RFC 9171 : Bundle Protocol Version 7

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 février 2022

Date de publication du RFC : Janvier 2022

https://www.bortzmeyer.org/9171.html

Après plusieurs itérations, le protocole Bundle, qui sert à transmettre des messages dans des situations où la connectivité est intermittente et la latence <https://www.bortzmeyer.org/latence.html> importante (par exemple dans l'espace), a atteint sa version 7, la première qui soit officiellement norme de l'Internet. Ce nouveau RFC décrit cette version 7, assez modifiée par rapport à la version 6 qui faisait l'objet du RFC 5050 [1]. Les deux versions n'interopèrent pas.

« Dans l'espace, personne ne vous entend crier », disait l'annonce du film Alien. Peut-être, mais on veut quand même communiquer, échanger des messages et des fichiers. Les particularités de cet environnement (et de quelques autres analogues comme par exemple des étendues peu connectées sur Terre ou bien des situations post-catastrophe) ont suscité la création du groupe DTN <https://datatracker.ietf.org/wg/dtn/> ("*Delay-Tolerant Networking*"), qui travaille à normaliser des techniques pour envoyer une photo de chat à un autre vaisseau spatial. La version 6 du protocole Bundle, spécifié dans le RFC 5050, avait été programmée dans de nombreux langages, et l'expérience acquise avec Bundle v6 a mené cette version 7. Dans les environnements concernés, avec leur latence <https://www.bortzmeyer.org/latence.html> élevée (et parfois très variable), leur taux de pertes de paquets souvent important, et leur connectivité intermittente, les protocoles de transport comme TCP et les protocoles applicatifs qui les accompagnent ne conviennent pas. Il n'est pas possible de faire du synchrone (on ne peut pas rester à attendre l'autre pendant on ne sait pas combien de temps). D'où la nécessité de protocoles spécifiques, architecturés autour de l'idée de « enregistre et fais suivre » ("*store and forward*"), où chaque machine intermédiaire va stocker les messages en attendant une occasion de les livrer. (Oui, ça ressemble à UUCP et c'est logique, les cahiers des charges étant analogues.) Notre RFC résume ce qu'on attend d'un tel protocole :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5050.txt

— Capacité à utiliser des déplacements physiques des données (en gros, la clé USB dans un camion, mécanisme dont on sait qu'il a une latence <https://www.bortzmeyer.org/latence.html> désastreuse mais une énorme capacité <https://www.bortzmeyer.org/capacite.html>; cf. l'anecdote amusante de l'Allemand et son cheval <https://korii.slate.fr/et-caetera/technologie-cheval-contre-internet-idee-genie-allemand-frustre-debits-lenteur-re ou bien l'offre SnowMobile d'Amazon <https://aws.amazon.com/fr/snowmobile/>).
— Capacité d'un envoyeur à oublier, une fois qu'il a transmis le message au nœud suivant (alors que TCP doit garder les données, en attendant l'accusé de réception de la destination finale).
— Capacité à gérer les connectivités intermittentes, voire les cas où émetteur et récepteur ne sont jamais joignables en même temps.
— Capacité à utiliser les cas où la connectivité arrive à un moment prédit à l'avance, aussi bien que ceux où il faut être opportuniste, utilisant la connectivité quand elle arrive.

Des détails sur ce cahier des charges sont dans les RFC 4838 et dans l'article de K. Fall à SIGCOMM « "*A delay-tolerant network architecture for challenged internets*" <https://conferences.sigcomm.org/sigcomm/2003/papers.html#p27-fall> ».

La section 3 du RFC rappelle les termes utilisés, notamment :

— "*Bundle*" (j'ai été trop paresseux pour traduire ce terme mais Bertand Petit suggère « ballot ») : une unité de données transmises. Un message (tel que compris par l'utilisateur humain) va être découpé en "*bundles*" qui seront transmis sur le réseau.
— BPA ("*Bundle Protocol Agent*") : le logiciel qui, sur un nœud, met en œuvre le protocole décrit dans ce RFC.
— CLA ("*Convergence Layer Adapter*") : la partie du protocole qui parle aux couches basses, pour effectuer l'envoi et la réception. Cela peut, par exemple, être TCP (RFC 9174).
— Nœud ("*node*") : une machine qui peut envoyer et recevoir des "*bundles*" mais aussi relayer les "*bundles*" des autres. Chaque nœud a un identificateur, le "*node ID*", qui est un URI (RFC 3986), en général de plan `dtn` (comme `dtn://example.com/foobar`) ou, de plus bas niveau et de signification seulement locale, `ipn`, qui avait été créé par le RFC 6260 (comme `ipn:9.37`). Les plans utilisés ont été mis dans le registre IANA <https://www.iana.org/assignments/uri-schemes/uri-schemes.xml#uri-schemes-1>.
— Transmission ("*Forwarding*") : faire suivre le message au nœud suivant (tout ce protocole est « saut par saut », avec des nœuds qui se refilent les "*bundles*").
— Terminal ("*endpoint*") : un ou plusieurs nœuds qui mettent en œuvre un service. Une sorte de nœud virtuel, quoi. Comme les nœuds, ils ont un identificateur qui est un URI.

La section 4 du RFC décrit le format des "*bundles*". Ce sera du CBOR (RFC 8949). Un "*bundle*" comprend un certain nombre de booléens qui indiquent si le récepteur doit accuser réception de l'arrivée du "*bundle*", de la transmission au nœud suivant, etc. (Leur liste figure dans un registre IANA <https://www.iana.org/assignments/bundle/bundle.xml#processing-control>.) Il indique également la destination (sous le forme d'un "*endpoint ID*") et la source (facultative).

Un "*bundle*" est en fait composé de blocs, chaque bloc étant un tableau CBOR. Le premier bloc porte les métadonnées indispensables pour les nœuds qui relaieront le "*bundle*" (comme l'identificateur de destination) ou comme la durée de vie du "*bundle*" (analogue au TTL d'IP). Chaque bloc contient un type et des données. La charge utile du "*bundle*" est dans un bloc de type 1, les autres types servant à divers détails (qui, dans IP, seraient mis dans les options du paquet en IPv4 et dans un en-tête d'extension en IPv6). Par exemple, un bloc de type 7 sert à transporter l'âge du "*bundle*", alors qu'un bloc de type 10 à indiquer le nombre de sauts maximum et le nombre déjà effectués. Les différents types figurent dans un registre IANA <https://www.iana.org/assignments/bundle/bundle.xml#block-types>.

La section 5 du RFC précise comment on traite les "*bundles*". Lors de la réception d'un "*bundle*", le nœud qui n'est pas destination (dont l'identificateur n'apparait pas dans le champ de destination) doit décider s'il transmet le "*bundle*" ou pas. S'il décide de le transmettre, il doit ensuite regarder s'il « connait » le nœud de destination (s'il sait le joindre directement) ou bien s'il doit le transmettre à un nœud mieux placé. Comment trouve-t-on ces nœuds mieux placés ? De même que le RFC sur IP ne

décrit pas les protocoles de routage, le "*Bundle Protocol*" ne précise pas. Cela est fait par des protocoles externes comme SABR <https://public.ccsds.org/Pubs/734x3b1.pdf>.

Une fois prise la décision d'envoyer le "*bundle*" à un autre nœud, on sélectionne un CLA adapté à cet autre nœud et on lui transmet. (Puis on le supprime du stockage local.) Bref, tout cela ressemble beaucoup au traitement d'un paquet IP par un routeur IP classique, avec le CLA jouant le rôle de la couche 2. La plus grosse différence est sans doute que le nœud peut garder le paquet pendant un temps assez long, s'il ne peut pas le transmettre tout de suite. La gestion des files d'attentes et des retransmissions sera donc assez différente de celle d'un routeur IP.

En cas d'échec, le nœud peut décider de renvoyer le "*bundle*" à l'expéditeur ou de le jeter. Si le "*bundle*" a expiré (durée de vie dépassée), il est jeté.

À la réception d'un "*bundle*", le nœud regarde aussi si ce "*bundle*" nécessitait un accusé de réception et, si nécessaire, il en envoie. Si le "*bundle*" a une destination locale, il est livré à la machine. (Le "*bundle*" a pu être fragmenté et, dans ce cas, il faudra attendre que tous les fragments soient là, pour le réassemblage, cf. section 5.8).

La section 6 du RFC couvre le cas des enregistrements administratifs, des messages (envoyés évidemment sous forme de "*bundles*") qui servent à gérer le bon fonctionnement du protocole (un peu comme ICMP pour IP). Ces enregistrements sont un tableau CBOR de deux entrées, un type (un nombre) et les données, un seul type d'enregistrement administratif existe actuellement, de type 1 <https://www.iana.org/assignments/bundle/bundle.xml#admin-record-types>, le rapport d'état, qui va servir à indiquer ce que le message est devenu. Son format et son contenu sont détaillés dans la section 6.1.1. Il dispose d'une liste de codes <https://www.iana.org/assignments/bundle/bundle.xml#status-reason>, comme 1 pour « durée de vie dépassée » ou 5 pour « je n'arrive pas à joindre la destination ».

On a vu que BP, le Bundle Protocol, dépend d'un CLA ("*Convergence Layer Adapter*") pour parler aux couches basses. La section 7 précise ce qu'on attend de ce "*convergence layer*" :

— Il doit évidemment permettre d'envoyer un "*bundle*" d'un nœud à l'autre,
— il doit rendre des comptes au BP, lui disant s'il a pu transmettre le "*bundle*" ou pas,
— il doit pouvoir recevoir des "*bundles*" et les livrer localement.
— Certaines extensions peuvent ajouter des exigences au CLA.

Le CLA doit faire tout cela tout en respectant le réseau sous-jacent, par exemple en évitant la congestion. Notez que dans la plupart des cas d'usage de BP, la latence <https://www.bortzmeyer.org/latence.html> est telle qu'on ne peut pas compter sur les réactions du récepteur, et il faut donc se débrouiller autrement. Un exemple d'un CLA est celui avec TCP du RFC 9174 (mais dans la plupart des scénarios envisagés pour le Bundle Protocol, TCP ne sera pas utilisable).

Il y a actuellement pas moins de sept mises en œuvres différentes de ce BP (Bundle Protocol) :

— μPCN <https://upcn.eu/>. Écrit en C, pour le système d'exploitation FreeRTOS. Libre. Il semble que ce soit la première mise en œuvre de BP qui intègre la version de notre RFC, la 7.
— μD3TN <https://gitlab.com/d3tn/ud3tn>. Écrit en Python. Licence libre.
— PyDTN (développé par X-works <https://x-works.sk/>). Écrit en Python. Il ne semble pas publiquement disponible <https://git.ifne.eu/space-public/pyDTN>.
— LibDTN/Terra <https://github.com/RightMesh/Terra/>, bibliothèque et applications. En Java.
— dtn7-go <https://github.com/dtn7/dtn7-go> est écrit en Go. Licence libre. Met en œuvre la version 7 de BP.
— dtn7-rs <https://github.com/dtn7/dtn7-rs/> est en Rust, également sous une licence libre, et gère également la version 7. Il vise les machines contraintes.

— Bien sûr, vu le plus spectaculaire cas d'usage du DTN, l'espace, la NASA a également développé du logiciel <https://www.nasa.gov/directorates/heo/scan/engineering/technology/disruption_tolerant_networking_software_options>. À propos de la NASA, notez qu'ils produisent beaucoup de documentation sur le DTN <https://www.nasa.gov/directorates/heo/scan/engineering/technology/disruption_tolerant_networking_resources>.

Notez que deux mises en œuvre de BP ne peuvent interopérer que si elles utilisent le même CLA.

Par défaut, BP n'offre aucune sécurité (cf. section 8). Un attaquant peut suivre le trafic, le modifier, etc. Certes, notre RFC recommande un CLA « sûr » mais cela ne suffit pas. La sécurité doit être fournie par des extensions, comme celle du RFC 9172. Notons aussi que le protocole utilise à plusieurs endroits des estampilles temporelles, donc sa sécurité dépend de celle de son horloge.

L'annexe A détaille les changements depuis le RFC 5050, qui normalisait la version 6 du protocole. Outre le changemente de statut (BP est désormais une norme et plus une « expérience »), les changements techniques sont importants à tel point qu'il n'y a pas d'interopérabilité entre les deux versions. Entre autres, les concepts de "*node ID*" et "*endpoint ID*" sont désormais séparés, le premier bloc d'un "*bundle*" est désormais immuable, de nouveaux types de blocs ont été créés, l'encodage abandonne l'ancien SDNV (RFC 6256) pour CBOR, etc. (Notez qu'il existe un registre IANA des versions de BP <https://www.iana.org/assignments/bundle/bundle.xml#primary-version>.)

Un des avantages de CBOR est l'existence du langage de description CDDL (RFC 8610), ce qui permet de donner une description formelle des "*bundles*" dans l'annexe B. Par exemple, voici la description du premier bloc de chaque "*bundle*", en CDDL (`eid` étant l'"*Endpoint ID*") :

```
primary-block = [
     version: 7,
     bundle-control-flags,
     crc-type,
     destination: eid,
     source-node: eid,
     report-to: eid,
     creation-timestamp,
     lifetime: uint,
     ...
     ? crc-value,
   ]
```

Avec un cahier des charges présentant beaucoup de ressemblances avec celui du BP, notez l'existence du système NNCP <http://www.nncpgo.org/>, qui est également de type « enregistre et fais suivre ». Ça semble très intéressant mais je ne l'ai jamais testé.