RFC 8352 : Energy-Efficient Features of Internet of Things Protocols

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 avril 2018

Date de publication du RFC : Avril 2018

https://www.bortzmeyer.org/8352.html

Les objets contraints, engins connectés ayant peu de capacités (un capteur de température dans la nature, par exemple), ont en général une réserve d'énergie électrique très limitée, fournie par des piles ou batteries à la capacité réduite. L'usage des protocoles de communication traditionnels, souvent très bavards, épuiserait rapidement ces réserves. Ce RFC étudie les mécanismes utilisables pour limiter la consommation électrique, et faire ainsi durer ces objets plus longtemps.

La plupart des protocoles de couche 2 utilisés pour ces « objets contraints » disposent de fonctions pour diminuer la consommation électrique. Notre RFC décrit ces fonctions, et explique comment les protocoles de couches supérieures, comme IP peuvent en tirer profit. Si vous ne connaissez pas le monde des objets contraints, et les problèmes posés par leur connexion à l'Internet, je recommande fortement la lecture des RFC 6574 [1], RFC 7228, et RFC 7102. Le but des techniques présentées dans ce RFC est bien de faire durer plus longtemps la batterie, ce n'est pas un but écologique. Si l'objet est alimenté en permanence (une télévision connectée, ou bien un grille-pain connecté), ce n'est pas considéré comme problème.

Des tas de travaux ont déjà eu lieu sur ce problème de la diminution de la consommation électrique de ces objets. Il y a eu moins d'efforts sur celle des protocoles réseau, et ce sont ces efforts que résume ce RFC. Les protocoles traditionnels étaient conçus pour des machines alimentées en permanence et pour qui la consommation électrique n'était pas un problème. Diffuser très fréquemment une information, même inutile, n'était pas perçu comme du gaspillage, contrairement à ce qui se passe avec l'Internet des Objets. (Voir le RFC 7772 pour une solution à un tel problème.)

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc6574.txt

L'IETF a déjà développé un certain nombre de protocoles qui sont spécifiques à cet « Internet des Objets ». Ce sont par exemple 6LoWPAN (RFC 6282, RFC 6775 et RFC 4944), ou bien le protocole de routage RPL (RFC 6550) ou encore l'alternative à HTTP CoAP (RFC 7252). En gros, l'application fait tourner du CoAP sur IPv6 qui est lui-même au-dessus de 6LoWPAN, couche d'adaptation d'IP à des protocoles comme IEEE 802.15.4. Mais l'angle « économiser l'énergie » n'était pas toujours clairement mis en avant, ce que corrige notre nouveau RFC.

Qu'est ce qui coûte cher à un engin connecté (cher en terme d'énergie, la ressource rare) ? En général, faire travailler le CPU est bien moins coûteux que de faire travailler la radio, ce qui justifie une compression énergique. Et, question réseau, la réception n'est pas forcément moins coûteuse que la transmission. L'étude Powertrace <http://soda.swedishict.se/4112/> est une bonne source de mesures dans ce domaine mais on peut également lire « "*Measuring Power Consumption of CC2530 With Z-Stack*" <http://www.ti.com/lit/an/swra292/swra292.pdf> ». Le RFC contient aussi (section 2) des chiffres typiques, mesurés sur Contiki et publiés dans « "*The ContikiMAC Radio Duty Cycling Protocol*" <http://soda.swedishict.se/5128/> » :

— Écouter pendant une seconde : 63 000 [Caractère Unicode non montré [2]]J,
— Réception d'un paquet : 178 [Caractère Unicode non montré]J pour un paquet diffusé et 222 [Caractère Unicode non montré]J autrement,
— Transmission d'un paquet : de 120 à 1 790 [Caractère Unicode non montré]J selon les cas.

Une technique courante quand on veut réduire la consommation électrique lors de la transmission est de réduire la puissance d'émission. Mais cela réduit la portée, donc peut nécessiter d'introduire du routage, ce qui augmenterait la consommation.

On a vu que la simple écoute pouvait consommer beaucoup d'énergie. Il faut donc trouver des techniques pour qu'on puisse couper l'écoute (éteindre complètement la radio), tout en gardant la possibilité de parler à l'objet si nécessaire. Sans ce "*duty-cycling*" (couper la partie radio de l'objet, cf. RFC 7228, section 4.3), la batterie ne durerait que quelques jours, voire quelques heures, alors que certains objets doivent pouvoir fonctionner sans entretien pendant des années. Comment couper la réception tout en étant capable de recevoir des communications ? Il existe trois grandes techniques de RDC ("*Radio Duty-Cycling*", section 3 de notre RFC) :

— Échantillonage : on n'est pas en réception la plupart du temps mais on écoute le canal à intervalles réguliers. Les objets qui veulent me parler doivent donc émettre un signal pendant une durée plus longue que la période d'échantillonage.
— Transmissions aux moments prévus. On annonce les moments où on est prêt à recevoir. Cela veut dire que les objets qui veulent parler avec moi doivent le faire au bon moment.
— Écouter quand on transmet. Dans ce cas, ceux qui ont envie de me parler doivent attendre que j'émette d'abord.

Dans les trois cas, la latence <https://www.bortzmeyer.org/latence.html> va évidemment en souffrir. On doit faire un compromis entre réduire la consommation électrique et augmenter la latence. On va également diminuer la capacité <https://www.bortzmeyer.org/capacite.html> du canal puisqu'il ne pourra pas être utilisé en permanence. Ce n'est pas forcément trop grave, un capteur a peu à transmettre, en général.

Il existe plein de méthodes pour gagner quelques [Caractère Unicode non montré]J, comme le regroupement de plusieurs paquets en un seul (il y a un coût fixe par paquet, indépendamment de leur taille). Normalement, le RDC est quelque part dans la couche 2 et les protocoles IETF, situés plus haut, ne devraient pas avoir à s'en soucier. Mais une réduction sérieuse de la consommation électrique nécessite que tout le monde participe, et que les couches hautes fassent un effort, et connaissent ce que font les couches basses, pour s'adapter.

2. Car trop difficile à faire afficher par LaTeX

La section 3.6 de notre RFC détaille quelques services que fournissent les protocoles de couche 2 de la famille IEEE 802.11. Une station (un objet ou un ordinateur) peut indiquer au point d'accès (AP, pour "*access point*") qu'il va s'endormir. L'AP peut alors mémoriser les trames qui lui étaient destinées, pour les envoyer plus tard. IEEE 802.11v va plus loin en incluant des mécanismes comme le proxy ARP (répondre aux requêtes ARP à la place de l'objet endormi).

Bluetooth a un ensemble de services pour la faible consommation, nommés Bluetooth LE (pour "*Low Energy*"). 6LoWPAN peut d'ailleurs en tirer profit (cf. RFC 7668).

IEEE 802.15.4 a aussi des solutions. Il permet par exemple d'avoir deux types de réseaux, avec ou sans annonces périodiques ("*beacons*"). Dans un réseau avec annonces, des machines nommés les coordinateurs envoient régulièrement des annonces qui indiquent quand on peut émettre et quand on ne peut pas, ces dernières périodes étant le bon moment pour s'endormir afin de diminuer la consommation : on est sûr de ne rien rater. Les durées de ces périodes sont configurables par l'administrateur réseaux. Dans les réseaux sans annonces, les différentes machines connectées n'ont pas d'informations et transmettent quand elles veulent, en suivant CSMA/CA.

Enfin, DECT a aussi un mode à basse consommation, DECT ULE. Elle est également utilisable avec 6LoWPAN (RFC 8105). DECT est très asymétrique, il y a le FP ("*Fixed Part*", la base), et le PP ("*Portable Part*", l'objet). DECT ULE permet au FP de prévenir quand il parlera (et le PP peut dormir jusque là). À noter que si la « sieste » est trop longue (plus de dix secondes), le PP devra refaire une séquence de synchronisation avec le FP, qui peut être coûteuse en énergie. Il faut donc bien calculer son coup : si on s'endort souvent pour des périodes d'un peu plus de dix secondes, le bilan global sera sans doute négatif.

La section 4 de notre RFC couvre justement ce que devrait faire IP, en supposant qu'on utilisera 6LoWPAN. Il y a trois services importants de 6LoWPAN pour la consommation électrique :

— 6LoWPAN a un mécanisme de fragmentation et de réassemblage. IPv6 impose que les liens aient une MTU d'au moins 1 280 octets. Si ce n'est pas le cas, il faut un mécanisme de fragmentation sous IP. Mais la fragmentation n'est pas gratuite et il vaut mieux que les applications s'abstiennent de faire des paquets trop gros (un cas que traite CoAP, dans le RFC 7959).
— 6LoWPAN sait que le processeur consomme moins d'énergie que la partie radio de l'objet connecté, et qu'il est donc rentable de faire des calculs qui diminuent la quantité de données à envoyer, notamment la compression. Par exemple, 6LoWPAN permet de diminuer sérieusement la taille de l'en-tête de paquet IPv6 (normalement 40 octets), en comptant sur le fait que les paquets successifs d'un même flot se ressemblent.
— 6LoWPAN a un mécanisme de découverte du voisin optimisé pour les réseaux contraints, nommé 6LoWPAN-ND.

Il y a aussi des optimisations qui ne changent pas le protocole. Par exemple, Contiki ne suit pas, dans sa mise en œuvre, une stricte séparation des couches, afin de pouvoir optimiser l'activité réseau.

De même, les protocoles de routage doivent tenir compte des contraintes de consommation électrique (section 5 du RFC). Le protocole « officiel » de l'IETF pour le routage dans les réseaux contraints est RPL (RFC 6550). L'étude Powertrace <http://soda.swedishict.se/4112/> déjà citée étudie également la consommation de RPL et montre qu'il est en effet assez efficace, si le réseau est stable (s'il ne l'est pas, on consommera évidemment du courant à envoyer et recevoir les mises à jours des routes). On peut adapter les paramètres de RPL, via l'algorithme Trickle (RFC 6206, rien à voir avec le protocole de transfert de fichiers de BitNet), pour le rendre encore plus économe, mais au prix d'une convergence plus lente lorsque le réseau change.

À noter que le RFC ne parle que de RPL, et pas des autres protocoles de routage utilisables par les objets, comme Babel (RFC 8966).

Et les applications ? La section 6 du RFC leur donne du boulot, à elles aussi. Bien sûr, il est recommandé d'utiliser CoAP (RFC 7252) plutôt que HTTP, entre autre en raison de son en-tête plus court, et de taille fixe. D'autre part, CoAP a un mode de pure observation, où le client indique son intérêt pour certaines ressources, que le serveur lui enverra automatiquement par la suite, lorsqu'elles changeront, économisant ainsi une requête. Comme HTTP, CoAP peut utiliser des relais qui mémorisent la ressource demandée, ce qui permet de récupérer des ressources sans réveiller le serveur, si l'information était dans le cache. D'autres protocoles étaient à l'étude, reprenant les principes de CoAP, pour mieux gérer les serveurs endormis. (Mais la plupart de ces projets semblent. . .endormis à l'heure actuelle.)

Enfin, la section 7 de notre RFC résume les points importants :

— Il ne faut pas hésiter à violer le modèle en couches, en accédant à des informations des couches basses, elles peuvent sérieusement aider à faire des économies.
— Il faut comprimer.
— Il faut se méfier de la diffusion, qui consomme davantage.
— Et il faut s'endormir souvent, pour économiser l'énergie (la méthode Gaston Lagaffe, même si le RFC ne cite pas ce pionnier).