CoAP over BP for a Delay-Tolerant Internet of Things

Maël Auzias, Yves Mahéo and Frédéric Raimbault
{Mael.Auzias,Yves.Maheo,Frederic.Raimbault}@univ-ubs.fr
IRISA, Université de Bretagne-Sud, France.

***Abstract*—With the advent of the Internet of Things (IoT) a myriad of new devices will become part of our everyday life. Constrained Application Protocol (CoAP), and its extensions, are specifically designed to address the integration of these constrained devices. However, due to their limited resources, they are often unable to be fully connected and instead form intermittently connected and sparse networks in which Delay Tolerant Networking (DTN) is more appropriate, in particular through the Bundle Protocol (BP). This paper addresses the implementation of a BP binding for CoAP as a means to enable Delay Tolerant IoT. After an overview of CoAP and BP, we present a basic implementation of CoAP/BP that we developed and some first experimentation results that validate the feasibility of the approach. Several leads are then explored regarding ways to take advantage of the BP features in order to achieve an optimized CoAP/BP implementation.**

***Keywords*—Constrained Application Protocol, Bundle Protocol, Delay-Tolerant Networking, Internet of Things**

I. Introduction

The recent evolution of the Internet is characterized by a huge increase in the number of physical devices on the network. The Internet is gradually switching from a network of servers and people, online through their personal terminals, to a populated network of sensors, actuators, receivers, RFID and all the little new smart devices that are emerging at a rapid pace. Their common characteristics are their limited computing and storage capacity, their network interconnection capability and their energy constraints. The emerging network forms the so called Internet of Things (IoT). The diversity of these things makes the network more heterogeneous, both in terms of its components and its sub-networks. The extension of existing protocols and sometimes new protocols have been proposed in different layers of protocol stack by the IETF to accompany this profound transformation. The work presented in this article is part of the will to arrange existing protocols into a conceptual framework, to revisit them when needed and to enable the integration and the emergence of new applications that exploit the IoT. Our work is based on the Constrained Application Protocol (CoAP) and the Bundle Protocol (BP).

We will take the example of a smart city to illustrate the interest of these two protocols. The smart city is an emerging concept for urban development in which physical resources, infrastructures, information flows and services have to be monitored, managed and optimized. A common solution is based on the dissemination of a multitude of sensors within the city: traffic detectors, instruments for measuring the air pollution, the noise level, the temperature, etc.; and also actuators coupled with valves, gates, signs, etc. The resources are monitored, their data are collected and forwarded to some central structure, e.g. the city hall. These data then need to be aggregated and analyzed to choose which action should be performed in return. Figure 1 contains an architecture diagram of a smart city based on a sensor network. In this first version it is assumed that smart devices are connected to some infrastructure (via Wi-Fi or cellular access points), directly in the case of smart sensors and through a gateway in the case of lightweight sensors. As far as we know, this connectivity assumption is commonly made in many works on Wireless Sensor Networks (WSN): when sensors are not in standby mode, or off due to lack of energy, they are supposed to always be accessible.

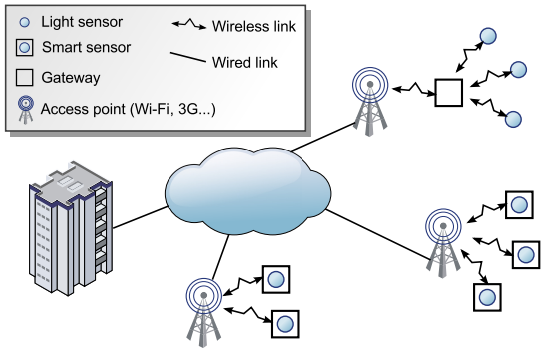


Figure 1. Smart city in a connected IoT environment.

The application protocol CoAP was designed specifically to address the integration of smart devices in such a context; it uses the HTTP features that contributed to the success of the WWW while taking into account the limited capacity and energy constraints of small objects. Each smart device is identifiable and addressable by a URI, just like any resource on the Web. Smart objects may contain a lightweight server, with a small memory footprint, and thus are able to respond to a query and even to send queries to other devices. Lighter objects can only send their measures to a gateway. The advantage of using an HTTP-like protocol in applications for smart cities lies in its ability to expose all these devices as if they were regular web resources. It then becomes possible to search and query the sensors scattered around the city, to aggregate their data with those extracted from other sources

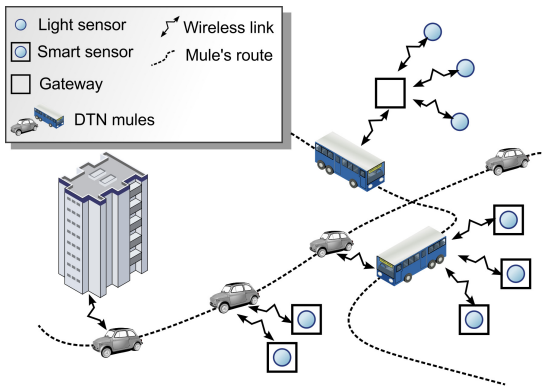


Figure 2. Smart city in a DTN IoT environment.

and, as a result, activate back actuators as data are updated to the web. To limit the number of messages exchanged between the client and the server and to allow for short disconnections CoAP uses the UDP transport protocol in place of TCP on which HTTP is based. However, this type of transport protocol relies on the assumption that the network is stable, shows a rather good reliability and is fully meshed; this assumption is not so frequent in the world of the Internet of Things. For example, in the context of the smart city many physical and financial impediments arise when we try to connect all sensors and actuators to a stable network infrastructure. The subscription costs of cellular networks, the presence of walls and electromagnetic interferences are among the most common barriers. In practice, short-range radio technologies (Wi-Fi, Bluetooth, NFC) are best suited to meet the energy, size and cost constraints of smart devices, as well as the network scaling. In fact, a full mesh ensuring the flow of information between the network of sensors, the actuators and the central servers over an entire city at all times is unrealistic due to network partitioning. The end-to-end connectivity assumption on which CoAP relies may no longer be valid.

The intermittent connectivity is the fundamental assumption in Delay Tolerant Networks (DTNs) for which a specific protocol was proposed: the BP. This kind of network was initially proposed in the context of space telecommunications, but the solutions have been proven useful in other areas, such as military battlefield, the submarine, disaster areas and more recently in sensors networks [1]. The principle is to leverage contacts with mobile nodes to convey information when there is no continuous end-to-end connection between two nodes. The information is entrusted to mobile carriers, which will deliver it to the destination or to other carriers susceptible to reach the destination.

Figure 2 contains a second version of the architecture of a smart city using a DTN network. The figure describes how smart devices may respond to a query initiated by the city hall. A bus passing nearby sensor gateways, or smart sensors, collects their data and carries them through the town. When a taxi intersects the bus, it stores and carries the data, forwarding them until they finally reach the city hall. The BP supports this so called store-carry-and-forward principle without any assumption on the delay taken by messages.

Thus the example presented in this introduction shows how the association of BP and CoAP protocols may help to provide a solution for two problems present at two different levels in the protocol stack of the IoT.

The remainder of this paper is structured as follows. In Section II, a few works concerned by the DTN approach in the context of IoT are introduced, as well as alternative transports for CoAP. Section III gives an overview of both CoAP and the BP, before describing the implementation of CoAP/BP that we developed. A series of experiments based on this implementation are then presented, as a proof of concept of the approach of using the BP as a transport protocol for CoAP. Section IV details several technical aspects of a BP binding for CoAP and leads for optimization. Finally, Section V concludes the paper.

II. Related Work

A. DTN in IoT

Although it has been clear for several years that one of the challenges of the IoT is being able to cope with the high level of dynamism in the network [2], it is only very recently that the DTN related research has been identified as potential enablers for the IoT. For example, in [3], the authors acknowledge the need to handle intermittent connectivity in the IoT and propose a form of opportunistic communication to transmit user interfaces between smart objects. Similarly in [4], delay-tolerant communication is considered in the proposal of an architecture and a routing scheme that form the basis of a framework for integrated RFID sensor networks in the IoT; and in [5], an architecture is proposed in order to interconnect standard-based machine-to-machine platforms for DTNs.

However, there has been a large amount of research effort in DTN that has targeted environments bearing similarities with the IoT, and in particular in the domain of delay-tolerant WSN, with a focus on routing algorithms [6]. Nevertheless, the vast majority of the proposals do not exploit standard protocols. Instead, they design mechanisms dedicated to targeted sensors or applications (e.g., works on underwater sensor networks [7]). One noticeable exception is the work by Pöttner et al in [1] that studies the use of the BP for WSN. The obtained results on computational overhead and network capacity showed that the BP is lightweight enough for WSN even on low-power platforms, with an implementation adapted to a 802.15.4-based underlying network as a replacement for layers 3 and 4 of the IP stack.

B. CoAP bindings

UDP is the standard binding for CoAP. However, several other bindings have been envisaged. The informational Internet Draft [8] examines the requirement of several alternative transport protocols for CoAP, and mentions the potential interest in using the BP. To our knowledge, only the SMS binding has led to an actual test implementation [9].

III. Constrained Application Protocol and the Bundle Protocol

A. Constrained Application Protocol (CoAP)

CoAP [10] offers an application layer protocol that allows resource-constrained devices to interact together asynchronously. It is designed for machine-to-machine use cases and is compliant with the Representational State Transfer (REST) architecture style. CoAP defines a simple messaging layer, with a compact format, that runs over UDP (or DTLS when security is enabled). Its low header overhead and low complexity simplify the processing of CoAP messages for constrained nodes. On top of this message layer, CoAP uses request/response interactions between clients and servers.

If a node needs to send a message in a reliable fashion, in spite of UDP unreliability, then the node will send the message and wait for an acknowledgment. If no acknowledgment is received, the node will retransmit the message several times with an exponential back-off. This retransmission mechanism aims to overcome the unreliability. These messages are referred to as CON (confirmable), in contrast to NON messages (non-confirmable) that nodes can afford to lose.

CoAP applications and resources are identified by URIs following the *coap* scheme (or *coaps* with DTLS), defined as coap:// *host* [:*port*] / *path-abempty* [*?query*] (e.g., coap://zeus.foo.bar:7800/museum/outside-light?number=3). The host part is compulsory, the path identifies the resource within the scope of the host and port (5683 by default), and the query part details the resource access. Group messaging is also possible with CoAP, by specifying a multicast address in the URI host part. This allows several resources to be accessed with a single request.

CoAP requests are derived from the main HTTP methods (GET, PUT, POST or DELETE) and the responses from HTTP statuses. PUT creates a resource, GET retrieves it, POST updates it and DELETE deletes it. If the resource happens to be large, instead of uploading the complete resource with PUT, the method PATCH is recommended as it only uploads a set of changes, that is lighter than the resource itself [11]. As for the responses, CoAP uses HTTP statuses with some slight semantic differences. Informational and redirection HTTP statuses are not used in CoAP.

In addition to its UDP binding, CoAP differs from HTTP regarding its message options. Messages may have one or more options. The list of options includes Content-Format, Accept, Max-Age, Uri-Host, Uri-Path, Uri-Port, Uri-Query. Content-Format informs of the representation format used in the message payload. Accept indicates which Content-Format is preferred by the client. Max-Age is the maximum amount of time during which a response can be cached before its freshness is out-dated. Uri-* options are used to target the suitable resource(s). Some options are meaningful when appearing only once in a request or response, while some others are repeatable, e.g., Uri-Path or Uri-Query.

An interesting work in progress worth citing is the Observe option, [12]. A client subscribes to resource updates by sending a GET request with the Observe option so the server sends notifications upon resource modifications.

B. Bundle Protocol (BP)

The BP is the *de facto* standard for the bundle-layer of the DTN architecture [13]. The BP forms a message-based overlay that follows the store-carry-and-forward principle. The BP defines the format of the messages, called bundles, and the logic layout to process them.

As a network overlay, the BP relies on subnet-specific protocols called Convergence Layers (CL) to transport bundles (e.g., TCP, UDP, LTP). Bundles have a lifetime and will be deleted if it expires. In order to overcome network disruptions and high delays, the BP uses a cache to store bundles. These bundles are either processed by an application (if the destination is on the node), or forwarded to other nodes toward the bundle destination. A bundle destination (or bundle endpoint) is identified by an Endpoint Identification (EID) that takes the form of a URI. A BP endpoint can either be a singleton or a set of BP nodes that register themselves by an EID, thus allowing multicast-like operations to be performed.

The BP bundles have to be routed from node to node. The BP specification does not fix a routing method, and many routing algorithms exist, each of them intended to be adapted to a networking context (e.g. the mobility of the nodes) or to a type of application. A key characteristic of a routing algorithm is its choice to allow multiple copies of a bundle in the network (e.g., as in the epidemic approach).

Bundles are constituted of one primary block (header), then zero or more extension blocks, and one or more payload blocks. The primary block carries options that influence the treatment performed by the nodes that forward and receive the bundle. For example, a Report-When-Bundle-Delivered option will make the destination node emit an administrative bundle when receiving the bundle. Extension blocks (called Metadata Extension Blocks [14]) can be used to make specific processing decisions regarding bundles, e.g., routing decisions.

The BP does not offer a reliable means of communication. Nevertheless, a built-in mechanism, named “custody transfer”, aims to enhance reliability. The custody transfer requests that a BP node takes the responsibility for delivering a bundle to its destination. The responsibility is released when the node forwards the bundle to some other node accepting this responsibility.

C. BoAP

There is at present no available implementation of CoAP over the BP. A possibility to obtain one would be to modify an existing UDP-based implementation. However, simply encapsulating CoAP UDP datagrams into bundles is not feasible because datagram destinations are IP addresses. All BP nodes should then be explicitly identifiable by an IP address, which is not the case. Even in a Java COAP/UDP implementation with a well structured layering, like Californium [15], replacing the entire UDP-binding is neither an option: many layers are too tightly coupled with the notions of IP address and port to perform a straightforward adaptation for the BP, in which these notions are irrelevant. The last solution, the one we choose, is to develop our own implementation of CoAP, named BoAP, with the objective to firstly include a BP binding, and

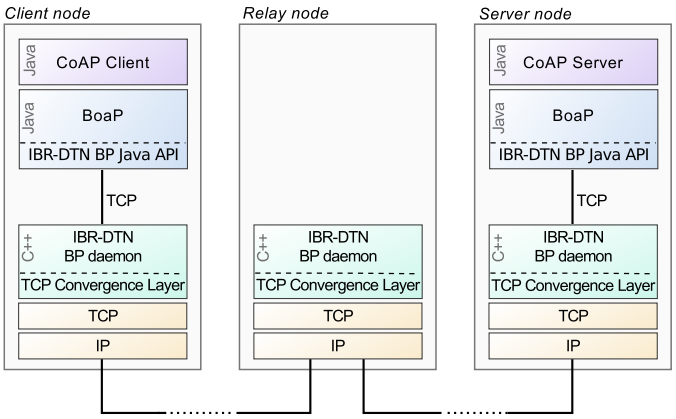


Figure 3. DTN connection between a BoAP Client and a BoAP server

then potentially test some future extensions or modifications of CoAP that would be suited for DTN. The BP binding of BoAP has been developed thanks to IBR-DTN [16]. Besides the fact that it provides elements of a Java API, IBR-DTN is well suited for constrained process-power nodes.

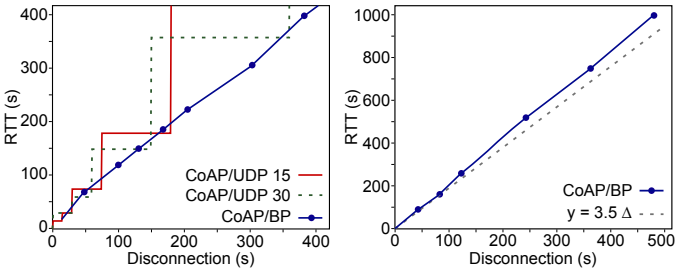
The architecture of a client-server connection through BoAP is illustrated in Figure 3. The CoAP client and server use the BoAP Java API to request or provide resources. BoAP uses the IBR-DTN Java API to communicate with the IBR-DTN BP daemon via a textual TCP socket. The IBR-DTN daemon implements the BP, storing bundles and exchanging them with other IBR-DTN BP daemons in the DTN network. This way of implementing the BP in a separate daemon process (accessible via a TCP socket or another IP-based communication facility) is a versatile solution: the BP daemon could be detached from a sensor node and placed on a more powerful gateway if the sensor node is too constrained. On the down side, it induces an evident additional delay.

D. Proof-of-concept Tests

In order to verify that a CoAP/BP implementation offers reasonable performance when the connectivity is good while supporting long disconnections, performance tests have been conducted in two scenarios. The first one is when the client node and the server node were directly connected with an Ethernet link that was cut on demand, and second one when the client and the server nodes were never directly connected, but formed a DTN network. The speed of a request/response exchange using respectively a CoAP/BP and a CoAP/UDP implementation has been assessed by these tests.

The CoAP/BP tests were performed with BoAP (with a TCP convergence layer but no custody) whereas the CoAP/UDP used Californium (with a standard UDP binding). Note that the implementation of UDP is a part of the kernel and is therefore faster than an implementation of the BP that needs to run above its convergence layer and consequently induces delay; on the installation made for the tests, an ICMP ping is 50 times faster than a BP ping.

We measured the round-trip-time (RTT) in function of the duration of disconnection. The RTT is, as usual, defined as



(a) Intermittent direct connectivity (b) No end-to-end connectivity

Figure 4. Measures of the RTT obtained in the two tested scenarios

the duration between the time the client calls the send method and the time when the response is received back.

1) Scenario 1 – Intermittently directly connected: In this scenario the CoAP/BP client sends a NON request while the CoAP/UDP sends a CON request just after the link between the client and the server has been interrupted. This interruption lasts for what is called the disconnection duration. CoAP/UDP is therefore forced to use its retransmission mechanism, which is triggered after a certain amount of time defined by the CoAP parameter ACK_TIMEOUT. Two values of ACK_TIMEOUT were tested (15 and 30 seconds). CoAP/BP do not retransmit the request but wait for the reestablishment of the link to forward the request.

The obtained RTTs are displayed in Figure 4a. The two curves for CoAP/UDP (one for each value of ACK_TIMEOUT) exhibit a stair-like shape, as expected, due to the exponential back-off. However, CoAP/BP is almost linear, with a slope close to 1. CoAP/BP is slightly slower than CoAP/UDP when disconnections are very short or when the disconnection durations coincide with the retransmission back-off time, yet CoAP/BP is definitely faster than CoAP/UDP when disconnections stop between two dates of retransmission. Indeed, BoaP does not wait to transmit requests.

2) Scenario 2 – No end-to-end path: In this scenario there is no end-to-end path, at any time, between the client and the server. The client sends NON requests to the server periodically (every 30 seconds). In this context, "sends" means that requests were added locally to the BP queue every 30 seconds, whether the node was isolated or not. As shown in Figure 3, a third node, I, is used as an intermediary relay node (simulating a mule) that runs an IBR-DTN BP daemon so that bundles can be transmitted between the client and the server. A cycle of connections/disconnections is enforced in the network, composed of four successive periods: (1) during Δ seconds, only the link between the client node and node I is active; (2) during $\Delta/2$ seconds, both links are inactive; (3) during Δ seconds, only the link between I and the server node is active; (4) during $\Delta/2$ seconds, both links are inactive again. The disconnection duration displayed in abscissa in Figure 4b is the time during which a CoAP node (client or server) is isolated, i.e., a period of 2Δ seconds. Of course, this scenario has not been tested with CoAP/UDP as UDP datagrams cannot be routed to the server and would be lost.

Figure 4b shows the obtained RTT values. When the emission dates are uniformly distributed and the two extreme values

are 2Δ and 5Δ then the expected average of the optimal RTT is $y = 3.5\Delta$. In practice, the measured RTT is slightly longer (around 4Δ), due to the variable reconnection cost, and the processing of BoAP messages. These results show that CoAP/BP can perform reasonably even when the client and the server never meet.

As a conclusion, it can be said that CoAP/UDP is fast and can easily overcome short disconnections. However, if long disconnections are to be expected or if an end-to-end path between client and server is unlikely to exist, then, CoAP/BP is a better alternative.

IV. Toward an optimized CoAP/BP implementation

As we did in BoAP, the first approach when implementing a BP binding for CoAP is to put each CoAP message into a BP bundle and preserve all the CoAP features that had been designed for a UDP binding. However, the BP is actually not a transport-layer protocol and offers richer capabilities than UDP. Therefore several characteristics of the BP may be exploited to simplify a CoAP/BP implementation or improve its performance by delegating some tasks in the BP instead of the CoAP level. We discuss below technical aspects raised by the use of the BP as a CoAP binding and optimization opportunities.

A. Multi-payload

CoAP suggests that each CoAP message should fit into a single UDP datagram. However, the BP allows several payload blocks to be included in the bundles, and it is common for a node to stay isolated during a significant period, without the possibility to forward messages. In a CoAP/BP implementation, it is thus recommended that all the CoAP messages destined for the same endpoint be appended in a single bundle (with each message in a different payload block) until the bundle is ready to be forwarded. This avoids the transport of several primary blocks. Note that adopting an intermediary policy by dispatching the CoAP messages in more than one bundle is generally not feasible because no information is available on the actual convergence layers that will be used along the path, information that would be necessary to ensure a beneficial dispatching.

B. Addressing scheme

In CoAP, a URI is used to identify a server resource. The host and the port number present in the URI serve as the destination IP address and port number to be included in the UDP datagram being sent, and CoAP options URI-Host, URI-Port, URI-Path and URI-Query are inserted into the payload of the UDP datagram to describe the specifically targeted resource.

Contrary to IP networks, in networks running the BP, a BP destination is identified by a URI. Therefore, in CoAP/BP, the BP destination is already described as a URI, which is a part of the bundle meta-data. The CoAP options URI-Host, URI-Port, URI-Path, and URI-Query are therefore either meaningless or redundant. Indeed, the URI-Port option does not have any use in networks running the BP as the nodes do not have any ports. As for the host, the path, and the query URI options, these are redundant with the bundle meta-data and should not be added into the bundle payload.

While the CoAP RFC specifies to use the URI scheme *coap* when UDP is used (or *coaps* with DTLS), there is no mention of alternative transport layers. As the registered *dtn* scheme is not yet precisely defined, we suggest to use a CoAP-compatible scheme in which *host*[:*port*] is replaced by any alphanumeric string, that typically represents a node or a group of nodes. This scheme could be named *coap+dtn*, as suggested in [8].

C. Multicast

CoAP supports requests sent to a group of servers by using an IP multicast address in the URI. In the BP, an EID can be a group of different applications running on different nodes, in other words a set of BP endpoints. This feature can be used as a substitute to the IP multicast group, thus allowing requests to be sent to several destinations at once. Note that it can also be interesting to combine the multi-payload with the multicast. In short, one bundle can carry several responses (or requests) while being sent to a group EID.

D. Confirmable messages

In CoAP, NON messages are sent if no answer is required. This type of message probably forms the majority of the exchanges in a DTN as, in many cases, a DTN exhibits a dynamic behavior that will induce long and often very variable delays likely to be incompatible with a sufficient reliability.

However, CoAP CON messages may be useful in some circumstances, i.e. when the sender of a request needs an acknowledgment. Indeed, CON messages are used to ensure two distinct things: a) the request packet has not been lost during its transport, and b) the server was active, so it received and treated the request. In a DTN, bundle losses are reduced by the routing algorithm itself (for example by generating redundant copies in the network) or via the use of the custody mechanism (in the case of a single-copy routing). The BP ensures the retransmissions of a bundle when necessary, as long as the lifetime of this bundle has not expired. With a BP binding, CoAP message retransmission is then harmful as it would put an unnecessary load on the network. The important, but challenging, thing is to correctly set the lifetime of the bundles. At first, a default lifetime should be fixed to the upper bound of the time required by a bundle to cross the network, plus the time for a destination CoAP node to resume from its potential sleep state. It is suggested that a CoAP option for setting the lifetime of a CoAP message be added, so that a specific lifetime, potentially shorter than the default one, can be assigned to a request or a response.

The following BP mechanisms may also be examined:

- Report-When-Bundle-Acknowledged-By-Application option: this option requests that the BP daemon running on the destination node generate an administrative bundle once the bundle is received and taken in charge by the application (in our case the CoAP layer). This administrative

bundle can be a substitute to the CoAP acknowledgment. Setting this option for CON messages avoids taking care of the acknowledgment at the CoAP level as it is sent directly by the BP daemon. The similar option Report-When-Bundle-Delivered is not so convenient because it does not involve the BP application (i.e. CoAP in our case) and therefore prevents CoAP from piggybacking the acknowledgment to the response if it is quickly available.

- Expedited class of service: there are three classes of service for bundles. The classes are, from the least to the most important, bulk, normal and expedited. Higher-class bundles are forwarded with priority over others, as long as the source is the same. If bulk (or normal) is used for NON messages, it is suggested that expedited be used for CON messages in order to accelerate their transport, and, hence, reduce the probability of their lifetime expiring.

E. Option Accept

The CoAP Accept option of a request can be used to indicate which format is acceptable to the client regarding the content of a response. At present, this option is not repeatable, in other words only one format can be specified at a time. This leads to several exchanges between client and server in order to find a matching content format. Allowing the Accept option to be repeatable could offer a significant benefit by avoiding these exchanges for negotiation. The advantage is not specific to CoAP/BP, but it is particularly important to avoid unnecessary end-to-end transmissions in a DTN.

F. Caching

In CoAP, endpoints may cache responses of GET requests in order to reduce the response time and network bandwidth consumption on future, equivalent requests. The option Max-Age is set on a response to assign a duration of freshness to it. CoAP endpoints are allowed to provide a cached response if it is sufficiently fresh. This caching mechanism is only performed locally, that is, by the sender of the request, or by a proxy (specified in the Proxy-URI option of the CoAP message) that issues the request on behalf of this client.

In a DTN, this caching mechanism could be extended so that relay nodes would also be allowed to provide a cached response, as if these relay nodes were proxies. The gain in response time could therefore be drastically reduced. However, some cross-layering is necessary to implement this mechanism: the BP relay node must pass the bundle payload to a local CoAP proxy code in function of some CoAP-level information. This information includes the URI but also the method type and other CoAP options. These are normally only available in the payload of the bundle, which should remain opaque.

We propose using an extension of the BP called "Metadata Extension Block" [14] to support this cross-layering. A Metadata Extension Block (MEB) is designed to carry additional information used by DTN nodes to make processing decisions regarding bundles. In our case, the MEB would contain all the necessary data about the request so that the BP layer can trigger a local CoAP Proxy code that will potentially send a CoAP response.

V. Conclusion

This paper studied the replacement of UDP by the BP as an underlying transport for CoAP, in order to enable a delay-tolerant Internet of Things. A first implementation of CoAP/BP, called BoAP, allowed us to perform some preliminary tests that showed that the BP can be an effective substitute to UDP as a CoAP binding: BoAP does not largely degrade transmission delays when disconnections are short, and, contrary to CoAP/UDP, it continues to play its role when the connectivity is strongly intermittent. A number of optimizations have been detailed that should permit a more efficient BoAP implementation. In the future, we plan to include these optimizations into BoAP and evaluate them. Besides, some other CoAP and BP features still need some attention, in particular resource discovery and security.

Acknowledgment

This work is supported by the French ANR (Agence Nationale de la Recherche) grant number ANR-13-INFR-012.

References

[1] W.-B. Pöttner, F. Büsching, G. von Zengen, and L. Wolf, "Data elevators: Applying the Bundle Protocol in Delay Tolerant Wireless Sensor Networks," in *9th International Conference on Mobile Adhoc and Sensor Systems (MASS 2012)*. Las Vegas, NV, USA: IEEE, Oct. 2012, pp. 218–226.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] H. Wirtz, J. Rüth, M. Serror, J. A. Bitsch Link, and K. Wehrle, "Opportunistic Interaction in the Challenged Internet of Things," in *9th Workshop on Challenged Networks (CHANTS 2014)*. Maui, Hawaii, USA: ACM, Sep. 2014, pp. 7–12.

[4] F. M. Al-Turjman, A. E. Al-Fagih, W. M. Alsalih, and H. S. Hassanein, "A delay-tolerant framework for integrated RSNs in IoT," *Computer Communications*, vol. 36, no. 9, pp. 998–1010, 2013.

[5] A. Elmangoush, R. Steinke, M. Catalan, A. Corici, T. Magedanz, and J. Oller, "Interconnecting Standard M2M Platforms to Delay Tolerant Networks," in *2nd International Conference on Future Internet of Things and Cloud (FiCloud 2014)*. Barcelona, Spain: IEEE, Aug. 2014, pp. 258–263.

[6] M. J. Khabbaz, A. Chadi M., and F. Wissam F., "Disruption-Tolerant Networking: A Comprehensive Survey on Recent Developments and Persisting Challenges," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 2, pp. 607–640, 2012.

[7] H.-H. Cho, S. T. K. Chen, Chi-Yuan and, and H.-C. Chao, "Survey on underwater delay/disruption tolerant wireless sensor network routing," *Wireless Sensor Systems*, vol. 4, no. 3, pp. 112–121, 2014.

[8] B. Silverajan and T. Savolainen, "CoAP Communication with Alternative Transports, v07," IETF Internet Draft, Dec. 2014.

[9] N. Gligoric, T. Dimcic, D. Drajic, S. Krco, I. Dejanovic, N. Chu, and A. Obradovic, "CoAP over SMS: Performance evaluation for machine to machine communication," in *20th Telecommunications Forum (TELFOR 2012)*. Belgrade, Serbia: IEEE CS, Nov. 2012, pp. 1–4.

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

[11] P. Van Der Stock and A. Sehgal, "Patch Method for Constrained Application Protocol (CoAP), v00," IETF Internet Draft, Mar. 2015.

[12] K. Hartke, "Observing Resources in CoAP, v16," IETF Internet Draft, Dec. 2014.

[13] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proc. of ACM SIGCOMM03*, Aug. 2003.

[14] S. F. Symington, "Delay-Tolerant Networking Metadata Extension Block," IRTF RFC 6258, May 2011.

[15] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: Scalable Cloud Services for the Internet of Things through CoAP," in *International Conference on the Internet of Things (IoT 2014)*. Cambridge, MA, USA: ACM Press, Oct. 2014.

[16] M. Doering, S. Lahde, J. Morgenroth, and L. Wolf, "IBR-DTN: an efficient implementation for embedded systems," in *3rd ACM workshop on Challenged networks*. ACM, Sep. 2008, pp. 117–120.