Comparing different implementations of Bundle Protocol version 7

Marcel Beyer

TU Dresden
marcel.beyer@mailbox.tu-dresden.de

July 13, 2020

Abstract

After gaining experiences from version 6 of the Bundle Protocol, which was published as RFC 5050, the protocol was subject of major changes in version 7, which is currently on the way to be published as an RFC from it's current experimental state.

Version 7 of the Bundle Protocol specifies some features which may be implemented. Furthermore there are already different protocols implementing features on top of the Bundle Protocol, which may or may not be implemented by different implementations. The aim of this paper is to compere which of these features and protocols are implemented in different implementations of the Bundle Protocol version 7.

I. Introduction

The Bundle Protocol is a protocol designed for the end-to-end communication of messages in Delay (or Disruption) Tolerant Networks (DTN). Version 6 of the protocol was published in RFC 5050 [BPv6] in 2007. Until now it is being reworked as version 7 [BPv7] regarding the collected experiences from implementations of version 6.

An overview of the fundamental changes from version 6 to version 7 can be found in [BP-Changes].

This paper compares current implementations of the Bundle Protocol in version 7[1]. Therefor a set of features, which are marked as optional in the standard or may be implemented in different ways, is checked for each of the observed implementations. Furthermore some self-standing protocols which are related to DTN or the BP in special are part of the analysis in this paper.

[1] BPbis-24

II. Introduction to DTN/Bundle Protocol

Delay/Disruption Tolerant Networks [DTN] were originally designed to be used for the Interplanetary Internet, which means that it has to take care of occasionally-connected links in the network which may rely on different protocols and have high delays.

DTN are not limited to interplanetary networks, they may also be used for sensor-based networks, satellite networks with periodic connectivity or underwater networks with frequent interruptions.

Existing protocols had some issues when they were used in DTN, which lead to the development of the DTN architecture and the Bundle Protocol. One of those problems is, that there is no guarantee for stable end-to-end paths between the source and destination for the duration of the communication. Furthermore the established protocols rely on a small end-to-end loss and latency.

The Bundle Protocol is part of the DTN archi-

tecture and is used as an end-to-end overlay for the different links in the DTN. The just named problems are solved by the Bundle Protocol as it is sending all data in form of small portions, named "Bundles" between the communication entities of the network, called "Bundle Nodes". The Bundles, consisting of some metadata and the payload data, are not sent directly to the destination node but on a hop-by-hop approach from one node to another. Each of the nodes is calculating possible routes to the destination, where the bundle may be sent to several next hops. For calculating the routes, several aspects as the physical movement of the nodes or planned link-disruptions may be treated. The Bundle Protocol comes also with the principle of Custody, which means that a bundle is stored on a node until the next node took the custody for the bundle – so it is ensured that there is no packet loss.

III. Implementations

i. ION

The Interplanetary Overlay Network is an implementation for DTN developed at the Jet Propulsion Laboratory. It is developed in C, but also has a Python interface. ION supports version 7 of the Bundle Protocol from version 4.0.0 on.

ii. μPCN

μPCN (Micro Planetary Communication Network) is an implementation of the Bundle Protocol in version 6 and 7 for POSIX compliant operation systems and the ARM Cortex STM32F4. It is written in C and licensed under the BSD 3-Clause License.

μPCN comes with its own *μPCN Application Agent Protocol* (AAP). AAP enables applications to communicate with μPCN over a simple TCP connection to register (sub)EIDs, send and receive bundles. Bundles which were enqueued for sending by μPCN are acknowledged with an transmission confirmation to the application. For the application it is also possible to cancel already enqueued bundles.

iii. pyDTN

PyDTN is an implementation of the Bundle Protocol which started with version 7 of the BP and is developed in Python by the Slovakian company X-Works.

iv. LibDTN

The core of LibDTN is the basis for the DTN node application "Terra" and implements the Bundle Protocol in version 7. It is written in Java and developed by the swiss company RightMesh.

IV. Compared Features

This section will provide you a short overview about the features compared in this paper.

i. Status Reporting

The sender of a bundle can request status reports to get information how the bundle is processed through the network. So he will get updates when the bundle is being forwarded, deleted, received by a node or finally delivered to the destination node.

To request status reports the corresponding bits for each event (reception, forwarding, delivery, deletion) can be set in the Bundle Processing Control Flags. Furthermore there is a bit to request the time at which an event caused a status report. The endpoint ID (EID) of the node to which the status reports will be send is defined in the Primary Bundle Block.

Beside the information what happened to the bundle, the reporting node should also write when and why the action took place in the status report. Some general information of the original bundle, like the creation timestamp or the original source node, are also part of the status report.

ii. Fragmentation and reassembly

ii.1 Fragmentation

In some cases it might be possible that a node needs to reduce the size of a bundle to transmit it to the next node. In this case the bundle can be fragmented if the original sender of the bundle did not set the "do not fragment" flag in the bundle processing flags. It is also possible that the same bundle gets fragmented several times.

When a bundle is fragmented, the first fragment has to include all extension blocks of the original bundle. All fragments have the same source node ID and timestamp as the original bundle. But the primary block of the fragments must differ from that of the original bundle, in that it must indicate that it is a fragment and that the fragment offset and total application data unit length must be given.

ii.2 Reassembly

Fragmented bundles need to be reassembled at latest when they arrive on their destination. But it is also possible that a node on the way is already reassembling a bundle.

To reassemble a fragmented bundle, the payloads of all fragments with the same source EID and creation timestamp are concatenated to a single payload for the reassembled bundle. It has to be verified that the payload is non-overlapping and has the same size as given in the primary block.

iii. URI schemes

There are different URI schemes to address EIDs in the bundles.

iii.1 dtn

The `dtn` URI scheme is defined by the Bundle Protocol itself. The Endpoints represented by URIs with this scheme may be single nodes or groups of nodes.

iii.2 ipn

URIs of the `ipn` scheme are always representing single nodes and were originally specified by the Compressed Bundle Header Encoding [RFC 6260] for BPv6. While the compressed encoding was convicted to BPv7 from CBHE, the scheme and functionality kept the same.

The EID format originally specified in the BP results in long ASCII strings, which has potential for improvement, especially when short payloads are transferred. With CBHE a new URI scheme ("ipn:") was defined, where EIDs are represented by a node and a service number, each in a range from 1 to $2^{64} - 1$. CBHE was not intended to be used universally in Delay Tolerant Networking, so it is no problem, that the address space is downscaled by using it.

iii.3 imc

Interplanetary Multicast [IMC] specifies, how multicast communication could be implemented in a DTN using the Bundle Protocol. To address a multicast group in a way compatible to CBHE, the `imc` URI scheme was specified.

iv. Bundle-in-Bundle Encapsulation (BIBE)

In version 7 of the Bundle Protocol the concept of custody is not further implemented in the Bundle Protocol itself but specified in the Bundle-in-Bundle Encapsulation Protocol [BIBE]. If custody is needed for a bundle, BIBE will be used as the CLA for sending the bundle. This results in the encapsulation of the original bundle in a new bundle formed according to the BIBE specification. The outer bundles, also called BIBE protocol data units, are Bundle Protocol administrative records with the record type 3. Other nodes implementing BIBE are sending custody signals, which are also BP administrative records, if they have some relevant information regarding the custody of a bundle (e. g. accepting custody).

v. Extension Blocks

There are some extension blocks defined by the Bundle Protocol, but there might – and probably will – be more extension blocks defined separately. Due to this independence from the BP not all nodes will support every type of extension blocks.

v.1 Previous node extension block

This block provides the EID of the node which sent the bundle to the node it is residing currently.

v.2 Bundle age extension block

This block sums up all transmission times of the bundle added up with the sum of time it resided on nodes. With this information, nodes without a precise clock are able to estimate if the lifetime of the bundle has already exceeded.

v.3 Bundle hop count limit extension block

The hop counter in this block is incremented every time the bundle is being forwarded. As soon as the hop counter exceeds the hop limit, the bundle should be deleted. This is a safety mechanism to prevent routing loops – comparable to the hop limit[2] in IP.

v.4 Metadata extension block

To improve the decisions, how bundles are handled by nodes, e. g. to which node they will be forwarded or if it will be stored, it is useful to have information which data is carried by a bundle. Therefor [RFC 6258] specifies the Metadata Extension Block which enables to store different metadata types. In [RFC 6258] only the URI Metadata Type is defined. Also a range of type IDs is reserved for private use.

vi. BPSec

Originally there was the Bundle Security Protocol (BSP) [RFC 6257] to provide integrity and confidentiality for BPv6 bundles. Later the Streamlined BSP [SBSP] was developed to overcome some problems the BSP had. So the routing of bundles and the security features became decoupled. For version 7 of the bundle protocol, the BPSec [BPSec] protocol was developed and supersedes SBSP.

[2] or time to life in IPv4

vii. DTN IP Neighbor Discovery

In DTN networks it is often not possible to plan in advance which nodes will be reachable. Therefor the IP Neighbor Discovery [IPND] provides a way to discover reachable nodes by sending small UDP messages in the IP underlay. By receiving those messages neighbors can be learned.

viii. Implementation details

Some procedures defined in the Bundle Protocol are kept open as implementation details.

viii.1 Malformed bundles

The BP specifies, that malformed parts of bundles “may” be discarded or corrected.

viii.2 Retransmitting when sending failed

If a node was unable to forward a bundle with at least one convergence layer adapter, it “may” resend the bundle.

V. Features of the implementations

i. ION

Status reporting ION is supporting status requests for sending and receiving bundles. The decision, which status reports are requested, is kept for the application layer software. If another node requested status reports, ION is able to follow the “report status time” flag and include the current time.

Fragmentation and reassembly ION implements bundle fragmentation and is taking care of the don't fragment flag of forwarded bundles. If a DCCP-based [DCCP] CLA is used, fragmentation is not supported. For bundles created on a node running ION, proactive fragmentation is applied – which means that bundles will already be fragmented if it is known that one of the nodes on the way of the bundle only accepts smaller bundles. The fragmentation is done at the moment when the bundle gets dequeued for transmission.

Reassembly of bundles is done, when all fragments were received successfully and the bundle was destined for the local node.

URI schemes ION is supporting both `dtn` and `ipn` as the two URI schemes for unicast bundles. Sending to multicast destinations with `imc` is also implemented in ION. The implementation of BP version 6 included in ION supports CBHE [RFC 6260] too, so that the `ipn` scheme is available also in BPv6.

Bundle-in-Bundle Encapsulation BIBE is implemented in ION, but the documentation [ION] states, the configuration to run ION as a CLA too would be "not as simple as one might think". ION has a separate utility called `bibeadmin` to realize BIBE. The parameters for the encapsulation (like Timeouts etc.) are configured in an autonomous configuration file for `bibeadmin`.

Extension blocks ION is shipped with support for the Previous Node, Bundle Age, Bundle Hop Count Limit and Metadata Extension block.

Additionaly to these extension blocks, ION includes also an implementation for the Query Extension Block [BPQ] and and the Spray and Wait Block.

The letter one is used to propagate forwarding permits using the Spray and Wait [SNW] routing protocol.

With the Query Extension Block it is possible to reuse information which is still stored on intermediate nodes in the DTN. So if a node A has received some information and node B requests the same information, B can get the bundle from an intermediate node which still stores the answer originally sent to A. By doing this, the transmission can be sped up since the answer needs fewer hops to reach B.

ION is built in a way that makes it easy to support other Extension Blocks without changing the ION code base.

BPSec Achieving integrity or confidentiality for bundles by using BPSec is supported by ION.

DTN IP Neighbor Discovery Discovering nearby nodes running the Bundle Protocol is supported by listening to IPND packets. If the IPND messages indicate a change of a neighboring node (becoming reachable or connection loss) the local CLAs are informed. Nodes running ION are also able to announce themselves by IPND.

Malformed bundles If a node running ION is receiving a malformed bundle, the bundle will be discarded.

Retransmission Per default ION is using reliable Convergence Layer Adapters like TCP or red-part LTP, which automatically ensures retransmission if data gets lost. This is especially the case when custody was requested. But it is also possible to request ION to use unreliable protocols operating on a best effort basis by setting a ION specific tag before the transmission of a bundle gets initiated. To protect these bundles, ION is also retransmitting bundles if it was not able to send them successfully. If the next node, the bundle was sent to, is denying custody, ION will try to send the bundle to another node which takes custody for the bundle.

ii. μPCN

Status Reporting μPCN is sending status reports if they were requested by the origin node.

If the status time was requested for status reports, it will be added to the reports.

For bundles sent with μPCN status reports can be requested by setting the corresponding bundle control flags.

Fragmentation and reassembly During the processing of bundles, μPCN is able to fragment every bundle – so also fragments may be fragmented again. If a bundle is being forwarded, it is possible that it is fragmented by μPCN although the "must not be fragmented" bit was set, since this bit is not checked by μPCN.

During the reassembly of fragmented bundles, it is checked that there are no blanks in the payload of the assembled bundle. If any gaps are recognized, the bundle is seen as malformed.

URI schemes μPCN is supporting the `dtn` and `ipn` schemes for sending and receiving bundles. The `imc` scheme for multicast bundles is not supported.

The checks for the URI scheme are not implemented the same way in all parts of the code. Sometimes the EID is checked to be the null endpoint ID or a URI in `dtn` scheme while the URI is assumed to be in the `ipn` scheme if none of those two matches. In other parts there is also an explicit check for the EID to be in `ipn` scheme and if neither the null endpoint, `dtn` scheme nor `ipn` scheme match, an error is thrown.

Bundle-in-Bundle Encapsulation BIBE is not implemented in μPCN. But since the code for processing bundles is the same for version 6 and 7 of the Bundle Protocol, all checks and procedures for handling custody are already in place. They are just not executed because there are checks if the processed bundle is in version 6 of the Bundle Protocol. The code for handling custody is also still using only the control flags as defined in version 6 of the Bundle Protocol.

Extension blocks μPCN is able to forward bundles with unknown extension blocks without discarding them (except discarding was requested by the control flags).

Further support is only implemented for the bundle age and bundle hop count limit block. During the lifetime check of a received bundle, the bundle age delivered with the extension block is not yet used to decide if the lifetime of the bundle exceeded – this check is marked as a to-do in the code. Any more there is no serializer for the bundle age extension block yet. For the bundle hop count limit extension block the serializer and deserializer is already finished and for all received bundles, which come with the extension block, the hop count is checked and increased.

BPSec There is no support for BPSec at μPCN yet.

DTN IP Neighbor Discovery IPND was implemented in μPCN but the developers decided to remove IPND from the μPCN Bundle Protocol agent. They see IPND comparable with a routing daemon which should run as a own process on the nodes and communicate with μPCN. The contacts learned by a IPND daemon could be propagated to μPCN for example using AAP.

Malformed bundles If a bundle is detected to be malformed during processing it, it will be discarded.

Retransmission If it was not possible to send the bundle to the next node, it will be deleted. This means that possibly requested status reports will be sent before the bundle is discarded on the local node. The custody implementation for version 6 is also sending a custody release at this point.

iii. pyDTN

Currently the destination EID of all received bundles is checked for registration on the local node. If the EID is unknown, the bundle

will not be processed further. Bundles with a known EID, for which the local node is registered as a receiver, are checked for correct formatting before the payload is handed over to the corresponding application layer software. With this behavior pyDTN is not forwarding bundles it received for other destinations.

Status Reporting The status reports and the corresponding flags for requesting them are implemented, however the flags of received bundles are not checked and in this way no status reports are sent. The functions for serializing new bundles are supporting to set the control flags to request status reports, but the functions creating new bundles are not using this possibilities.

Fragmentation and reassembly The *is fragmented* flag of bundles is transferred to the internal representation of the bundles but neither the functions for receiving nor sending bundles are using the flags, since the procedures for fragmentation or reassembly are not implemented.

URI schemes The `dtn` and `ipn` schemes for unicast communication are implemented in pyDTN. Addressing multicast destinations using the `imc` scheme is not supported. If a bundle contains an EID, which is not using the `dtn` or `ipn` scheme, pyDTN will throw an *unknown schema* error.

Bundle-in-Bundle Encapsulation Currently BIBE is not implemented.

Extension blocks Extension Blocks are implemented in the pyDTN core, so adding support for new Extension Blocks is not possible without touching the code of the pyDTN core.

The block types for the Previous Node, Bundle Age Block and Bundle Hop Count Limit Extension Blocks are implemented in pyDTN. Currently the Metadata Extension Block is not supported.

When serializing a new bundle, it is possible to set the Previous Node Block. But since there is no forwarding of bundles yet, this is never used.

As the functions for receiving bundles is quite minimalistic and the lifetime of received bundles is not checked, there is also no need to use the information carried in the Bundle Age block. Like the Previous Node block, the Bundle Age block is supported in the bundle serialization, but not used while sending bundles.

The same counts for the Hop Count Limit Block. It is possible to serialize bundles with the hop count and limit set to a specific value, but the functions for sending a bundle are not using it. Furthermore, it is not checked if the hop count exceeds the hop limit, when a bundle is received.

BPSec BPSec is not supported by pyDTN yet.

DTN IP Neighbor Discovery The implementation of DTN IPND was started but is not finished yet. Currently the code for sending and receiving beacons is existing but not further integrated. Also datatypes for strings and SDNV in the IPND messages have to be implemented.

Malformed bundles If a received bundle is not passing the checks for correct formatting, it will be discarded.

Retransmission If there is no CLA which was able to send a bundle successfully or if there is no known route to send the bundle, the bundle will not be retransmitted automatically.

REST API pyDTN comes with an REST API to easily send bundles. There is also an REST API to schedule contacts with microPCN nodes.

iv. libDTN

Status Reporting Sending status reports is supported by libDTN. During the processing of a received bundle, all requested status reports are created and gathered before they are sent out when the processing finished. Requesting status reports is possible by just setting the corresponding control flags in the primary block of the bundle which should be transmitted. The creation timestamp is added to all status reports.

Fragmentation and reassembly Bundle fragmentation (and therewith reassembly) is currently not supported by libDTN.

URI schemes Currently the `dtn` and `ipn` scheme are implemented, the `imc` scheme for multicast addresses is not supported.

Bundle-in-Bundle Encapsulation There is no support for BIBE.

Extension blocks The Previous Node, Bundle Age and Hop Count Hop Limit block are implemented in libDTN. So it is possible to create those blocks and send them in bundles created with libDTN. libDTN is also able to forward bundles containing those blocks correctly, but it is not checking the content of those blocks when a bundle is received. The metadata block is not supported at all.

libDTN comes with an implementation for an extension block called "Routing Block" which has the Block Type 42. This block type is currently not registered as a Bundle Block Type at IANA. Following the comments in the libDTN code, this block should be used to select the routing strategy. I could not find any further specification of the extension block type.

BPSec The Authentication, Confidentiality and Integrity Blocks of BPSec are supported by libDTN. The implementation of those blocks comes also with functions to use them, for example to encrypt a given plain text block.

DTN IP Neighbor Discovery IPND is not supported.

Malformed bundles If there occur any errors during the processing of a bundle because the bundle is malformed, the bundle will be deleted silently by libDTN. All status reports collected until this point will be sent anyhow.

Retransmission If forwarding a bundle fails, for example because the convergence layer was not able to send the bundle, it will be discarded.

VI. Summary

While the basic functionality is implemented in all four implementations, there are some differences in advanced features like BPSec. While the differences in the supportet Extension Blocks should not be critical for interoperability, one has to take care of fragmentation. Since the imc URI scheme is only supported by ION it should not be used if interoperability with one of the other implementations is needed yet.

An Overview of the supported features is given in table 1. Table 2 lists which Extension Blocks are supported by each implementation.

	ION	**μPCN**	**pyDTN**	**libDTN**
Status reports	✓	✓	∼	✓
Fragmentation	✓	✓	✗	✗
URI schemes	dtn, ipn, imc	dtn, ipn	dtn, ipn	dtn, ipn
BIBE	✓	✗	✗	✗
BPSec	✓	✗	✗	✓
DTN IPND	✓	✗	started	✗
Malformed bun.	discarded by all implementations			
Transmission err.	retransmited	deleted	discarded	

Table 1: *Implemented features*

	Prev. Node	Bundle Age	Hop Count/Limit	Metadata
ION	✓	✓	✓	✓
μPCN	✗	∼	✓	✗
pyDTN	∼	∼	∼	✗
libDTN	✓	✓	✓	✗

Table 2: *Implemented extension blocks*

References

[BCP14] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. *BCP 14*, March 1997

[BIBE] S. Burleigh. Bundle-in-Bundle Encapsulation (draft-ietf-dtn-bibect-03). *Internet Draft*, February 2020

[BPSec] E. Birrane, K. McKeever. Bundle Protocol Security Specification (draft-ietf-dtn-bpsec-22). *Internet Draft*, March 2020

[BPv6] K. Scott and S. Burleigh. Bundle Protocol Specification. *RFC 5050*, November 2007

[BPv7] S. Burleigh, K. Fall and E. Birrane. Bundle Protocol Version 7 (draft-ietf-dtn-bpbis-24). *Internet Draft*, March 2020

[BPQ] S. Farrell, A. Lynch, D. Kutscher, A. Lindgren. Bundle Protocol Query Extension Block (draft-irtf-dtnrg-bpq-00). *Internet Draft*, May 2012

[BP-Changes] M. Beyer. Comparison of Bundle Protocol version 6 and 7. August 2019, beyerm.de

[DCCP] H. Kruse, S. Jero, S. Ostermann. Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP). *RFC 7122*, March 2014

[DTN] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss. Delay-Tolerant Networking Architecture. *RFC 4838*, April 2007

[IMC] S. Burleigh. CBHE-Compatible Bundle Multicast (draft-burleigh-dtnrg-imc-00). *Internet Draft*, May 2011

[ION] Jet Propulsion Laboratory, California Institute of Technology. Interplanetary Overlay Network (ION) Design and Operation. November 2018, included in the ION release

[IPND] D. Ellard, R. Altman, A. Gladd, D. Brown, R. in 't Velt. DTN IP Neighbor Discovery (draft-irtf-dtnrg-ipnd-03). *Internet Draft*, November 2015

[RFC 6257] S. Symington, S. Farrell, H. Weiss, P. Lovell. Bundle Security Protocol Specification. *RFC 6257*, May 2011

[RFC 6258] S. Symington. Delay-Tolerant Networking Metadata Extension Block. *RFC 6258*, May 2011

[RFC 6260] S. Burleigh. Compressed Bundle Header Encoding (CBHE). *RFC 6260*, May 2011

[SBSP] E. Birrane. Streamlined Bundle Security Protocol Specification (draft-irtf-dtnrg-sbsp-00). *Internet Draft*, July 2013

[SNW] Spyropoulos, Thrasyvoulos and Psounis, Konstantinos and Raghavendra, Cauligi S. Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks. *Association for Computing Machinery*, 2005 `https://doi.org/10.1145/1080139.1080143`