RFC 6971 : Depth-First Forwarding in Unreliable Networks (DFF)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 29 juin 2013

Date de publication du RFC : Juin 2013

https://www.bortzmeyer.org/6971.html

Traditionnellement, l'acheminement à bon port d'un paquet IP nécessitait deux processus distincts : le **routage** ("*routing*") à proprement parler, où les routeurs calculent des tables de routage indiquant, pour divers préfixes IP, la prochaine étape ("*next hop*") à atteindre, et la **transmission** ("*forwarding*") où les routeurs choisissent le prochain routeur, et l'interface de sortie, pour un paquet donné. Le premier processus, le routage, est fait à l'avance, indépendemment d'un paquet précis, et nécessite des protocoles complexes comme OSPF. Le second processus, la transmission, nécessite de faire tourner l'algorithme de plus long préfixe, et se fait par contre en temps réel, pour chaque paquet entrant. Pourquoi cette séparation en deux ? Car ces deux processus ont des caractéristiques très différentes. Le premier, nécessitant des protocoles et des algorithmes élaborés, est mieux réalisé sur un processeur classique. Le second est typiquement le domaine d'ASIC spécialisés. Les routeurs haut de gamme utilisent d'ailleurs les deux types de matériel, selon la tâche. Mais cette séparation a aussi des inconvénients. Ainsi, la détection qu'un lien ne marche plus est typiquement du ressort des algorithmes de routage, qui ajusteront alors leurs routes. Pourtant, certains réseaux physiques permettent la détection, lors de la transmission d'un paquet, d'une panne et d'une impossibilité d'envoi. Pourquoi ne pas utiliser cette information pour que les paquets suivants aillent essayer une autre étape suivante ? C'est ce que propose ce RFC.

Notez que, comme souvent en matière de réseaux informatiques, le vocabulaire est flou et incohérent. Ainsi, le terme de routage est souvent utilisé pour désigner les deux processus qui concourent à l'acheminement, le routage à proprement parler et la transmission. Par exemple, j'ai parlé de table de routage alors que le sigle FIB ("*Forwarding Information Base*") aurait été plus correct. Et le terme routeur désigne une boîte qui fait de la transmission et parfois (sur le moyen et haut de gamme) le routage. On lit ainsi des phrases fausses comme « Quagga permet de transformer un PC Unix en routeur » alors que le noyau Unix a toujours été capable de faire de la transmission de paquets (Quagga met en œuvre les protocoles de routage, qui ne sont **pas** nécessaires à un routeur).

Mais passons. Il faut bien faire avec le vocabulaire que nous avons. Revenons à la proposition expérimentale de notre RFC 6971 [1]. Elle ne cherche pas à couvrir tous les cas de transmission de paquets. Ce RFC s'applique aux paquets IPv6, lors de leur transmission sur des liens peu fiables, avec une topologie sans cesse changeante, un trafic assez faible, et où le protocole de couche 2 permet de savoir si un paquet a été transmis avec succès ou pas. C'est typiquement le monde des MANET et des LowPAN, des réseaux de petites machines connectées en radio. L'idée centrale de ce protocole DFF est d'avoir, dans le processus de transmission, certaines techniques utilisées d'habitude uniquement dans le routage, pour chercher un chemin alternatif. DFF va transmettre un paquet au premier routeur, s'il ne reçoit pas de nouvelles, il va essayer le second et ainsi de suite. Par contre, si la transmission vers la première étape se fait avec succès, DFF va laisser le paquet continuer, passer au routeur suivant, etc. Il fonctionne donc en **profondeur d'abord** (on essaie d'aller le plus loin possible, avant de tester d'autres « frères »). Les essais successifs se font séquentiellement (et non pas en parallèle comme dans les protocoles d'inondation) pour éviter qu'un destinataire ne reçoive plusieurs copies du même message. Cela peut donc prendre pas mal de temps s'il y a beaucoup de routeurs potentiels. À noter que le tout fonctionne de manière répartie, sans composant central.

N'est-ce pas un peu bizarre que de refaire une partie du travail du processus de routage dans celui de transmission ? Mais c'est lié à la façon dont fonctionnent les protocoles de routage. Travaillant à l'avance, ils ne peuvent pas s'adapter assez vite à tous les changements de topologie, qu'on ne découvrira qu'en envoyant réellement des paquets. Les protocoles de routage fonctionnent typiquement en envoyant de temps en temps des messages d'information. L'information peut donc être dépassée au moment de la transmission. Pour avoir une information plus à jour (section 1.1), il faudrait augmenter le rythme des messages d'information et cela se ferait au détriment de la consommation d'énergie des machines (rappelez-vous que DFF est prévu pour des réseaux de petites machines, souvent sans autre source d'énergie que leur batterie). Se placer sur le dos du processus de transmission (qui doit avoir lieu de toute façon) permettra donc, si ce protocole DFF tient ses promesses (ce RFC est expérimental), d'augmenter le pourcentage de paquets effectivement transmis, sans augmenter les coûts.

DFF va donc récolter de l'information sur le fonctionnement effectif du réseau, information qui peut être utile au processus de routage. DFF peut donc la réinjecter dans ce dernier, pour l'aider à calculer de meilleures routes, mais ce point n'est pas détaillé dans ce RFC.

La section 3 du RFC revient sur l'applicabilité limitée de ce protocole DFF. On l'a dit, il ne s'agit pas de remplacer tous les mécanismes de routage existants par ce mécanisme utilisant le succès de la transmission comme indication. D'abord, tous les réseaux n'offrent pas de possibilité de détecter cet échec. 802.15.4 ou 802.11 le permettent mais pas Ethernet, par exemple. D'autre part, DFF n'est utile que si la topologie change souvent. Dans un réseau filaire stable et fiable, les mécanismes de routage actuels marchent très bien et n'ont pas besoin de DFF. Et DFF nécessite un réseau ayant une topologie dense, avec plusieurs liens possibles entre deux points. S'il n'existe qu'un seul lien, DFF ne servira à rien puisque, en cas de perte d'un paquet, il n'y aura pas d'alternative. Enfin, DFF, contrairement à la transmission IP traditionnelle, nécessite un **état** dans le routeur. Il faut se souvenir de chaque paquet envoyé, le temps qu'on soit sûr de son bon acheminement. DFF ne peut donc être utilisé qu'en cas de faible trafic.

DFF peut fonctionner aussi bien dans les réseaux "*route-over*" (réseaux routés par les méthodes IP classiques) que dans les réseaux "*mesh-under*" (réseaux utilisant un mécanisme de « routage » dans la couche 2, cette terminologie est empruntée au RFC 6775).

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc6971.txt`

La section 4 résume le fonctionnement du protocole : pour chaque paquet qu'il a reçu et qu'il doit transmettre, le routeur construit une liste des prochaines étapes possibles. Il tente d'envoyer le paquet en essayant successivement ces étapes. Si toutes échouent, le routeur renvoie le paquet à l'expéditeur, qui va alors essayer l'étape suivante dans sa liste à lui (rappelez-vous qu'on est en profondeur d'abord). On voit donc que, pour chaque paquet pas encore transmis avec succès, le routeur doit garder cette information en mémoire : routeur précédent (qui a envoyé le paquet), numéro et prochaines étapes possibles en indiquant lesquelles ont déjà été essayées. C'est quoi, ce numéro ? C'est un identifiant unique par routeur et permettant de distinguer un paquet des autres (section 12). La liste des paquets en cours de transmission, avec les métadonnées, est nommée le "*Processed Set*" (section 6.2).

DFF doit connaître la liste des prochaines étapes possibles. Il a pu utiliser un protocole de découverte des voisins (comme le NHDP du RFC 6130), ou bien il a pu trouver cette information dans la RIB ("*Routing Information Base*"), la base de données construite par le processus de routage (voir la section 5 pour les détails).

Les informations dont a besoin DFF peuvent être représentées de deux manières différentes dans le paquet. En "*route-over*", on utilise un en-tête d'extension IPv6 standard, le "*hop-by-hop options*" (et pas un "*destination options*", malgré les recommandations du RFC 6564 car cet en-tête doit être examiné par tous les routeurs sur le trajet). Le type de l'option DFF est 0xEE. Comme il commence par deux bits à Un, le paquet sera rejeté par les routeurs non-DFF (RFC 2460, section 4.2). La section 13.1 de notre RFC fournit les détails sur cette adaptation à IPv6. En "*mesh-under*", on se sert des en-têtes 6LoWPAN, avec les adresses du RFC 4944 (section 13.2 pour les précisions). Outre le numéro, il faut mettre dans chaque paquet deux booléens : `RET` indique un paquet renvoyé à l'expéditeur car non distribuable, et `DUP` indique une retransmission, au moins un essai ayant apparemment échoué. Comme le paquet a pu être distribué quand même (même si le routeur émetteur n'a pas été notifié), `DUP` sert à indiquer la possibilité de duplication d'un paquet, avec laquelle les couches hautes vont devoir se débrouiller.

Rien ne garantit que le réseau soit un arbre pur. Des boucles sont parfaitement possibles et DFF doit les détecter. Ces booléens `RET` et `DUP` permettent notamment de distinguer un paquet délibérement renvoyé à l'émetteur d'un paquet qui boucle dans le réseau. Si le routeur reçoit un paquet qu'il a déjà transmis (il le sait d'après le numéro), avec `DUP` et `RET` tous les deux à Zéro, c'est que le paquet est en train de boucler. Si `RET` est à Un, c'est un retour délibéré car le routeur suivant n'a pas pu transmettre le paquet.

Le format exact des paquets figure en section 7. Le traitement des paquets par les routeurs est détaillé en sections 10 et 11 du RFC. Des examples de fonctionnement de DFF dans diverses topologies, et pour divers cas de figure, se trouvent dans l'annexe A.

Notez que, comme la plupart des protocoles conçus pour ce type de réseau (sans administrateur réseaux, et où il y a beaucoup de machines, qu'il n'est pas question de configurer une par une), DFF ne présente guère de sécurité. Par exemple, en envoyant beaucoup de paquets à une adresse inexistante, on force les routeurs à retenir ces paquets en mémoire le temps de détecter la non-délivrabilité, réalisant ainsi facilement une attaque par déni de service (section 16.3.1).

Maintenant, il reste à tester DFF en vrai, et à récolter de l'information sur les points détaillés en section 1.2 : quelles sont les bonnes valeurs pour le délai d'attente `P_HOLD_TIME` (trop long, et on consomme de la mémoire sur les routeurs, qui doivent se souvenir du paquet, trop court, on risque d'oublier un paquet et de le croire nouveau lorsqu'il revient après un échec, cf. section 8) ? Et les valeurs idéales pour le nombre de sauts maximum `MAX_HOP_LIMIT` ? Et l'interaction idéale entre DFF et les protocoles de routage ? Tous ces points vont devoir être étudiés en pratique.

L'annexe B du RFC rassemble un certain nombre d'informations sur le déploiement effectif de DFF et ses mises en œuvre (au moins deux à l'heure actuelle, une sur 802.11 et une sur 802.15.4). Par exemple, les compteurs électriques de la "*smart grid*" au Japon utilisent apparemment DFF mais les détails n'ont pas été publiés. Un autre distributeur d'électricité qui utilise DFF est la Kit Carson Electric Cooperative <http://www.kitcarson.com/> au Nouveau-Mexique, qui en a <http://www.kitcarson.com/index.php?option=com_content&view=article&id=45&Itemid=1>fait état publiquement. Une mise en œuvre en logiciel libre de DFF en Java a été annoncée par Fujitsu mais apparemment pas encore publiée (en tout cas, je ne l'ai pas trouvée).

Merci à Laurent Toutain pour sa relecture et ses remarques pertinentes sur l'algorithme DFF.