQP的消耗者消耗掉发布者发布的资源，然后将最新值存放到CARINA的数据存储处。这样，当REST客户端发送访问资源的GET请求时，只需要根据对应的URL返回最新的资源就可以了。通过这种方式，即可达到通过REST请求资源的方式获取到AMQP资源的目的，实现从REST到AMQP方向的通信。

4.3 MQTT协议和AMQP协议的转换

MQTT协议和AMQP协议转换的关键就是将MQTT定位资源的主题(Topic)与AMQP定位资源的队列名(Queue)对应。发布订阅模式和消息队列模式比较类似，更新资源的方式都是通过发布包，MQTT通过订阅包的提前订阅来获取更新，AMQP通过消耗包提前监听队列来获取更新。下面描述两者之间的交互细节。

将MQTT资源表示为AMQP资源，也就是说，可以通过AMQP协议的方式获取到MQTT协议的资源。AMQP的消耗者一直在监听想要监听的队列；MQTT的发布者发布资源时，只要将资源放置到AMQP的队列上就可以达到上述目的。因此，CARINA在接收到MQTT客户端发布的消息之后，以AMQP发布资源的方式在AMQP队列上发布消息，对应的AMQP消耗者就可以接收到消息。

将AMQP资源表示为MQTT资源是通过MQTT协议的方式获取AMQP发布的更新。CARINA会在内部检测AMQP队列的变化。当AMQP客户端向队列中发布消息后，方案会模拟AMQP协议消耗资源的方式消耗掉该资源，并将最新值存储起来且同步推送给订阅该主题的MQTT客户端，即可通过MQTT协议的方式获取AMQP发布的更新。

5 实验评估

5.1 环境条件

本文选取了不同协议的标准客户端进行测试，图7给出了参与测试的客户端类型。由于客户端运行条件的限制，本文最终在表2所列环境的计算机上运行了基于所提方案实现的协议转换系统，同时用第5.2节的测试场景评估了所提方案CARINA在不同协议之间的互操作性。

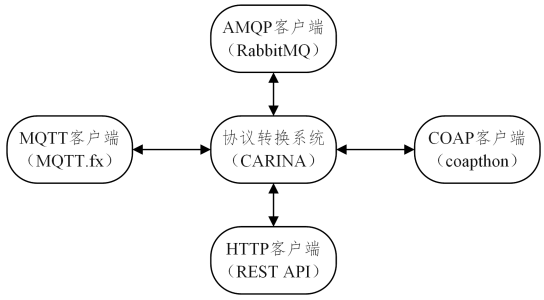

图7 测试客户端的类型

Fig. 7 Clients in test

表2 评估环境

Table 2 Simulation environment

名称	参数
OS	Windows 10 pro
CPU	Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz
Memory	16GB 2666MHz DDR4

5.2 测试场景

为了表明CARINA的转换效果，考虑这样一种简单的家庭自动化场景。对于4种不同的协议，MQTT客户端进行发布订阅式通信，AMQP客户端进行消息队列式的通信，HTTP和CoAP客户端进行请求响应式通信。假设有一个湿度传感器作为消息发布者将湿度发布到主题humidity，当其中

一个客户端发布消息时,CARINA 进行协议转换,通知相关联的其他客户端进行后续处理。

CoAP:资源/humidity 在 CoAP 服务器上进行创建,发布消息时,湿度传感器客户端采用 PUT 请求发布湿度值。

MQTT:客户端提前订阅/humidity,CoAP 客户端发布消息时,会发送对应的 MQTT 数据包通知订阅客户端。假设有一个窗户开关提前订阅了该主题,那么在收到最新湿度值时可判断是否开窗。

AMQP:客户端提前监听队列名为 humidity 的消息队列,CoAP 客户端发布消息时,客户端能监听到消息队列的变化,从中取出对应的资源,进行任务处理。假设有一个 AMQP 协议的报警器,那么该报警器在接收到大于某个阈值的湿度时就会启动报警通知主人。

HTTP:客户端可以通过 HTTP 的 GET 请求获取/humidity 对应资源下的湿度值。最简单的情形就是主人可以通过 APP 端查看湿度值的变化。

5.3 转换结果

本文采用第 5.2 节的案例研究进行 HTTP,CoAP,MQTT 和 AMQP 客户端相互通信的测试,其中 HTTP 使用相关的 REST API,MQTT 客户端采用可视化的 MQTT.fx,COAP 客户端采用 Coapthon 来发布信息,AMQP 采用 RabbitMQ 提供的客户端。

表 3 列出了使用抓包工具 Wireshark 捕获到整个转换过程的数据包。其中,序号 1－16 的数据包代表 MQTT 和 AMQP 提前订阅的数据包,序号 17－18 的数据包代表 CoAP 协议进行数据发布的数据包,序号 19－29 代表 CARINA 发往 MQTT 客户端和 AMQP 客户端的数据包,30－31 代表 HTTP 客户端通过 GET 请求成功获取到资源的数据包。

表 3 Wireshark 在转换过程中捕获到的数据包

Table 3 Data packets in Wireshark during conversion

No	Time	Src	Dst	Protocol	Info
1	0	127.0.0.1	127.0.0.1	MQTT	Connect Command
2	0.000533	127.0.0.1	127.0.0.1	MQTT	Connect Ack
3	1.588049	127.0.0.1	127.0.0.1	MQTT	Subscribe Request(id=1) [humidity]
4	1.589487	127.0.0.1	127.0.0.1	MQTT	Subscribe Ack(id=1)
5	11.928174	::1	::1	AMQP	Connection.Start
6	11.929321	::1	::1	AMQP	Connection.Start-Ok
7	11.929573	::1	::1	AMQP	Connection.Tune
8	11.92995	::1	::1	AMQP	Connection.Tune-Ok
9	11.93002	::1	::1	AMQP	Connection.Open vhost=/
10	11.930356	::1	::1	AMQP	Connection.Open-Ok
11	11.930985	::1	::1	AMQP	Channel.Open
12	11.931427	::1	::1	AMQP	Channel.Open-Ok
13	11.931806	::1	::1	AMQP	Queue.Declare q=humidity
14	11.932011	::1	::1	AMQP	Queue.Declare-Ok q=humidity
15	11.932364	::1	::1	AMQP	Basic.Consume q=humidity
16	11.932585	::1	::1	AMQP	Basic.Consume-Ok
17	61.494895	127.0.0.1	127.0.0.1	CoAP	CON,MID:53579,PUT,TKN:e6 bf,/humidity
18	61.495609	127.0.0.1	127.0.0.1	CoAP	ACK,MID:53579,2.04 Changed,TKN:e6 bf,/humidity
19	61.495807	127.0.0.1	127.0.0.1	MQTT	Publish Message [humidity]
20	61.589377	127.0.0.1	127.0.0.1	MQTT	Ping Request
21	61.589598	127.0.0.1	127.0.0.1	MQTT	Ping Response
22	63.513263	::1	::1	AMQP	Queue.Declare q=humidity
23	63.513452	::1	::1	AMQP	Queue.Declare-Ok q=humidity
24	63.513732	::1	::1	AMQP	Basic.Publish x= rk=humidity
25	63.513792	::1	::1	AMQP	Content-Header
26	63.513843	::1	::1	AMQP	Content-Body
27	63.514027	::1	::1	AMQP	Channel.Close reply=Normal shutdown
28	63.514087	::1	::1	AMQP	[TCP ACKed unseen segment] [TCP Previous segment not captured] Basic.Deliver x= rk=humidity Content-Header Content-Body
29	63.514165	::1	::1	AMQP	Channel.Close-Ok
30	242.73529	::1	::1	HTTP	GET /humidity HTTP/1.1
31	242.736761	::1	::1	HTTP	HTTP/1.0 200 OK(text/html)

上述案例只是简单地表示了 4 种协议可以进行互操作性的一种验证。CARINA 可以做到这 4 种协议中任何两种协议都可以进行交互,一共包括 12 种情况的相互操作。

5.4 转换时间

为了更精准地表明 CARINA 的性能,本文测试了 4 种协议两两进行转换的时间。由于没有其他工作和本文中研究的协议完全相同,且它们对转换时间的计算不尽相同,因此本文选取 Eclipse 官方项目下已经成熟使用的 Ponte 子项目作为对比对象,不同协议的转换时间统一以在抓包工具中捕获到不同协议数据包的间隔时间为基准。例如从 HTTP 到 MQTT 的转换,本文以 HTTP 的请求包和 MQTT 的订阅包之间的时间间隔作为转换时间。表 4 列出了 Ponte 和 CARINA 对不同协议进行转换的时间。从表中可以发现,CARINA 的转换时间在大多数的情况下明显优于 Ponte 的转换时间。CARINA 最快的转换是从 CoAP 到 MQTT 的转换,其转换时间几乎是 Ponte 转换时间的 1/10。其他两者都有的

转换，CARINA 在时间上也明显优于 Ponte。这是因为 Ponte 在进行转换时使用了 Mosquitto[22] 这样的中间件，有大量的时间花费在了中间件的转发上，在相同的协议下，CARINA 直接在最底层实现了中间件的功能，因此可以达到较小的耗时，这也证明了本文所提方案的转换效率较高。

表 4 转换时间

Table4 Conversion time

(ms)

	Ponte	CARINA
HTTP->MQTT	6.344	1.540
MQTT->HTTP	2.419	1.171
CoAP->MQTT	4.074	0.416
MQTT->CoAP	2.994	1.181
HTTP->CoAP	1.287	0.193
CoAP->HTTP	1.056	1.181
HTTP->AMQP	×[1]	11.169
AMQP->HTTP	×	1.238
CoAP->AMQP	×	7.761
AMQP->CoAP	×	0.232
MQTT->AMQP	×	9.300
AMQP->MQTT	×	1.500

注：×[1]表示不支持的转换类型。

此外，CARINA 相比 Ponte 增加了对 AMQP 协议的支持，整体来说支持的转换增加了一倍。对新协议的支持同样只耗费较少的转换时间，基本上都是在毫秒量级的，最坏的情况也只需要 11ms，最好的情况能达到 0.1ms。这也就意味着协议转换的时间并不会对通信造成很大的负荷。

5.5 数据量对转换时间的影响

数据包的大小对于物联网设备很重要，因为它们意味着更少的处理时间和传输时间。同样地，数据包的大小也是影响转换时间的一个重要因素，好的转换方法不会因为数据量的增加而导致转换时间急剧增加。本文测量了所提方案分别在数据量大小为 10，100，500，1 000 字节下的转换时间。图 8 显示了数据量对转换时间的影响。

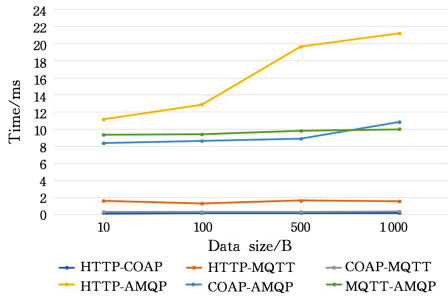

图 8 数据量对转换时间的影响

Fig. 8 Impact of data size on conversion time of CARINA

可以明显看到，协议转换时间并没有随着数据量的增长而成倍波动。与 AMQP 有关的转换由于使用了 RabbitMQ，转换时间呈现上升的趋势。由于增加的转换时间降低到了微秒量级，在图 8 中甚至看不到明显的变化。事实上，在一定的数据量内，系统的转换时间基本上都不会有很大的变化，因为 CARINA 处理的是最底层的字节，耗费的只是 CPU 时间，在如今 CPU 性能如此强大的今天，转换时间也就很难表现出较大的增长。因此，数据量的大小对于 CARINA 来说并不是一个很明显的影响因素。

结束语 随着接入物联网设备的增多，以及通信协议的日益复杂，物联网设备之间的通信变得愈加困难。针对物联网设备的互操作性问题，本文提出了一种高效可扩展的应用层协议转换方案 CARINA。该方案可以在 HTTP，CoAP，MQTT 和 AMQP 协议之间进行相互转换。具体地，CARINA 直接对原始协议数据包进行解析和信息存储，通过构造不同协议关键方法的映射表完成不同协议的方法转换，最后构造目标协议的数据包进行发送，具有较高的效率和扩展性。实验结果表明，该方案比同类方案支持更多的协议且花费的转换时间更短。此外，本文方案的转换时间也不会随着数据量大小的增加而线性增长。

本文的方案同样存在一些局限性，如只测量了单一系统下所提方案的有效性和效率。由于物联网环境复杂，场景众多，未来我们希望该方案能够适用于更多的主流系统以及更多的实验场景，探索物联网设备互操作的更多可能。

参 考 文 献

[1] AL-FUQAHA A, GUIZANI M, MOHAMMADI M, et al. Internet of things: A survey on enabling technologies, protocols, and applications[J]. IEEE Communications Surveys & Tutorials, 2015, 17(4): 2347-2376.

[2] LOMBARDI M, PASCALE F, SANTANIELLO D. Internet of things: A general overview between architectures, protocols and applications[J]. Information, 2021, 12(2): 87.

[3] ZHOU J, WEI G. Two patterns in conversion between HTTP2 and CoAP: Request-Reponse and Publish-Subscribe[C]// 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, 2019: 1178-1184.

[4] DA CRUZ MAURO A A, RODRIGUES J J P C, LORENZ P, et al. A proposal for bridging application layer protocols to HTTP on IoT solutions[J]. Future Generation Computer Systems, 2019, 97: 145-152.

[5] DA CRUZ MAURO A A, RODRIGUES J J P C, PARADELLO E S, et al. A proposal for bridging the message queuing telemetry transport protocol to HTTP on IoT solutions[C]// 2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech). IEEE, 2018: 1-5.

[6] WAHER P. Bridging MQTT & XMPP Internet of Things networks[J]. Computer Science, Engineering, 2014, 19(5): 253-261.

[7] DAVE M, PATEL M, DOSHI J, et al. Ponte Message Broker Bridge Configuration Using MQTT and CoAP Protocol for Interoperability of IoT[C]// International Conference on Computing Science, Communication and Security. Singapore: Springer, 2020: 184-195.

[8] SAITO K, NISHI H. Application Protocol Conversion Corresponding to Various IoT Protocols[C]// IECON 2020 The 46th

Annual Conference of the IEEE Industrial Electronics Society. IEEE,2020:5219-5225.

[9] SHELBY Z, HARTKE K, BORMANN C. The constrained application protocol(CoAP)[EB/OL]. https://tools. ietf. org/html/rfc7252.

[10] HUNKELER U, TRUONG H L, STANFORD-CLARK A. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks[C]// 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops(COMSWARE'08). IEEE,2008:791-798.

[11] STANDARD O. MQTT version 3. 1. 1[EB/OL]. https://docs. oasis-open. org/mqtt/mqtt/v3. 1. 1/os/mqtt-v3. 1. 1-os. html.

[12] STANDARD O. Oasis advanced message queuing protocol (amqp) version 1. 0[J]. International Journal of Aerospace Engineering,2012,2018:1-124.

[13] DOBBELAERE P, ESMAILI K. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper[C]// Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems. 2017:227-238.

[14] COLLINA M, CORAZZA G, VANELLI-CORALLI A. Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST[C]// 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications(PIMRC). IEEE,2012:36-41.

[15] ZONG N. Research and implementation of protocol conversion proxy between HTTP and CoAP in M2M communication [D]. Beijing: Beijing University of Posts and Telecommunications, 2016.

[16] ZHOU J P. Research on interoperability between HTTP2 and CoAP based on IoT [D]. Beijing: Beijing University of Posts and Telecommunications, 2019.

[17] ZAINUDIN A, SYAIFUDIN M F, SYAHRONI N. Design and implementation of node gateway with MQTT and CoAP protocol for IoT applications[C]// 2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering(ICITISEE). IEEE,2019:155-159.

[18] Eclipse Foundation. Eclipse IoT [EB/OL]. Available: https://projects. eclipse. org/projects/iot.

[19] DE AM MACÊDO W L, DA R T, MORENO E D. GoThings—An Application-layer Gateway Architecture for the Internet of Things[C]// WEBIST. 2015:135-140.

[20] YANG S J, WEI T C. Design Issues for Communication Protocols Conversion Scheme of IoT Devices[J]. Journal of Internet Technology,2021,22(3):657-667.

[21] SEEGER M, ULTRA-LARGE-SITES S. Key-value stores: a practical overview[D]. Stuttgart: Computer Science and Media, 2009:1-21.

[22] LIGHT R A. Mosquitto: server and client implementation of the MQTT protocol[J]. Journal of Open Source Software, 2017, 2(13):265-267.

WANG Lina, born in 1964, Ph. D, professor, Ph. D supervisor. Her main research interests include information hiding, deep learning and Internet of things.

(责任编辑:柯颖)