Core   P. Wang
Internet Draft   C. Pu
Intended status: Standards Track   H. Wang
Expires: March 11, 2018   Y. Yang
                                                          L. Shao
                                     Chongqing University of
                                     Posts and Telecommunications
                                                          J. Hou
                                          Huawei Technologies
                                          September 07, 2017

OPC UA Message Transmission Method over CoAP
draft-wang-core-opcua-transmission-02

Abstract

OPC Unified Architecture (OPC UA) is a data exchange standard that
provides interoperability in industrial automation. With the arrival
of Industry 4.0, it is of great importance to implement the exchange
of semantic information utilizing OPC UA Transmitting in CoAP. This
document provides some transmission methods for message of OPC UA
over CoAP.

Status of this Memo

Copyright Notice

   Copyright (c) 2017 IETF Trust and the persons identified as the
   document authors. All rights reserved.

   This document is subject to BCP 78 and the IETF Trust's Legal
   Provisions Relating to IETF Documents
   (http://trustee.ietf.org/license-info) in effect on the date of
   publication of this document. Please review these documents
   carefully, as they describe your rights and restrictions with
   respect to this document. Code Components extracted from this
   document must include Simplified BSD License text as described in
   Section 4.e of the Trust Legal Provisions and are provided without
   warranty as described in the Simplified BSD License.

Table of Contents

   1. Introduction

   Internet of things is one of the attractive applications for CoAP
   [RFC7252]. Utilizing OPC UA [IEC TR 62541-1] Transmitting over CoAP
   could meet the demand for industry 4.0 based on the exchange of
   semantic information [I-D.wang-core-opcua-transmition-requirements].
   Similar to OPC UA, CoAP message is exchanged in server/client mode.
   However, their transmission are different. Driven by this, to
   implement OPC UA Transmitting over CoAP, the major problem to be
   solved is how OPC UA packets are transmitted over CoAP. For the
   transport layer of OPC UA, the main message transmission method is
   TCP or HTTP. It is worth noting that the design of CoAP is inspired

by HTTP, thus, there are some similarities in transmission method between them. This document provides some transmission methods for message of OPC UA over CoAP, so that the communication between OPC UA client and OPC UA server could be established.

1.1. Conventions and Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

OPC: OLE for Process Control

OPC UA: OPC Unified Architecture

SOAP: Simple Object Access Protocol

2. Overview of OPC UA

OPC Unified Architecture (OPC UA), standardized as IEC 62541, is a client-server communication protocol developed by OPC Foundation for safety, reliable data exchange in industrial automation. It is the evolution product of OPC (OLE for Process Control, where OLE denotes Object Linking and Embedding), the widely used standard process for automation technology, and is of great importance in realizing industry 4.0. By introducing Service-oriented architecture (SOA), OPC UA enables an open, cross-platform communication with the advantages of web services, robust security and integrated data model.

2.1. Protocol Stack

OPC UA is an application layer protocol that can be built on an existing layers 5, 6 or 7 protocol such as TCP/IP, TLS or HTTP. The OPC UA application layer consists of four sublayers: UA Application, Serialization Layer, Secure Channel Layer and Transport Layer (see Figure 1).

Serialization Layer includes two kinds of data encoding methods: UA Binary and UA XML. The UA XML, based on SOAP/HTTP or SOAP/HTTPS, is firewall friendly. On the other hand, the UA Binary, with least overhead and resource cost, offers an optimized speed and throughput.

The security layer varies according to the selected encoding format. For the HTTPS-based situation, security is implemented at TLS but Security Channel should still be presented even empty. It is

worthwhile noting that the communication based on SOAP/HTTP has been deprecated since 2015, due to the lack of industrial approbation in the WS Secure Conversation.

For the transport layer (not the layer in OSI 7 layer model), options can be UA TCP, HTTPS, SOAP/HTTPS, and SOAP/HTTP. OPC UA defines a UA TCP protocol, which differs from HTTP in two main features: the allowance of responses to be returned in any order and to be returned on a different TCP transport end-point. In addition, UA TCP defines the interaction with the upper security channel.

```
          +-----------------------------------------------+ ------
          |                 UA Application                |
          +-----------------------------------------------+
          +-----------------------+ +---------------------+
          |       UA Binary       | |       UA XML        |
          +-----------------------+ +---------------------+
          +-------------+                   +-------------+ App
          | UA Secure   |                   | WS Secure   |
          | Conversation|                   | Conversation| Layer
          +-------------+                   +-------------+
          +-------------+ +---------+ +---------+ +-------------+
          |             | |         | |  SOAP   | |    SOAP     |
          |   UA TCP    | | HTTPS   | |---------| |-------------|
          |             | |         | |  HTTPS  | |    HTTP     |
          +-------------+ +---------+ +---------+ +-------------+ ------
          +-----------------------------------------------+
          |                    TCP/IP                     |
          +-----------------------------------------------+
```
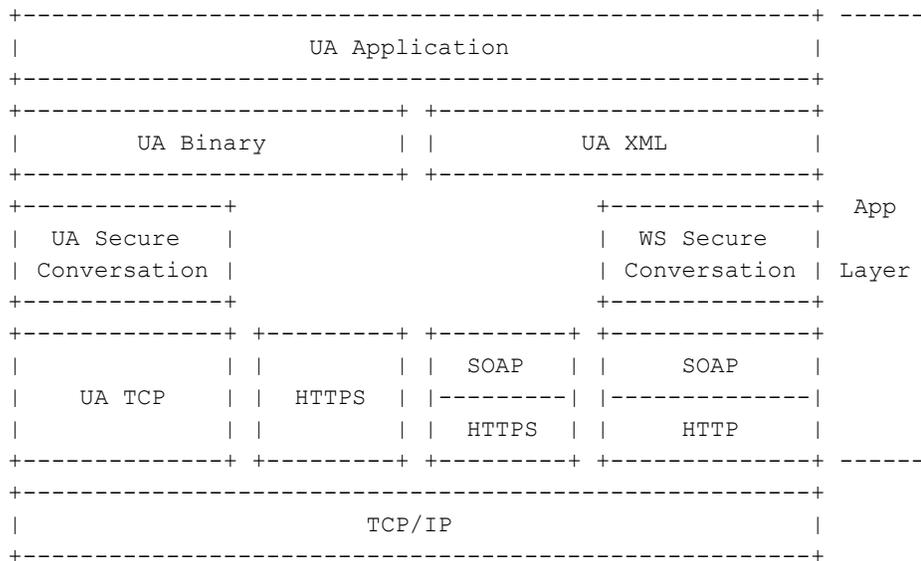
Figure 1: Layering of OPC UA over TCP/IP

2.2. Request/Response Model

The message exchange in UA binary mode is illustrated in Figure 2. After opening the socket, the client starts the connection with the server by using "hello" (HEL) and "acknowledge" (ACK) messages. Afterwards, a pair of messages is needed to open the security channel and define the encryption property. Then another two pairs of messages are exchanged so as to create and activate a session between the client and the server respectively. After these steps, the connection is initiated and the client can send request messages for services. When the request/response process is finished, a reverse process is required for disconnection.

```
        Client Secure    UA                UA    Secure   Server
               Channel   TCP               TCP   Channel
            |       |       |   Open Socket  |       |       |
            |       |       | - - - - - - - >|       |       |
            |       |       |     Hello      |       |       |
            |       |       | - - - - - - - >|       |       |
            |       |       |   Acknowledge  |       |       |
            |       |       | < - - - - - - -|       |       |
            |       |       |                |       |       |
            |       |       |OpenSecureChannelReq    |       |
            |       | - - - - - - - - - - - - - - - >|       |
            |       |       |OpenSecureChannelRes    |       |
            |       | < - - - - - - - - - - - - - - -|       |
            |       |       |                |       |       |
            |       |       |  CreateSessionReq      |       |
            | - - - - - - - - - - - - - - - - - - - - - - - >|
            |       |       |  CreateSessionRes      |       |
            | < - - - - - - - - - - - - - - - - - - - - - - -|
            |       |       | ActivateSessionReq     |       |
            | - - - - - - - - - - - - - - - - - - - - - - - >|
            |       |       | ActivateSessionRes     |       |
            | < - - - - - - - - - - - - - - - - - - - - - - -|
            |       |       |                |       |       |
            |       |       |    Request     |       |       |
            | = = = = = = = = = = = = = = = = = = = = = = = >|
            |       |       |    Response     |       |       |
            | < = = = = = = = = = = = = = = = = = = = = = = = |
```
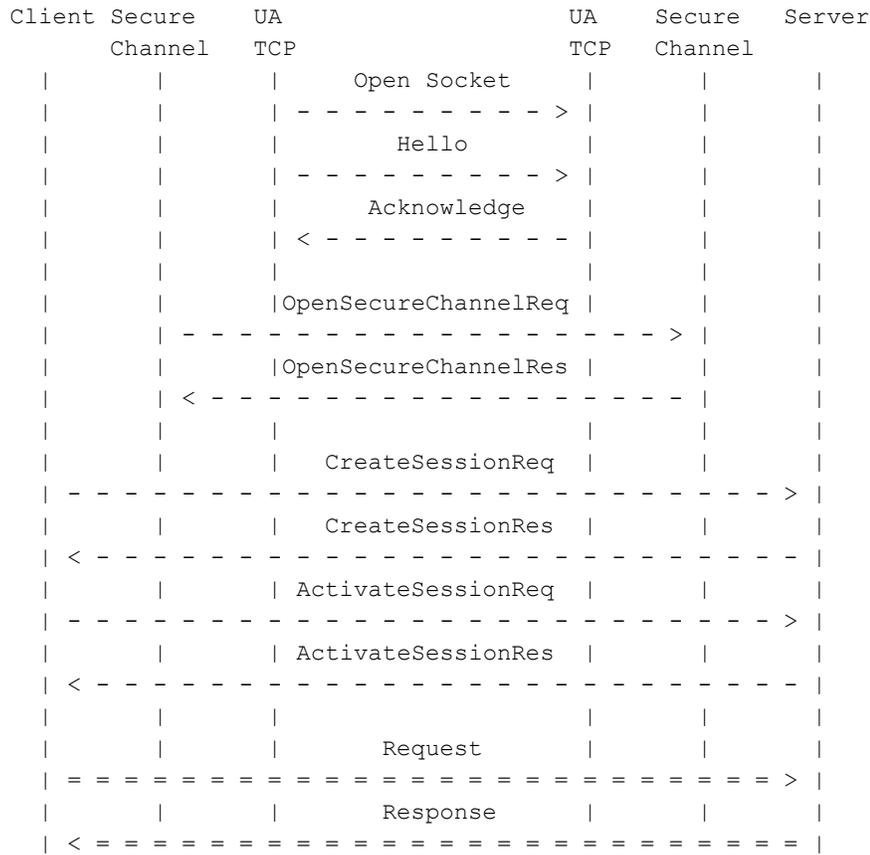
              Figure 2: Request/Response Process of UA TCP

3. Specification of OPC UA over CoAP

As mentioned in section 2.1, OPC UA communications can be conducted
through four options, among which two are related to HTTPS: HTTPS =>
UA Binary; HTTPS => SOAP => UA XML. HTTPS is a security-guaranteed
protocol consisting of a HTTP layer over Transport Layer Security
(TLS), thus the UA Security Channel can be left empty.

Constrained Application Protocol (CoAP) is an application layer
protocol for constrained nodes and networks, which is designed to
easily translate to HTTP for integration with the web. Although CoAP
is built on the unreliable transport layer UDP, it offers a security
mode binding to Datagram Transport Layer Security (DTLS). This
document proposes a transmission scheme based on CoAPs (CoAP + DTLS)
for constrained scenarios. The transmission based on CoAP over

Transport Layer Security (TLS) is also possible. Such "CoAP + TLS" transmission scheme is under development [I-D.ietf-core-coap-tcp-tls] and would be covered in the future version.

The protocol stack of the CoAP based OPC UA is illustrated in Figure 3, including two options at Serialization Layer: UA Binary and UA XML. OPC UA packets are encoded in either binary or xml format, and the option field in the CoAP header can specify parameters that support both formats. Therefore, according to the format specified by the CoAP header, the entire packet of the OPC UA can be encapsulated in the payload of the CoAP message for direct transmission.

```
          +------------------------------+ ------
          |        UA Application         |
          +------------------------------+
          +-------------+ +-------------+
          | UA Binary   | | UA XML      | App
          +-------------+ +-------------+
          +------------------------------+
          |        Secure Channel         | Layer
          +------------------------------+
          +------------------------------+
          |             CoAP              |
          +------------------------------+ ------
          +------------------------------+
          |           UDP, DTLS           |
          +------------------------------+
```
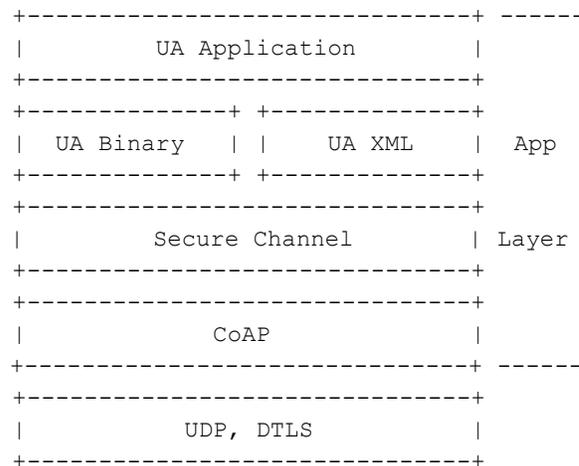
Figure 3: Layering of OPC UA over UDP

Both binary and XML encoding modes are based on the CoAP with an empty UA secure channel in between. For the XML encoding mode, since CoAP layer supports XML encoding format, the SOAP layer in the original stack is not needed.

4. Transmission scheme

4.1. Proxy for OPC UA-CoAP

OPC UA is a protocol mainly for application layer, which defines many services for the different needs of industrial applications. Message is exchanged mainly through server/client mode, utilizing TCP or HTTPS. When security is ignored, OPC UA can be considered to support HTTP transmission. CoAP's design inspiration comes mainly

from HTTP, the two can be mapped between each other to meet the
needs of some special scenes [RFC8075]. Combined with the
characteristics of OPC UA and CoAP, a CoAP proxy can be established
between OPC UA client and OPC UA server. The architecture is shown
in Figure 4.

```
                                   - - - - - - - - - - - - - - - -
                                 //          + - - - - +        \\
                                 ||          | UA-CoAP |        ||
                                 ||      //   | server  |        ||
                                 ||     //    + - - - - +        ||
                                 ||    //                        ||
                                 ||   //                         ||
  + - - - - +  HTTP Request   + - - - - - +  CoAP Request  + - - - - +  ||
  |   UA    | - - - - - - - > | HTTP-to-CoAP| - - - - - - - > | UA-CoAP | ||
  | client  | < - - - - - - - |    PROXY    | < - - - - - - - | server  | ||
  + - - - - +  HTTP Response  + - - - - - +  CoAP Response + - - - - +  ||
                                 ||  \\                          ||
                                 ||   \\                         ||
                                 ||    \\                        ||
                                 ||     \\   + - - - - +         ||
                                 ||      \\  | UA-CoAP |         ||
                                 ||          | server  |         ||
                                  \\         + - - - - +        //
                                   - - - - - - - - - - - - - - - -
```
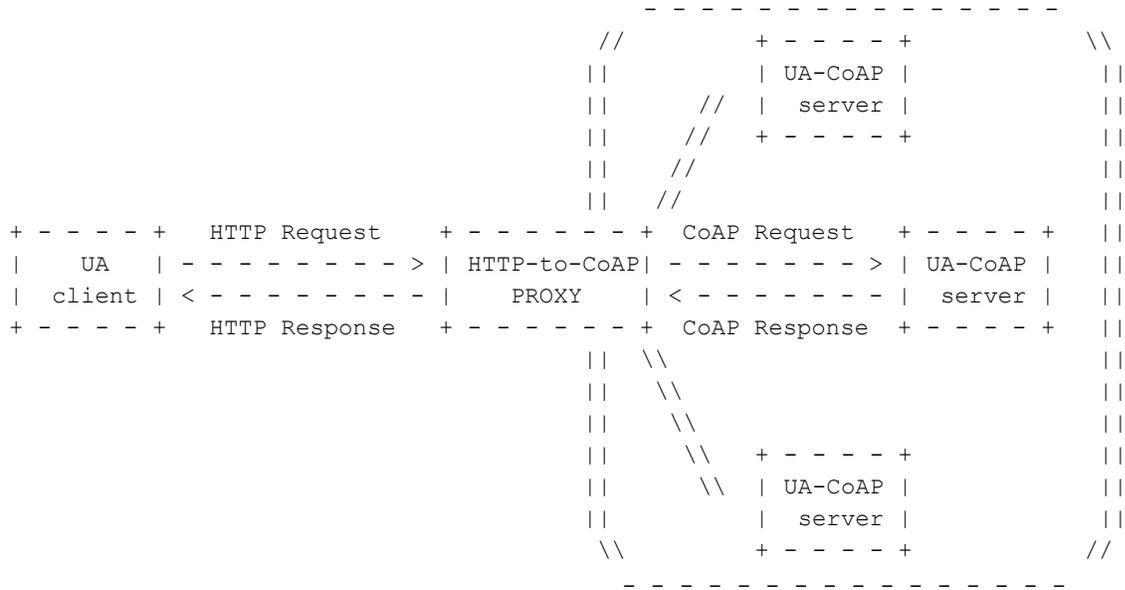
Figure 4: Proxy for OPC UA to CoAP

As shown in Figure 5, assuming all OPC UA servers are based on CoAP
[draft-wang-core-opcua-transmition-requirements], and all OPC UA-
CoAP server can be treated as a network, then introducing UA-to-CoAP
proxy at the boundary of the network. When a traditional OPC UA
client initiates an HTTP request to the UA-CoAP server in the
network, the UA-to-CoAP proxy maps the http request to the
corresponding CoAP request and sends it to the UA-CoAP server in the
network. After receiving the request, the UA-CoAP server sends a
response to the UA-CoAP proxy. The proxy maps the CoAP response to
the HTTP response and returns it to the UA client. For the UA client,
the network proxy and conversion are transparent, in this way, the
transfer of OPC UA in CoAP does not need to make any changes to the
UA Client.

4.2. Direct transmission

The transmission of OPC UA supports TCP protocol and HTTP protocol,
when security is ignored, OPC UA can be considered to support HTTP
transmission. On the other hand, CoAP is seen as a simplified HTTP

protocol so that it can be applied to resource-constrained network.
Therefore, this document considers the use of CoAP to directly
transfer OPC UA messages. OPC UA packets are encoded in either
binary or xml format, and the optional fields in the CoAP header
specify parameters to support these two formats, and the option
field in the CoAP header can specify parameters that support both
formats. Therefore, according to the format specified by the CoAP
header, the entire packet of the OPC UA can be encapsulated in the
payload of the CoAP message for direct transmission, as shown in
Figure 5. According to CoAP, noted that this method of transmission
needs to be modified on the server side and the client side of the
OPC UA according to CoAP.

```
        + - - - - - - +    CoAP  Request      + - - - - - - +
        | UA  client  |  - - - - - - - - - >  | UA  server  |
        |             |  < - - - - - - - - -  |             |
        + - - - - - - +    CoAP  Response     + - - - - - - +
```
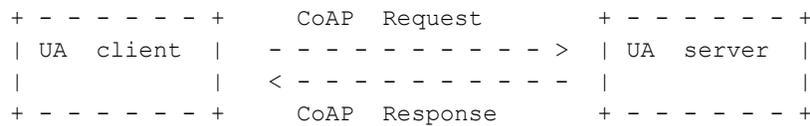
        Figure 5: Direct transmission OPC UA based on CoAP

4.3. REST transmission for OPC UA

   OPC UA is a set of data which exchange specifications for industrial
   communication, the core of the OPC UA protocol are information
   modeling and transmission, which marks each node in the address
   space with a unique identifier. A series of state interactions are
   needed before performing normal reading and writing, including
   message handshaking, opening a secure channel, creating a session,
   activating a session, etc. Besides, some states also need to be
   maintained during read and write operations.

   In OPC UA, each node has an independent identifier in the address
   space, and different types of nodes can establish contact with each
   other by referencing. OPC UA defines a variety of services, and
   these services are fixed, the user cannot arbitrarily modify, each
   service is invoked through a single message, without relying on the
   previous message, the service response is also completed by a
   separate message and do not rely on other messages. The above
   features are in line with the REST architecture, due to CoAP is
   based on the REST architecture. Therefore, it is possible to
   simplify the interaction before the OPC UA performs the normal
   communication, and carry the OPC UA message by using the
   communication mode of the CoAP. Communication process is shown in
   Figure 6.

```
           + - - - - +          + - - - - +
           | client  |          | server  |
           + - - - - +          + - - - - +
                |                     |
                |                     |
                |    CoAPRequest      |
                | - - - - - - - - - > |
                |    CoAPResponse     |
                | < - - - - - - - - - |
```
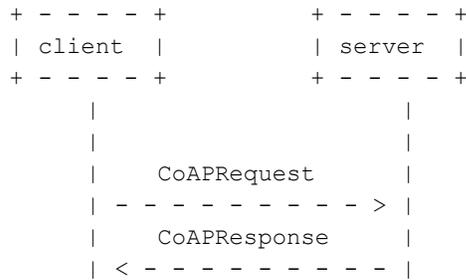
              Figure 6: REST architecture communication of OPC UA

   In Figure 2, the traditional OPC UA requires a series of
   interactions between normal read and write operations. Figure 6
   shows that when using CoAP to carry OPC UA message, the interaction
   process is significantly reduced, which is conducive to the
   application of OPC UA in the restricted scenes. The cost of
   simplifying the interaction process is that the secure channel
   number is set to 0 by default, how to conduct secure data
   interaction needs further discussion.

   5. Publish subscription for OPC UA and CoAP

   As an application sublayer, CoAP provides publish-subscribe
   functionality, primarily for resource or network-constrained
   scenarios. Introducing proxy into the network [I-D.ietf-core-coap-
   pubsub], when a node needs to sleep, the node information is sent to
   the proxy agent, when another node requests to obtain information of
   this node, the broker release function can provide information. OPC
   UA defines the publish-and-subscribe function as a service in the
   service set. The client initiates the subscription request directly
   to the server, and the server periodically sends the information to
   the client. Comparing the characteristics of the two protocols, it
   is found that each of them has its own advantages. Joint design can
   be conducted for constrained applications.

   TODO.

   6. Security Considerations

   This document does not add any new security considerations beyond
   what the referenced technologies already have.

   7. IANA Considerations

   This memo includes no request to IANA.

8. References

8.1. Normative References

[RFC7252]   Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
            Application Protocol", RFC 7252, June 2014,
            <https://tools.ietf.org/html/rfc7252>.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", RFC 2119, March 1997,
            <https://tools.ietf.org/html/rfc2119>.

[RFC8075]   Castellani, A., Loreto, S., and A. Rahman, "Guidelines for
            HTTP-to-CoAP Mapping Implementations", RFC 8075, November
            2016, <https://tools.ietf.org/html/rfc8075>.

8.2. Informative References

[IEC TR 62541-1]
            IEC, "OPC unified architecture-Part1:Overview and concepts-
            IEC 62541", 2016, <
            https://webstore.iec.ch/preview/info_iec62541-
            1%7Bed2.0%7Den.pdf>.

[I-D.ietf-core-coap-tcp-tls]
            Bormann, C., Lemay, S., Tschofenig, H., Hartke, K.,
            Silverajan, B., and B. Raymor, "CoAP (Constrained Application
            Protocol) over TCP, TLS, and WebSockets", draft-ietf-core-
            coap-tcp-tls-09 (work in progress), March 2017.

[I-D.wang-core-opcua-transmition-requirements]
            Wang, H., Pu, C., Wang, P., Yang, Y., and D. Xiong,
            "Requirements Analysis for OPC UA over CoAP", draft-wang-
            core-opcua-transmition-requirements-01 (work in progress),
            December 2016.

[I-D.ietf-core-coap-pubsub]
            Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe
            Broker for the Constrained Application Protocol(CoAP)",
            draft-ietf-core-coap-pubsub-02 (work in progress), October
            2016.

Authors' Addresses

    Ping Wang
    Chongqing University of Posts and Telecommunications
    2 Chongwen Road
    Chongqing, 400065
    China

    Phone: (86)-23-6246-1061
    Email: wangping@cqupt.edu.cn

    Chenggen Pu
    Chongqing University of Posts and Telecommunications
    2 Chongwen Road
    Chongqing, 400065
    China

    Phone: (86)-23-6246-1061
    Email: mentospcg@163.com

    Heng Wang
    Chongqing University of Posts and Telecommunications
    2 Chongwen Road
    Chongqing, 400065
    China

    Phone: (86)-23-6248-7845
    Email: wangheng@cqupt.edu.cn

    Yi Yang
    Chongqing University of Posts and Telecommunications
    2 Chongwen Road
    Chongqing, 400065
    China

    Phone: (86)-23-6248-7845
    Email: 382991208@qq.com

    Lun Shao
    Chongqing University of Posts and Telecommunications
    2 Chongwen Road
    Chongqing, 400065
    China

   Phone: (86)-23-6246-1061
   Email: yjsslcqupt@163.com

   Jianqiang Hou
   Huawei Technologies CO.,LTD
   101 Software Avenue,
   Nanjing 210012
   China

   Phone: (86)-15852944235
   Email: houjianqiang@huawei.com