Network Working Group
Request for Comments : 1058
Juin 1988

C. Hedrick
Rutgers University

Protocole d'Information de Routage (RIP)

Table des matières

Statut de ce document

Ce RFC décrit un protocole « existant » d'échange d'information de routage entre des passerelles et d'autres hôtes. Il est destiné à servir de base pour le développement de logiciels de passerelle pour la communauté Internet. La distribution de ce document est libre.
Cette traduction française a été effectuée par Frédéric Delanoy.
Vous pouvez le contacter ici.

Résumé

Ce document est destiné à faire les choses suivantes :

1. Documenter un protocole et des algorithmes qui sont actuellement largement utilisés pour le routage, mais qui n'ont jamais été formellement documentés.
2. Spécifier quelques améliorations aux algorithmes pour améliorer la stabilité des routes dans les grands réseaux. Ces améliorations n'introduisent aucune incompatibilité avec les implémentations existantes. Elles vont être incorporées dans toutes les implémentations de ce protocole.
3. Suggérer quelques fonctionnalités supplémentaires pour permettre une plus grande configurabilité et un plus grand contrôle. Ces fonctionnalités ont été spécialement développées pour résoudre des problèmes qui ont surgi dans l'utilisation réelle faite par la communauté NFSnet. Néanmoins, elles devraient avoir une utilité plus générale.

Le Protocole d'Information de Routage [1] (RIP) décrit ici est basé lâchement sur le programme « `routed` », distribué avec la Berkeley Software Distribution 4.3. Néanmoins, il existe plusieurs autres implémentations de ce qui est supposé être le même protocole. Malheureusement, ces différentes implémentations sont en désaccord dans divers points de détail. Les spécifications présentées ici représentent une combinaison des fonctionnalités empruntées à diverses implémentations. Nous croyons qu'un programme conçu en suivant ce document pourra interopérer avec `routed`, et avec toutes les autres implémentations de RIP dont nous avons connaissance.

Notez que cette description adopte une point de vue différent de celui de la plupart des implémentations existantes quant au moment où les métriques devraient être incrémentées. En procédant à un changement correspondant dans la métrique utilisée pour un réseau local, nous avons gardé la compatibilité avec d'autres implémentations existantes. Reportez-vous à la section 3.6 pour obtenir des détails sur ce sujet.

1 Introduction

Ce document décrit un protocole d'une série de protocoles de routage basés sur l'algorithme de Bellman-Ford (ou à vecteurs de distance). Cet algorithme a été utilisé pour des calculs de routage dans les réseaux informatiques depuis les débuts d'ARPANET. Les formats de paquets et le protocole particuliers décrits ici sont basés sur le programme « `routed` »,

[1] Routing Information Protocol

qui est inclus dans la distribution Berkeley de Unix. Il est devenu un standard *de facto* pour l'échange d'information de routage entre passerelles et hôtes. Il est implémenté dans ce but par la plupart des vendeurs commerciaux de passerelles IP. Notez néanmoins qu'un grand nombre de ces vendeurs ont leurs propres protocoles qui sont utilisés entre leurs passerelles.

Ce protocole est le plus utile comme « protocole interne à des passerelles » [2]. Dans un réseau à l'échelle d'une nation comme l'est Internet actuellement, il est très improbable qu'un unique protocole de routage soit utilisé dans l'entièreté du réseau. Le réseau sera plutôt organisé comme une collection de « systèmes autonomes » [3]. Un système autonome sera en général administré par une seule entité, ou disposera au moins d'un certain degré de contrôle technique et administratif. Chaque système autonome aura sa propre technologie de routage, qui peut être différente pour des systèmes autonomes distincts. Le protocole de routage utilisé à l'intérieur d'un système autonome est référencé en tant que protocole interne à des passerelles, ou « IGP ». Un protocole séparé est utilisé pour servir d'interface entre les systèmes autonomes. Le premier protocole de ce type, toujours utilisé sur Internet, est « EGP » (protocole externe à des passerelles [4]). De tels protocoles sont habituellement appelés protocoles de routage inter-AS. RIP a été conçu pour fonctionner avec des réseaux de taille modérée en utilisant une technologie raisonnablement homogène. Ainsi, il convient comme IGP pour beaucoup de campus et de réseaux régionaux utilisant des lignes série dont la vitesse ne varie pas grandement. Il n'est pas destiné à une utilisation dans des environnements plus complexes. Pour plus d'information sur les situations où RIP est supposé convenir, voyez Braden et Postel [3].

RIP fait partie d'une classe d'algorithmes connue sous le nom d'« algorithmes à vecteurs de distance ». La première description de cette classe d'algorithmes connue par l'auteur est présentée dans Ford et Fulkerson [6]. De ce fait, ils sont parfois connus sous le nom d'algorithmes de Ford-Fulkerson. Le terme Bellman-Ford est également utilisé. Il provient du fait que la formulation est basée sur l'équation de Bellman, la base de la « programmation dynamique ». (Pour une introduction standard dans ce domaine, voyez [1].) La présentation qui est en faite dans ce document est basée étroitement sur [2]. Ce texte contient une introduction aux mathématiques des algorithmes de routage. Il décrit et justifie plusieurs variantes de l'algorithme présenté ici, ainsi qu'un certain nombre d'autres algorithmes liés. Les algorithmes de base décrits dans ce protocole ont déjà été utilisés dans le routage informatique depuis 1969 dans ARPANET. Néanmoins, l'ancêtre spécifique de ce protocole se trouve dans les protocoles réseau de Xerox. Les protocoles PUP (voyez [4]) utilisaient le Gateway Information Protocol [5] pour échanger de l'information de routage. Une version quelque peu mise à jour de ce protocole a été adoptée pour l'architecture de systèmes de réseau Xerox (Xerox Network Systems, XNS), sous le nom de « Routing Information Protocol ». (Voyez [7].) Le `routed` de Berkeley est en grande partie identique au Routing Information Protocol, les adresses XNS étant remplacées par un format d'adresse plus général capable d'utiliser IP et d'autres types d'adresses, et où les mises à jour de routage sont limitées à une mise à jour toutes les 30 secondes. Du fait de cette similitude, le terme « Routing Information Protocol » (ou simplement RIP) est utilisé pour se référer à la fois au protocole XNS et au protocole utilisé par `routed`.

[2] Interior Gateway Protocol, IGP
[3] Autonomous Systems, AS
[4] Exterior Gateway Protocol
[5] Protocole d'information sur les passerelles

RIP est destiné à être utilisé à l'intérieur de l'Internet basé sur IP. Internet est organisé en un grand nombre de réseaux connectés par des passerelles. Les réseaux peuvent être soit des liaisons point-à-point, soit des réseaux plus complexes comme Ethernet ou ARPANET. Les hôtes et passerelles se voient remettre des datagrammes IP adressés à un hôte quelconque. Le routage est la méthode par laquelle l'hôte ou la passerelle décide de l'endroit où envoyer le datagramme. Il peut être possible d'envoyer le datagramme directement à la destination, si cette destination est située dans l'un des réseaux directement connectés à l'hôte ou la passerelle. Néanmoins, le cas intéressant se produit quand la destination n'est pas directement accessible. Dans ce cas, l'hôte ou la passerelle essaie d'envoyer le datagramme à une passerelle qui est plus proche de la destination. Le but d'un protocole de routage est très simple : fournir l'information nécessaire pour effectuer un routage.

1.1 Limitations du protocole

Ce protocole ne résout pas tous les problèmes de routage imaginables. Comme mentionné plus haut, il est principalement destiné à une utilisation en tant qu'IGP, dans des réseaux raisonnablement homogènes de taille modérée. De plus, les limitations spécifiques suivantes devraient être mentionnées :

- Le protocole est limité aux réseaux dont le plus long chemin implique 15 sauts[6]. Les concepteurs croient que la conception du protocole de base n'est pas appropriée pour les réseaux plus larges. Notez que cette affirmation suppose qu'un coût unitaire est associé à chaque réseau. C'est la façon dont RIP est normalement configuré. Si l'administrateur système choisit d'utiliser des coûts plus élevés, la limite supérieure de 15 peut facilement devenir un problème.
- Le protocole dépend du « comptage à l'infini » pour résoudre certaines situations inhabituelles. (Cela sera expliqué dans la section suivante.) Si le système de réseaux comporte plusieurs centaines de réseaux, et qu'une boucle de routage les impliquant tous survient, la résolution de la boucle requerrait soit beaucoup de temps (si la fréquence des mises à jour de routage est limitée), soit beaucoup de bande passante (si les mises à jour sont envoyées à chaque fois que des changements sont détectés). Une telle boucle consommerait une grande quantité de bande passante du réseau avant que la boucle ne soit corrigée. Nous croyons que, dans les cas réalistes, cela ne sera pas un problème sauf pour les lignes lentes (à bas débit). Même dans ce cas, le problème sera plutôt inhabituel, puisque différentes précautions sont prises, qui devraient éviter ces problèmes dans la plupart des cas.
- Ce protocole utilise des « métriques » fixes pour comparer des routes alternatives. Cela n'est pas approprié pour les situations où les routes doivent être choisies en fonction de paramètres temps-réel comme un délai, une fiabilité ou une charge mesurés. Les extensions évidentes permettant des métriques de ce type vont probablement introduire des instabilités que le protocole n'est pas censé pouvoir traiter.

1.2 Organisation de ce document

Le corps principal de ce document est organisé en deux parties, qui occupent les deux prochaines sections :

[6] en anglais, « hops »

section 2 Un développement conceptuel et une justifications des algorithmes à vecteurs de distance en général.

section 3 La description réelle du protocole.

Chacune de ces deux sections peut largement mériter un document à elle seule. La section 2 essaie de donner une présentation informelle des fondements mathématiques de l'algorithme. Notez que la présentation suit une méthode en « spirale ». Un algorithme initial, assez simple, est décrit. Ensuite, des raffinements y sont ajoutés dans les sections successives. La section 3 est la description réelle du protocole . À part en cas de référence spécifique à la section 2, il devrait être possible d'implémenter RIP entièrement à partir des spécifications fournies dans la section 3.

2 Algorithmes à vecteurs de distance

Le routage est la tâche consistant à trouver un chemin d'un émetteur à une destination souhaitée. Dans le « modèle IP Catenet », il se réduit essentiellement à trouver des passerelles entre des réseaux. Aussi longtemps qu'un message reste sur un réseau ou sous-réseau unique, tout problème de routage est résolu par une technologie qui est spécifique au réseau. Par exemple, Ethernet et ARPANET définissent tous deux un moyen par lequel tout émetteur peut parler à toute destination spécifiée à l'intérieur de ce propre réseau. Le routage IP entre en jeu essentiellement quand les messages doivent aller d'un émetteur sur un tel réseau vers une destination située sur un autre réseau. Dans ce cas, le message doit traverser des passerelles connectant les réseaux. Si les réseaux ne sont pas adjacents, le message peut traverser plusieurs réseaux intermédiaires, et les passerelles les connectant. Une fois que le message arrive sur une passerelle située sur le même réseau que la destination, la propre technologie de ce réseau est utilisée pour atteindre la destination.

Tout au long de cette section, le terme « réseau » est utilisé de façon générique pour couvrir un seul réseau à diffusion [7] (p.ex. Ethernet), une ligne point-à-point, ou ARPANET. Le point critique est qu'un réseau est traité comme une simple entité par IP. Soit aucun routage n'est nécessaire (comme pour une ligne point-à-point), soit ce routage est effectué d'une manière transparente pour IP, permettant à IP de traiter le réseau entier comme un système unique complètement connecté (comme pour un réseau Ethernet ou ARPANET). Notez que le terme « réseau » est utilisé d'une façon quelque peu différente dans les discussions concernant l'adressage IP. Un seul numéro de réseau IP peu être affecté à une collection de réseaux, où l'adressage de « sous-réseaux » est utilisé pour décrire les réseaux individuels. En fait, nous utilisons ici le terme « réseau » pour nous référer aux sous-réseaux dans le cas où un adressage des sous-réseaux est utilisé.

Un certain nombre d'approches différentes pour la découverte de routes entre réseaux sont possibles. Une manière utile de catégoriser ces approches est de se baser sur le type d'information que les passerelles doivent s'échanger afin d'être capables de trouver des routes. Les algorithmes à vecteurs de distance sont basés sur l'échange d'une petite quantité d'information. Chaque entité (passerelle ou hôte) qui participe au protocole de routage est supposée conserver de l'information sur toutes les destinations du système. Généralement, l'information concernant toutes les entités connectées au réseau est résumée par une seule

[7] broadcast

entité, qui décrit la route vers toutes les destinations de ce réseau. Ce résumé est possible car, en ce qui concerne IP, le routage à l'intérieur d'un réseau est invisible. Chaque entrée de la base de données de routage inclut la prochaine passerelle à laquelle les datagrammes destinés à l'entité doivent être envoyés. De plus, elle inclut une « métrique » mesurant la distance totale menant à l'entité. La distance est un concept quelque peu généralisé, qui peut couvrir le délai d'acheminement des messages vers l'entité, son coût d'émission en $, etc. Les algorithmes à vecteurs de distance tirent leur nom du fait qu'il est possible de calculer des routes optimales quand la seule information échangée est la liste de ces distances. En outre, l'information n'est échangée qu'entre entités adjacentes, c.-à-d. des entités partageant un réseau commun.

Bien que le routage soit la plupart du temps basée sur de l'information concernant les réseaux, il est parfois nécessaire de garder une trace des routes menant aux hôtes individuels. Le protocole RIP ne fait aucune distinction formelle entre réseaux et hôtes. Il décrit simplement l'échange d'information concernant des destinations, qui peuvent être soit des réseaux, soit des hôtes. (Notez néanmoins qu'un implémenteur peut choisir de ne pas supporter les routes menant à des hôtes. Voyez la section 3.2) En fait, les développements mathématiques se conçoivent le plus à propos en termes de routes menant d'un hôte ou d'une passerelle à un(e) autre. Quand on considère l'algorithme en termes abstraits, il vaut mieux se représenter une entrée de routage pour un réseau comme une abréviation des entrées de routage pour toutes les entités connectées à ce réseau. Ce type d'abréviation n'a de sens que parce que nous considérons que les réseaux n'ont pas de structure interne visible au niveau IP. Par conséquent, nous affectons généralement la même distance à chaque entité d'un réseau donné.

Nous disions au-dessus que chaque entité conserve une base de données de routage comprenant une entrée pour chaque destination possible du système. Une implémentation réelle va probablement devoir conserver l'information suivante sur chaque destination :

adresse	dans les implémentations IP de ces algorithmes, cela sera l'adresse IP de l'hôte ou du réseau.
passerelle	la première passerelle sur la route menant à la destination.
interface	le réseau physique qui doit être utilisé pour atteindre la première passerelle.
métrique	un nombre indiquant la distance vers la destination.
temporisateur	la durée écoulée depuis la dernière mise à jour de l'entrée

De plus, divers drapeaux et d'autres informations internes seront probablement inclus. Cette base de données est initialisée par une description des entités qui sont directement connectées au système. Elle est mise à jour en fonction de l'information reçue dans les messages des passerelles voisines.

L'information la plus importante échangée entre hôtes et passerelles est véhiculée par les messages de mise à jour. Chaque entité qui participe au processus de routage envoie des messages de mise à jour qui décrivent la base de données de routage comme elle existe actuellement dans cette entité. Il est possible de maintenir des routes optimales pour le système entier en utilisant uniquement les informations obtenues depuis les entités voisines. L'algorithme utilisé pour cela sera décrit dans la section suivante.

Comme nous l'avons mentionné plus haut, le but du routage est de déterminer un chemin pour amener des datagrammes à leur destination finale. Les algorithmes à vecteurs

de distance sont basés sur une table fournissant la meilleure route vers chaque destination du système. Bien sûr, pour définir quelle route est la meilleure, nous devons disposer d'un moyen de mesure de sa « bonté ». On la référence sous le nom de « métrique ».

Dans les réseaux simples, il est habituel d'utiliser une métrique qui compte simplement le nombre de passerelles qu'un message doit traverser. Dans des réseaux plus complexes, une métrique est choisie pour représenter le délai total qu'endure le message, son coût d'émission, ou une autre quantité pouvant être minimisée. L'exigence principale est qu'il doit être possible de représenter la métrique comme une somme de « coûts » pour les sauts individuels.

Formellement, s'il est possible de se rendre directement d'une entité i à une entité j (c.-à-d. sans traverser d'autres passerelles intermédiaires), alors un coût $d(i, j)$ est associé au saut entre i et j. Dans le cas normal où toutes les entités d'un réseau donné sont considérées être similaires, $d(i, j)$ est le même pour toutes les destinations d'un réseau donné, et représente le coût d'utilisation de ce réseau. Pour obtenir la métrique d'une route complète, il suffit d'additionner les coûts individuels des sauts composant la route. Dans le cadre de ce document, nous supposons que les coûts sont des entiers positifs.

Soit $D(i, j)$ la métrique de la meilleure route allant de l'entité i à l'entité j. Elle devrait être définie pour chaque paire d'entités. $d(i, j)$ représente les coûts des pas individuels. Formellement, soit $d(i, j)$ le coût du chemin direct allant de l'entité i à l'entité j. Il vaut ∞ si i et j ne sont pas des voisins immédiats. (Notez que $d(i, i)$ égale ∞, c.-à-d. que nous considérons qu'il n'existe pas de connexion directe d'un nœud vers lui-même.) Puisque les coûts s'additionnent, il est facile de montrer que la meilleure métrique doit être décrite par

$$\forall\, i, j \;:\; D(i,j) = \begin{cases} 0 & \text{si } i = j \\ \min\limits_{k} \left[d(i,k) + D(k,j) \right] & \text{sinon} \end{cases}$$

et que les meilleures routes débutent en allant de i aux voisins k pour lesquels $d(i, k) + D(k, j)$ possède la valeur minimale. (Cela peut être démontré par induction sur le nombre de pas des routes.) Notez que nous pouvons limiter la deuxième équation aux k qui sont des voisins immédiats de i. Pour les autres, $d(i, k) = \infty$, de sorte que le terme les impliquant ne peut jamais être le minimum.

Il s'avère que l'on peut calculer la métrique par un simple algorithme basé sur ceci : l'entité i contacte ses voisins k pour qu'ils lui envoient leurs estimations des distances vers la destination j. Quand i obtient les estimations de k, il ajoute $d(i, k)$ à chacun des nombres. C'est simplement le coût de traversée du réseau entre i et k. De temps à autre, i compare les valeurs de ses voisins et prend la plus petite.

Une preuve est données dans [2] que cet algorithme convergera vers les estimations correctes de $D(i, j)$ en un temps fini en l'absence de changement de topologie. Les auteurs ne font que peu de suppositions quant à l'ordre dans lequel les entités s'envoient leur information l'une l'autre, ou quand le min est recalculé. En gros, les entités ne peuvent pas simplement arrêter d'envoyer des messages ou de recalculer des métriques, et les réseaux ne peuvent retarder les messages indéfiniment. (Le crash d'une entité de routage est un changement de topologie.) De plus, leur preuve ne fait pas usage d'hypothèse relative aux estimations initiales de $D(i, j)$, à part qu'elles doivent être non négatives. Le fait que ces hypothèses plutôt faibles soient suffisamment bonnes est important. Puisqu'on ne doit pas

faire d'hypothèses sur le moment d'envoi des mises à jour, on peut exécuter l'algorithme de façon asynchrone en toute sécurité, c.-à-d. que chaque entité peut envoyer des mises à jour en fonction de sa propre horloge. Les mises à jour peuvent être perdues par le réseau, pour autant qu'elles ne le soient pas toutes. Puisqu'on ne doit pas faire d'hypothèses sur la condition de démarrage, l'algorithme peut gérer les changements. Quand le système change, l'algorithme de routage commence à converger vers un nouvel équilibre, en utilisant l'ancien comme point de départ. Il est important que l'algorithme converge en un temps fini quel que soit le point de départ. Sinon, certains types de changements pourraient mener à un comportement non convergent.

L'exposé de l'algorithme donné plus haut (et la preuve) suppose que chaque entité conserve des copies des estimations provenant de chacun de ses voisins, et calcule de temps à autre un minimum sur tous les voisins. En fait, les implémentations réelles ne font pas nécessairement cela. Elles se rappellent simplement de la meilleure métrique rencontrée jusqu'ici, et l'identité du voisin qui l'a envoyée. Elles remplacent cette information à chaque fois qu'elles voient une meilleure (c.-à-d. plus petite) métrique. Cela leur permet de calculer le minimum de façon incrémentale, sans avoir à stocker les données de tous les voisins.

Il y a une autre différence entre l'algorithme comme décrit dans les textes, et ceux utilisés dans des protocoles réels comme RIP : la description ci-dessus ferait inclure par chaque entité une entrée pour elle-même, en montrant une distance de zéro. En fait, ce n'est généralement pas le cas. Rappelez-vous que toutes les entités présentes sur un réseau sont normalement résumées en une seule entité pour le réseau. Considérez la situation d'un hôte ou d'une passerelle G qui est connectée au réseau A. C représente le coût d'utilisation du réseau A (habituellement une métrique de un). (Rappelez-vous que nous supposons que la structure interne d'un réseau n'est pas visible pour IP, et que le coût de déplacement entre deux entités quelconques est par conséquent toujours le même.) En principe, G devrait obtenir un message de chaque autre entité sur le réseau A, en montrant un coût de 0 pour aller de cette entité à elle-même. G calculerait ensuite C + 0 comme la distance la séparant de H. Plutôt que G doive regarder tous ces messages identiques, l'algorithme démarre simplement en créant une entrée pour le réseau A dans sa table, et en lui affectant une métrique de C. Cette entrée pour le réseau A devrait être perçue comme un résumé des entrées de toutes les entités du réseau A. La seule entité sur A qui ne peut être récapitulée par cette entrée commune est G elle-même, car le coût du voyage entre G et G est 0, et pas C. Mais puisque nous n'avons jamais besoin de ces entrées nulles, nous pouvons nous en passer sans problème en conservant uniquement l'entité pour le réseau A. Notez une autre implication de cette stratégie : puisque les entrées nulles ne sont absolument pas nécessaires, les hôtes ne fonctionnant pas comme passerelle ne doivent pas envoyez de message de mise à jour. À l'évidence, les hôtes qui ne font pas office de passerelle (c.-à-d. les hôtes qui ne sont connectés qu'à un seul réseau) ne peuvent disposer d'autre information utile pour contribuer que leur propre entrée $D(i, i) = 0$. Comme ils n'ont qu'une seule interface, il est facile de voir qu'une route vers n'importe quel réseau les traversant ira simplement sur cette interface et en ressortira immédiatement. Par conséquent, le coût d'une telle route sera plus élevé que le meilleur coût d'au moins C. Puisque nous n'avons pas besoin des entrées nulles, les hôtes non-passerelle n'ont aucunement besoin de participer au protocole de routage.

Résumons ce qu'un hôte ou une passerelle G fait. Pour chaque destination du système, G conservera une estimation de la métrique courante pour cette destination (c.-à-d. le coût total pour l'atteindre) et l'identité de la passerelle voisine dont les données ont servi de base

au calcul de la métrique. Si la destination est sur un réseau qui est directement connecté à G, alors G utilise simplement une entrée montrant le coût d'utilisation du réseau, et le fait qu'aucune passerelle n'est nécessaire pour atteindre la destination. Il est facile de montrer qu'une fois que le calcul a convergé vers les métriques correctes, le voisin qui est enregistré par cette technique est en fait la première passerelle sur le chemin vers la destination. (S'il y a plusieurs chemins de même coût, il s'agit de la première passerelle sur l'un d'entre eux.) La combinaison de la destination, de la métrique et de la passerelle est typiquement référencée comme une route vers la destination avec cette métrique, en utilisant cette passerelle.

La méthode vue jusqu'ici ne permet que de diminuer la métrique, car la métrique existante est conservée jusqu'à ce qu'une autre plus petite apparaisse. Il est possible que l'estimation initiale soit trop basse. Par conséquent, il doit y avoir un moyen d'augmenter la métrique. Il s'avère suffisant d'utiliser la règle suivante : supposez que la route actuelle vers une destination a la métrique D et utilise la passerelle G. Si un nouveau jeu d'informations arrive depuis une autre source que G, ne mettez à jour la route que si la nouvelle métrique est meilleure que D. Mais si de nouvelles informations arrivent depuis G elle-même, mettez *toujours* à jour D avec la nouvelle valeur. Il est facile de montrer qu'avec cette règle, le processus de mise à jour incrémentale produit les mêmes routes qu'un calcul se souvenant de la dernière information provenant de tous les voisins et obtiens un minimum explicite. (Notez que la discussion suppose à cet instant que la configuration du réseau est statique. Elle ne prend pas en compte la possibilité qu'un système puisse tomber en panne.).

Pour résumer, voici l'algorithme à vecteurs de distance de base comme il a été développé jusqu'à présent. (Notez que ce n'est pas une description du protocole RIP. Il y a encore plusieurs raffinements à ajouter.) La procédure suivante est entreprise par chaque entité qui participe au protocole de routage (Ceci doit inclure toutes les passerelle du système. Les hôtes qui ne sont pas des passerelles peuvent également participer) :

- conservez une table avec une entrée pour chaque destination possible du système. L'entrée contient la distance D vers la destination, et la première passerelle G sur la route vers ce réseau. Conceptuellement, il devrait y avoir une entrée pour l'entité elle-même, de métrique 0, mais elle n'est en fait pas incluse.
- Périodiquement, envoyez une mise à jour de routage à chaque voisin. La mise à jour est un groupe de messages contenant toute l'information de la table de routage. Elle contient une entrée pour chaque destination, avec la distance menant à celle-ci.
- Quand une mise à jour de routage arrive depuis un voisin G', ajoutez le coût associé au réseau partagé avec G'. (Cela devrait être le réseau par lequel la mise à jour est arrivée.) Appelons la distance résultante D'. Comparez les distances résultantes avec les entrées actuelles de la table de routage. Si la nouvelle distance D' pour N est plus petite que la valeur existante D, adoptez la nouvelle route, c.-à-d. modifiez l'entrée N de la table pour qu'elle ait une métrique D' et une passerelle G'. Si G' est la passerelle d'où provenait la route existante, c.-à-d. si G' = G, alors utilisez la nouvelle métrique même si elle est plus grande que l'ancienne

2.1 S'accommoder des changements dans la topologie

La discussion ci-dessus suppose que la topologie du réseau est fixe. En pratique, les passerelle et les lignes tombent souvent en panne et redeviennent actives. Pour traiter cette possibilité, nous devons modifier légèrement l'algorithme. La version théorique de l'algorithme impliquait un minimum sur tous les voisins immédiats. Si la topologie change, le jeu de voisins change. Par conséquent, la prochaine fois qu'un calcul sera effectué, le changement sera reflété. Néanmoins, comme mentionné plus haut, les implémentations réelles utilisent une version incrémentale de la minimisation. Seule la meilleure route vers toute destination est conservée. Si la passerelle impliquée dans cette route venait à se crasher, ou si la connexion réseau se rompait, le calcul ne reflèterait jamais le changement. L'algorithme énoncé jusqu'ici dépend du fait qu'une passerelle avertisse ses voisins si ses métriques changent. Si la passerelle crashe, elle n'a alors aucun moyen de prévenir ses voisins d'un changement.

Afin de traiter les problèmes de ce type, les protocoles à vecteurs de distance doivent prendre certaines dispositions pour invalider des routes. Les détails dépendent du protocole spécifique. Par exemple, dans RIP, chaque passerelle qui participe au routage envoie un message de mise à jour à tous ses voisins toutes les 30 secondes. Supposez que la route actuelle pour le réseau N utilise la passerelle G. Si nous n'avons pas de nouvelles de G depuis 180 secondes, nous pouvons supposer que soit la passerelle a crashé, soit la connexion nous y reliant est devenue indisponible. Ainsi donc, nous marquons la route comme étant invalide. Quand nous entendons un autre voisin qui a une route valide vers N, la route valide remplacera l'invalide. Notez que nous attendons 180 secondes avant d'invalider une route même si nous nous attendons à recevoir des nouvelles de chaque voisin toutes les 30 secondes. Malheureusement, des messages sont occasionnellement perdus par les réseaux. Par conséquent, il n'est probablement pas souhaitable d'invalider une route sur base d'un seul message manqué.

Comme nous le verrons ci-dessous, il est utile de disposer d'un moyen d'avertir les voisins qu'il n'y a actuellement pas de route valide vers un réseau donné. RIP, ainsi que plusieurs autres protocoles de cette classe, effectue cela via un message de mise à jour normal, en marquant ce réseau comme étant inaccessible. Une valeur de métrique spécifique est choisie pour indiquer une destination injoignable ; cette valeur de métrique est plus grande que la plus grande métrique valide que l'on s'attend à voir. Dans l'implémentation existante de RIP, la valeur 16 est utilisée. Cette valeur est habituellement référencée comme l'« infini », car elle est plus grande que la plus grande métrique valide. 16 peut sembler être un nombre étonnamment petit. On l'a choisi petit à ce point pour des raisons que nous verrons sous peu. Dans la plupart des implémentations, la même convention est utilisée en interne pour marquer une route comme étant invalide.

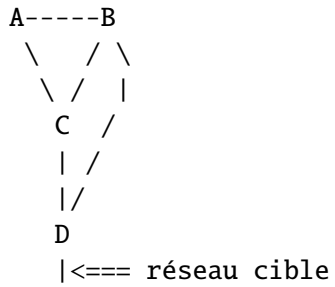
2.2 Éviter l'instabilité

L'algorithme présenté jusqu'ici permettra toujours à un hôte ou une passerelle de calculer une table de routage correcte. Néanmoins, cela n'est pas encore assez pour le rendre utile en pratique. Les preuves précitées auxquelles on se réfère montrent uniquement que les tables de routage convergeront vers les valeurs correctes en un temps fini. Elles ne garantissent

pas que ce temps sera suffisamment court pour être utile, ni ne disent ce qui arrivera aux métriques des réseaux devenus inaccessibles.

Il est assez facile d'étendre les mathématiques pour traiter les routes devenues inaccessibles. La convention suggérée plus haut fera cela. Nous choisissons une grande valeur de métrique pour représenter l'« infini ». Cette valeur doit être suffisamment grande pour qu'aucune métrique réelle ne puisse l'égaler. Pour les besoins de cet exemple, nous utiliserons la valeur 16. Supposons qu'un réseau devienne inaccessible. Toutes les passerelles voisines immédiates deviennent obsolètes et fixent la métrique pour ce réseau à 16. Pour les besoins de l'analyse, nous pouvons supposer que toutes les passerelles voisines ont obtenu un nouveau matériel les connectant directement au réseau disparu, avec un coût de 16. Puisque c'est la seule connexion au réseau disparu, toutes les autres passerelles du système vont converger vers de nouvelles routes qui traversent l'une de ces passerelles. Il est facile de voir qu'une fois que la convergence s'est produite, toutes les passerelles auront des métriques d'au moins 16 pour le réseau disparu. Les passerelles situées à un saut des voisines d'origine finiront avec des métriques d'au moins 17 ; celles situées à deux sauts des voisines d'origine finiront avec des métriques d'au moins 18, etc. Comme ces métriques sont plus grandes que la valeur de métrique maximale, elles sont toutes fixées à 16. Il est évident que le système convergera maintenant vers une métrique de 16 pour le réseau disparu, et ce pour toutes les passerelles.

Malheureusement, la question du temps que prendra la convergence n'est pas réductible en une réponse aussi simple. Avant d'aller plus loin, il sera utile de regarder un exemple (emprunté de [2]). Notez, à propos, que ce que nous sommes sur le point de montrer ne se passera pas avec une implémentation correcte de RIP. Nous essayons de montrer pourquoi certaines fonctionnalités sont nécessaires. Notez que les lettres correspondent à des passerelles, et les lignes à des réseaux.



Dans cet exemple, tous les réseaux ont un coût de 1, sauf le lien direct allant de C à D, qui a un coût de 10.

Chaque passerelle disposera d'une table montrant une route vers chaque réseau. Néanmoins, à des fins d'illustration, nous ne montrons que les routes menant de chaque passerelle au réseau marqué au bout du diagramme.
Voici les chemins d'accès au réseau cible depuis chacun des hôtes/passerelles :

```
D : directement connecté, métrique 1
B : route via D, métrique 2
C : route via B, métrique 3
A : route via B, métrique 3
```

Supposons maintenant que le lien de B à D tombe en panne. Les routes devraient maintenant être ajustées pour utiliser le lien allant de C à D. Malheureusement, cela prendra un moment pour que cela se produise. Les changements de routage débutent quand B remarque que la route menant à D n'est plus utilisable. Pour la simplicité, le tableau ci-dessous suppose que toutes les passerelles envoient des mises à jour au même moment. Le tableau montre la métrique du réseau cible, comme elle apparaît dans la table de routage de chaque passerelle.

Évolution de la métrique du réseau cible au cours du temps													
Hôte	via	coût	via	coût	via	coût	via	coût		via	coût	via	coût
D	dir[a]	1	dir	1	dir	1	dir	1	...	dir	1	dir	1
B	inac[b]		C	4	C	5	C	6		C	11	C	12
C	B	3	A	4	A	5	A	6		A	11	A	12
A	B	3	C	4	C	5	C	6		C	11	C	12

[a] directement connecté
[b] inaccessible

Voici le problème : B est capable de se débarrasser de la route en panne en utilisant un mécanisme de temporisation. Mais des vestiges de cette route persistent dans le système pendant une longue période. Initialement, A et C pensent toujours qu'ils peuvent atteindre D via B. Aussi, ils continuent d'envoyer des mises à jour indiquant des métriques de 3. Lors de l'itération suivante, B fera savoir qu'il peut atteindre D via soit A soit C. Bien sûr, il ne le peut pas. Les routes signalées par A et C ne sont maintenant plus envisageables, mais ils n'ont aucun moyen de déjà le savoir. Et même lorsqu'ils découvrent que leurs routes via B se sont volatilisées, ils pensent tous deux qu'il y a une route disponible via l'autre. Finalement, le système converge, comme les mathématiques le soutiennent, mais cela peut prendre un peu de temps avant que cela ne se produise. Le pire cas se présente quand un réseau devient complètement inaccessible depuis une partie du système. Dans ce cas, les métriques vont s'accroître lentement d'une manière semblable à celle vue plus haut qu'elles atteignent finalement l'infini. Pour cette raison, le problème est appelé « comptage à l'infini ».

Vous devriez maintenant voir pourquoi l'« infini » doit être choisi aussi petit que possible. Si un réseau devient complètement inaccessible, il est souhaitable que le comptage à l'infini cesse dès que possible. L'infini doit être suffisamment grand pour qu'aucune route ne soit aussi longue. Mais il ne devrait pas être plus grand que nécessaire. Ainsi, le choix de l'infini est un compromis entre taille du réseau et vitesse de convergence en cas de comptage à l'infini. Les concepteurs de RIP croyaient que le protocole ne serait probablement pas utilisable dans un réseau d'un diamètre supérieur à 15.

Il y a plusieurs méthodes qui peuvent être employées pour éviter des problèmes comme celui-ci. Celles utilisées par RIP sont appelées « Horizon partagé avec empoisonnement », et « Mises à jour déclenchées ».

2.2.1 Horizon partagé

Notez que certains des problèmes exposés plus haut proviennent du fait que A et C sont engagés dans une partie de tromperie mutuelle. Chacun prétend être capable de rejoindre D via l'autre. Cela peut être évité en faisant un peu plus attention à l'endroit où l'information est envoyée. En particulier, il n'est jamais utile de proclamer l'accessibilité d'un réseau

destination au(x) voisin(s) duquel(desquels) on a appris la route. L'« horizon partagé » est un mécanisme destiné à éviter les problèmes causés par l'inclusion de routes dans des mises à jour envoyées à une passerelle, alors que cette passerelle est elle-même à l'origine de ces informations. Le mécanisme d'« horizon partagé simple » omet les routes apprises depuis un voisin dans les mises à jour envoyées à ce voisin. L'« horizon partagé avec empoisonnement » inclut de telles routes dans les mises à jour, mais fixe leur métrique à l'infini.

Si A pense qu'il peut atteindre D via C, ses messages vers C devraient indiquer que D n'est pas joignable. Si la route vers C est réelle, alors soit C dispose d'une connexion directe vers D, soit d'une connexion passant par une autre passerelle quelconque. La route de C peut éventuellement ne pas revenir à A, puisque cela forme une boucle. En indiquant à C que D est inaccessible, A se prémunit simplement contre la possibilité que C puisse être embrouillé et croire qu'il y a une route passant par A. C'est évident pour une ligne point-à-point. Mais considérez le cas où A et C sont connectés par un réseau à diffusion comme Ethernet, et qu'il y a d'autres passerelles sur ce réseau. Si A a une route via C, il devrait indiquer que D est inaccessible quand il parle à une autre passerelle de ce réseau. Les autres passerelles du réseau peuvent atteindre C elles-mêmes. Elles n'auront jamais besoin de passer par A pour accéder à C. Si la meilleure route de A passe réellement par C, aucune autre passerelle de ce réseau n'a besoin de savoir que A peut atteindre D. C'est heureux, car cela signifie que le même message de mise à jour qui a été utilisé pour C peut être utilisé pour toutes les autres passerelles du même réseau. Par conséquent, les messages de mise à jour peuvent être émis par diffusion.

En général, l'horizon partagé avec empoisonnement est plus sûr que l'horizon partagé simple. Si deux passerelles possèdent des routes pointant l'une vers l'autre, l'annonce de routes empoisonnées avec une métrique de 16 cassera la boucle immédiatement. Si les routes empoisonnées ne sont simplement pas annoncées, les routes erronées devront être éliminées par l'attente de l'expiration d'une temporisation. Néanmoins, l'empoisonnement a un désavantage : il accroît la taille des messages de routage. Considérez le cas d'un backbone [8] de campus connectant plusieurs bâtiments différents. Dans chaque bâtiment, il y a une passerelle connectant le backbone à un réseau local. Réfléchissez à quelles mises à jour de routage ces passerelles devraient diffuser sur le réseau backbone. Tout ce que le reste du réseau doit réellement savoir sur chaque passerelle est l'identité des réseaux locaux qui y sont connectés. En utilisant l'horizon partagé simple, seules ces routes apparaîtront dans les messages de mise à jour envoyés par la passerelle au réseau backbone. Si l'horizon partagé avec empoisonnement est utilisé, la passerelle doit mentionner toutes les routes qu'elle apprend du backbone, avec une métrique de 16. Si le système est grand, cela peut résulter en un grand message de mise à jour, dont presque toutes les entrées indiquent des réseaux inaccessibles.

Dans un certain sens statique, l'annonce de routes empoisonnées avec une métrique de 16 ne fournit pas d'information supplémentaire. S'il y a beaucoup de passerelles sur un réseau à diffusion, ces entrées supplémentaires peuvent utiliser une bande passante significative. La raison pour laquelle elles sont présentes est d'améliorer le comportement dynamique. Quand la topologie change, mentionner les routes qui ne devraient pas traverser la passerelle aussi bien que celles qui le devraient peut accélérer la convergence. Néanmoins, dans certaines situations, les gestionnaires de réseaux peuvent préférer accepter une convergence un peu plus lente afin de minimiser la surcharge due au routage. De ce fait, les implémen-

[8] réseau fédérateur

teurs peuvent à leur convenance implémenter l'horizon partagé simple plutôt que l'horizon partagé avec empoisonnement, ou peuvent fournir une option de configuration permettant au gestionnaire de réseaux de choisir quel comportement utiliser. Il est également permis d'implémenter des mécanismes hybrides qui annoncent certaines routes empoisonnées avec une métrique de 16 et omettent les autres. Un exemple d'un tel mécanisme serait d'utiliser une métrique de 16 pour les routes empoisonnées pour une certaine période de temps après les changements de routage les impliquant, et après cela les omettre dans les mises à jour.

2.2.2 Mises à jour déclenchées

L'horizon partagé avec empoisonnement empêchera toute boucle de routage n'impliquant que deux passerelles. Néanmoins, il est toujours possible d'arriver à des situations où trois passerelles sont engagées dans une partie de tromperie mutuelle. Par exemple, A peut croire qu'il a une route vers B, B vers C, C vers A. L'horizon partagé ne peut arrêter une telle boucle. La boucle ne sera résolue que lorsque la métrique atteindra l'infini, et le réseau impliqué sera ensuite déclaré injoignable. Les mises à jour déclenchées constituent une tentative d'accélérer cette convergence. Pour utiliser des mises à jour déclenchées, nous ajoutons simplement une règle qui dit qu'à chaque fois qu'une passerelle change la métrique d'une route, elle doit envoyer des messages de mise à jour presque immédiatement, même si ce n'est pas encore le moment d'envoi du message de mise à jour régulier. (Les détails de chronométrage différeront de protocole à protocole. Certains protocoles à vecteurs de distance, RIP compris, spécifient un délai faible, afin d'éviter que des mises à jour déclenchées ne génèrent un trafic réseau excessif.). Notez la façon dont cela se combine avec les règles de calcul de nouvelles métriques. Supposons que la route allant d'une passerelle à la destination N emprunte la passerelle G. Si une mise à jour provient de G elle-même, la passerelle réceptrice *doit* croire la nouvelle information, que la nouvelle métrique soit supérieure ou inférieure à l'ancienne. Si le résultat est une modification de la métrique, alors la passerelle réceptrice enverra des mises à jour déclenchées à tous les hôtes et passerelles qui y sont directement connectés. Ils peuvent alors à leur tour envoyer des mise à jour à leurs voisins. Le résultat est une cascade de mises à jour déclenchées. Il est facile de montrer quels hôtes et passerelles sont impliqués dans la cascade. Supposez qu'une passerelle G soutient qu'une route vers la destination N est périmée. G enverra des mises à jour déclenchées à tous ses voisins. Néanmoins, les seuls voisins qui croiront la nouvelle information sont ceux dont les routes vers N passent par G. Les autres passerelles et hôtes considéreront ceci comme une information sur une nouvelle route moins bonne que celle qu'ils utilisent déjà, et l'ignoreront. Les voisins dont les routes passent par G mettront à jour leurs métriques et enverront des mises à jour déclenchées à tous leurs voisins. À nouveau, seuls les voisins dont les routes les traversent y prêteront attention. Par conséquent, les mises à jour déclenchées se propageront vers l'arrière le long de tous les chemins menant à la passerelle G, en mettant à jour les métriques vous leur donner la valeur « infini ». Cette propagation s'arrêtera dès qu'elle atteint une partie du réseau dont la route vers la destination N emprunte un autre chemin.

Si le système pouvait rester tranquille lorsque la cascade de mises à jour déclenchées se produit, il serait possible de prouver que le comptage à l'infini ne se produira jamais. Les mauvaises routes seront toujours supprimées immédiatement, et aucune boucle de routage ne pourrait se former.

Malheureusement, la réalité n'est pas aussi idyllique. Pendant que les mises à jour déclenchées sont envoyées, des mises à jour régulières peuvent se produire au même moment. Les passerelles qui n'ont pas encore reçu la mise à jour déclenchée enverront toujours de l'information basée sur la route qui n'existe plus. Il est possible qu'après que la mise à jour déclenchée ait traversé une passerelle, elle puisse recevoir une mise à jour normale de l'une des passerelles qui n'a pas encore été prévenue. Cela pourrait reconstituer un vestige orphelin de la route défectueuse. Si les mises à jour déclenchées se produisent suffisamment rapidement, c'est très improbable. Néanmoins, le comptage à l'infini est toujours possible.

3 Spécifications du protocole

RIP doit permettre à des hôtes et passerelles d'échanger de l'information pour calculer des routes au travers d'un réseau IP. RIP est un protocole à vecteurs de distance. Il possède donc les fonctionnalités générales décrites dans la section 2. RIP peut être implémenté à la fois par les hôtes et les passerelles. Comme dans la plupart de la documentation IP, le terme « hôte » sera utilisé ici indifféremment pour désigner l'un ou l'autre. RIP est utilisé pour véhiculer de l'information sur les routes vers des « destinations » pouvant être des hôtes individuels, des réseaux, ou une destination spéciale utilisée pour transmettre une route par défaut.

Tout hôte utilisant RIP est censé disposer d'interfaces vers un ou plusieurs réseaux. Ils sont référencés sous le terme de « réseaux directement connectés ». Le protocole se base sur l'accès à certaines informations sur chacun de ces réseaux. La plus importante est sa métrique ou « coût ». La métrique d'un réseau est un entier compris entre 1 et 15 inclus. Elle est définie d'une manière non spécifiée par ce protocole. La plupart des implémentations existantes utilisent toujours une métrique de 1. De nouvelles implémentations devraient permettre à l'administrateur système de fixer le coût de chaque réseau. En plus du coût, chaque réseau aura un numéro de réseau IP et le masque de sous-réseau associé. Ils doivent être spécifiés par l'administrateur système d'une façon non spécifiée par ce protocole.

Notez que les règles spécifiées dans la section 3.2 supposent qu'il y a un seul masque de sous-réseau s'appliquant à chaque réseau IP, et que seuls les masques de sous-réseau des réseaux directement connectés sont connus. Il peut y avoir des systèmes qui utilisent des masques de sous-réseau pour différents sous-réseaux à l'intérieur d'un unique réseau. Il peut également y avoir des exemples où il vaut mieux qu'un système connaisse les masques de sous-réseau des réseaux distants. Néanmoins, de telles situations requerraient des modifications des règles qui gouvernent la propagation d'information sur les sous-réseaux. De telles modifications soulèvent des problèmes d'interopérabilité, et doivent donc être considérées comme une modification du protocole.

Chaque hôte implémentant RIP doit posséder une table de routage. Cette table comprend une entrée pour chaque destination qui est accessible via le système décrit par RIP. Chaque entrée contient au moins les informations suivantes :

- L'adresse IP de la destination.
- Une métrique, qui représente le coût total de transport d'un datagramme de l'hôte à cette destination. Cette métrique est la somme des coûts associés aux réseaux qui seraient traversés pour arriver à la destination.

– L'adresse IP de la prochaine passerelle le long du chemin vers la destination. Si la destination est sur l'un des réseaux directement connectés, cet élément n'est pas nécessaire.
– Un drapeau pour indiquer que l'information sur la route a changé récemment. Il sera référencé sous le nom de « drapeau de changement de route ».
– Différents temporisateurs associés à la route. Voyez la section 3.3 pour plus de détails à leur sujet.

Les entrées pour les réseaux directement connectés sont définies par l'hôte, en utilisant des informations récoltées par des moyens non spécifiés par ce protocole. La métrique d'un réseau directement connecté est définie par le coût de ce réseau. Dans les implémentations RIP existantes, 1 est toujours utilisé comme coût. Dans ce cas, la métrique RIP se réduit à un simple comptage des sauts. Des métriques plus complexes peuvent être utilisées quand il est préférable d'indiquer une priorité de certains réseaux sur d'autres, par exemple à cause de différences de bande passante ou de fiabilité.

Les implémenteurs peuvent également choisir de permettre à l'administrateur système d'entrer des routes additionnelles. Celles-ci seraient plus que probablement des routes vers des hôtes ou réseaux à l'extérieur de la portée du système de routage.

Les entrées pour les destinations autres que celles initiales sont ajoutées et mises à jour par les algorithmes décrits dans les sections suivantes.

Afin que le protocole fournisse une information de routage complète, chaque passerelle du système doit participer au processus. Les hôtes qui ne sont pas des passerelles ne doivent pas participer, mais beaucoup d'implémentations prendre des dispositions pour qu'ils écoutent l'information de routage afin de leur permettre de maintenir leurs tables de routage.

3.1 Formats des messages

RIP est un protocole basé sur UDP. Chaque hôte utilisant RIP dispose d'un processus de routage qui envoie et reçoit des datagrammes sur le port UDP n° 520. Toutes les communications adressées à un processeur RIP d'un autre hôte sont envoyées au port 520. Tous les messages de mise à jour de routage sont envoyés depuis le port 520. Des messages de mise à jour de routage non sollicités ont pour port source *et* port destination le n° de port 520. Ceux envoyés en réponse à une requête sont envoyés au port d'où provenait la requête. Des requêtes spécifiques et des requêtes de déboggage peuvent être envoyées depuis des ports différents de 520, mais sont dirigées vers le port 520 de la machine cible.

Il y a des dispositions dans le protocole qui permettent des processus RIP « silencieux ». Un processus silencieux n'envoie normalement aucun message. Néanmoins, il écoute les messages envoyés par d'autres. Un RIP silencieux pourrait être utilisé par des hôtes qui n'agissent pas en tant que passerelles, mais qui veulent écouter les mises à jour de routage afin de surveiller les passerelles locales et de maintenir leurs tables de routage interne à jour. (Voyez [5] pour une discussion sur les différentes manières dont les hôtes peuvent garder une trace de la topologie d'un réseau.) Une passerelle qui a perdu le contact avec tous ses réseaux sauf un pourrait choisir de devenir silencieuse, puisqu'elle n'est alors plus une passerelle.

Néanmoins, cela ne devrait pas être fait s'il y a la moindre possibilité que des passerelles voisines dépendent de ses messages pour détecter que le réseau en panne est à nouveau

opérationnel. (Le programme 4BSD `routed` utilise le routage de paquets pour surveiller le fonctionnement des liens point-à-point.)

Le format des paquets est montré ci-après :

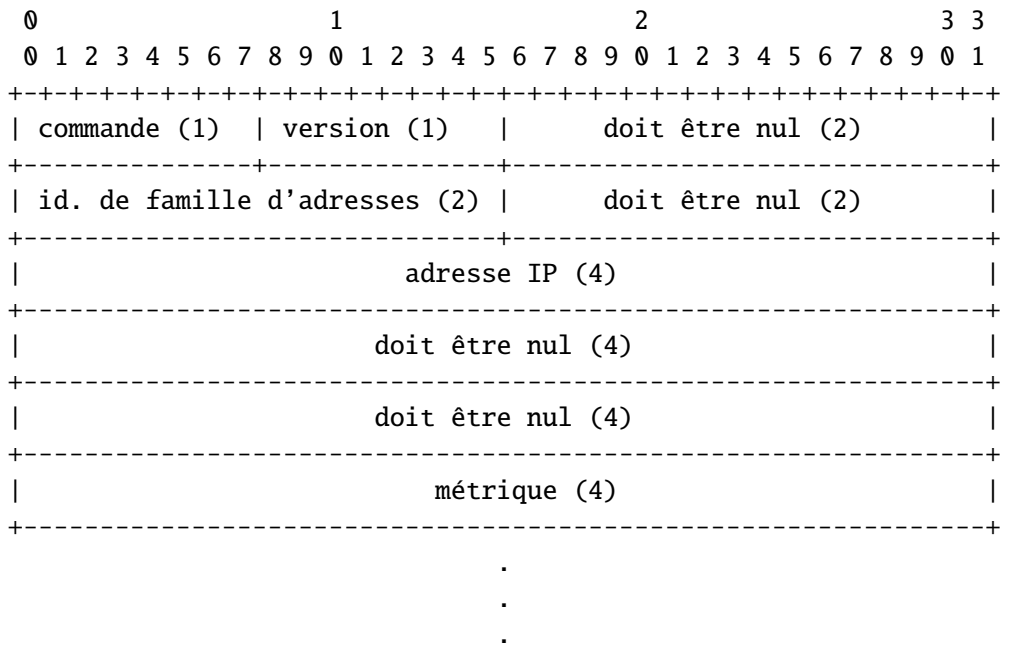


FIG. 1 – *Format des datagrammes contenant de l'information réseau. La taille des champs est donnée en octets. À moins que cela ne soit spécifié différemment, les champs contiennent des entiers binaires, dans l'ordre Internet normal avec l'octet le plus significatif en premier lieu. Chaque chiffre présent au-dessus de la ligne supérieure représente un bit.*

La partie du datagramme allant de l'identificateur de la famille d'adresses à la métrique peut apparaître jusqu'à 25 fois. L'adresse IP est l'adresse Internet habituelle sur 4 octets, dans l'ordre de transmission réseau.

Chaque datagramme contient une commande, un numéro de version et des arguments éventuels. Ce document décrit la version 1 du protocole. Les détails du traitement du n° de version sont décrits dans la section 3.4. Le champ « commande » est utilisé pour spécifier le but de ce datagramme. Voici un résumé des commandes implémentées dans la version 1 :

	commande	description
1	request	Une requête au système répondant indiquant d'envoyer tout ou partie de sa table de routage.
2	response	Un message contenant tout ou partie de la table de routage de l'émetteur. Ce message peut être envoyé en réponse à une requête ou sur demande, ou peut être un message de mise à jour généré par l'émetteur.
3	traceon	Obsolète. Les messages contenant cette commande doivent être ignorés.
4	traceoff	Obsolète. Les messages contenant cette commande doivent être ignorés.
5	reserved	Cette valeur est utilisée par Sun Microsystems pour ses propres besoins. Si de nouvelles commandes sont ajoutées dans une version ultérieure, elles devraient commencer par 6. Les messages contenant cette commande peuvent être ignorés en toute sécurité par les implémentations qui choisissent de ne pas y répondre.

Pour *request* et *response*, le reste du datagramme contient une liste des destinations, avec des informations sur chacune d'entre elles. Chaque entrée de cette liste contient un réseau ou un hôte destination, et sa métrique associée. Le format de paquet est destiné à permettre à RIP de transporter des informations de routage pour plusieurs protocoles différents. De ce fait, chaque entrée a un identificateur de famille d'adresses indiquant quel type d'adresse est spécifié dans cette entrée. Ce document ne décrit que le routage pour les réseaux de type Internet. L'identificateur de famille d'adresses pour IP est 2. Aucune des implémentations de RIP auxquelles l'auteur a accès n'implémente d'autre type d'adresse. Néanmoins, pour permettre un développement futur, les implémentations doivent sauter les entrées qui spécifient des familles d'adresses qui ne sont pas supportées par l'implémentation. (La taille de ces entrées sera la même que celle d'une entrée spécifiant une adresse IP.) Le traitement du message continue normalement après que toutes les entrées non supportées aient été sautées. L'adresse IP est l'adresse Internet normale, stockée sur 4 octets dans l'ordre de transmission réseau. Le champ ' métrique ' doit contenir une valeur comprise entre 1 et 15 inclus, spécifiant la métrique actuelle pour la destination, ou la valeur 16, qui indique que la destination est inaccessible. Chaque route envoyée par une passerelle supplante toute route vers la même destination provenue précédemment de la même passerelle.

La taille maximale d'un datagramme est de 512 octets. Cela n'inclut que les parties du datagramme décrites ci-dessus. Les en-têtes IP et UDP ne sont pas pris en compte. Les commandes impliquant des informations sur le réseau permettent que l'information soit éclatée sur plusieurs datagrammes. Aucune disposition spéciale n'est nécessaire pour les continuations, car des résultats corrects se produiront si les datagrammes sont traités individuellement.

3.2 Considérations d'adressage

Comme indiqué dans la section 2, le routage à vecteurs de distance peut être utilisé pour décrire des routes vers des hôtes individuels ou vers des réseaux. Le protocole RIP permet n'importe laquelle de ces possibilités. Les destinations apparaissant dans les messages *request* et *response* peuvent être des réseaux, des hôtes, ou un code spécial utilisé pour indiquer une adresse par défaut. En général, les types de routes réellement utilisées dépendront de la stratégie de routage utilisée pour le réseau particulier. Beaucoup de réseaux sont configurés

de sorte qu'une information de routage pour les hôtes individuels n'est pas nécessaire. Si chaque hôte d'un réseau ou d'un sous-réseau donné est accessible au travers des mêmes passerelles, alors il n'y a aucune raison de mentionner les hôtes individuels dans les tables de routage. Néanmoins, les réseaux qui incluent des lignes point-à-point requièrent parfois que les passerelles gardent une trace des routes vers certains hôtes. La nécessité ou non de cette fonctionnalité dépend de l'adressage et de l'approche du routage utilisés dans le système. Par conséquent, certaines implémentations peuvent choisir de ne pas supporter les routes vers des hôtes. Si les routes vers des hôtes ne sont pas supportées, elles doivent être supprimées quand elles sont reçues dans des messages *response*. (Voyez la section 3.4.2)

Les formats de paquets RIP ne font pas de distinction entre les différents types d'adresse. Les champs qui sont étiquetés « adresse » peuvent contenir un des éléments suivants :

- adresse d'hôte
- numéro de sous-réseau
- numéro de réseau
- 0, indiquant une route par défaut

Les entités qui utilisent RIP sont supposées utiliser l'information la plus spécifique disponible lors du routage d'un datagramme, c.-à-d. que lors du routage d'un datagramme, son adresse destination doit d'abord être comparée avec la liste des adresses d'hôtes. Ensuite, elle doit être examinée pour voir si elle correspond à un numéro de sous-réseau ou de réseau connu. Finalement, si aucun des cas précités ne convient, la route par défaut est utilisée.

Quand un hôte évalue l'information qu'il reçoit via RIP, son interprétation d'une adresse dépend de sa connaissance ou non du masque de sous-réseau qui s'applique au réseau. Si c'est le cas, alors il est possible de déterminer la signification de l'adresse. Par exemple, considérons le réseau 128.6. Il a un masque de sous-réseau de 255.255.255.0. Donc, 128.6.0.0 est un numéro de réseau, 128.6.4.0 est un numéro de sous-réseau, et 128.6.4.1 est une adresse d'hôte. Néanmoins, si l'hôte ne connaît pas le masque de sous-réseau, l'évaluation de l'adresse peut être ambiguë. S'il y a une partie hôte non nulle, il n'y a aucun mécanisme sûr pour déterminer si l'adresse représente un numéro de sous-réseau ou une adresse d'hôte. Comme un numéro de sous-réseau serait inutile sans le masque de sous-réseau, les adresses sont supposées représenter des hôtes dans cette situation. Afin d'éviter ce type d'ambiguïté, les hôtes ne doivent pas envoyer de routes de sous-réseaux aux hôtes dont on ne peut présumer qu'ils connaissent le masque de sous-réseau approprié. Normalement, les hôtes ne connaissent les masques de sous-réseau que des réseaux directement connectés. Par conséquent, à moins que des dispositions spéciales n'aient été prises, les routes menant à un sous-réseau ne doivent pas être envoyées à l'extérieur du réseau duquel le sous-réseau fait partie.

Ce filtrage est exécuté par les passerelles à la « frontière » du réseau comportant des sous-réseaux. Ce sont des passerelles qui connectent ce réseau avec d'autres réseaux. À l'intérieur de réseau découpé en sous-réseaux, chaque sous-réseau est traité comme un réseau individuel. Les entrées de routage pour chaque sous-réseau sont passées en revue par RIP. Néanmoins, les passerelles frontière n'envoient aux hôtes des autres réseaux qu'une seule entrée pour le réseau entier. Cela signifie qu'une passerelle frontière enverra des informations différentes à des voisins différents. Pour les voisins connectés au réseau composé de sous-réseaux, elle génère une liste de tous les sous-réseaux auxquels elle est directement connectée, en utilisant le n° de sous-réseau. Pour les voisins connectés à d'autres réseaux, elle crée une unique entité pour le réseau entier, en montrant la métrique associée à ce réseau. (Cette

métrique serait normalement la plus petite métrique des sous-réseaux à laquelle la passerelle est attachée.)

De façon similaire, les passerelles frontières ne doivent pas mentionner de route d'hôtes vers des hôtes situés dans l'un des réseaux directement connectés dans les messages envoyés à d'autres réseaux. Ces routes seront synthétisées par l'entrée unique pour le réseau considéré comme un tout. Nous ne spécifions pas ce qu'il faut faire avec les routes d'hôtes de réseaux « distants » (c.-à-d. les hôtes ne faisant pas partie d'un des réseaux directement connectés). Généralement, ces routes indiquent un hôte qui est accessible via une route qui ne supporte pas d'autres hôtes sur le réseau auquel fait partie l'hôte en question.

L'adresse spéciale 0.0.0.0 est utilisée pour décrire une route par défaut. Une route par défaut est utilisée quand il n'est pas commode de lister tous les réseaux possibles dans les mises à jour RIP, et quand une ou plusieurs des passerelles proches connectées au système sont préparées à traiter du trafic à destination de réseaux qui ne sont pas listés explicitement. Ces passerelles devraient créer des entrées RIP pour l'adresse 0.0.0.0, tout comme si c'était un réseau auquel elles sont connectées. La mise en œuvre pratique de la création d'entrées 0.0.0.0 par une passerelle est laissée aux soins de l'implémenteur. La plupart du temps, l'administrateur système disposera d'un moyen de spécifier quelles passerelles devraient créer des entrées pour 0.0.0.0. Néanmoins, d'autres mécanismes sont possibles. Par exemple, un implémenteur pourrait décider que toute passerelle parlant EGP devrait être déclarée passerelle par défaut. Il peut être utile de permettre à l'administrateur réseau de choisir la métrique à utiliser pour ces entrées. S'il y a plus d'une passerelle par défaut, cela lui permettra d'exprimer une priorité de l'une sur l'autre. Les entrées pour 0.0.0.0 sont traitées par RIP exactement de la même manière qu'un réseau réel ayant cette adresse. Néanmoins, l'entrée est utilisée pour router tout datagramme dont l'adresse de destination ne correspond à aucun des réseaux de la table. Les implémentations ne sont pas obligées de supporter cette convention. Néanmoins, cela est fortement recommandé. Les implémentations qui ne supportent pas 0.0.0.0 doivent ignorer les entrées comportant cette adresse. Dans de telles situations, ils ne doivent pas propager l'entrée dans leurs propres mises à jour RIP. Les administrateurs système devraient s'assurer que les routes vers 0.0.0.0 ne se propagent pas plus loin que prévu. Généralement, chaque système autonome a sa passerelle par défaut préférée. Par conséquent, les routes impliquant 0.0.0.0 ne devraient généralement pas quitter la frontière d'un système autonome. Les mécanismes permettant d'imposer cela ne sont pas spécifiés dans ce document.

3.3 Temporisateurs

Cette section décrit tous les événements déclenchés par des temporisateurs.

Toutes les 30 secondes, le processus de sortie reçoit l'ordre de générer une réponse complète pour chaque passerelle voisine. Quand il y a beaucoup de passerelles sur un même réseau, ces passerelles ont tendance à se synchroniser entre elles de sorte qu'elles émettent toutes des mises à jour au même moment. Cela peut se produire à chaque fois que le temporisateur de 30 secondes est affecté par la charge de travail du système. Il est indésirable que ces messages de mise à jour deviennent synchronisés, car cela peut mener à des collisions inutiles sur les réseaux à diffusion. De ce fait, les implémentations doivent prendre une des deux précautions suivantes :

– Les mise à jour 30-secondes sont déclenchées par une horloge dont le rythme n'est pas affecté par la charge du système ou le temps requis pour s'occuper du temporisateur de mise à jour précédent.
– Le temporisateur 30-secondes est retardé par l'ajout d'un petit temps aléatoire à chaque fois qu'il est déclenché.

Il y a deux temporisateurs associés à chaque route, une « temporisation » et un « temporisateur de ramassage des déchets » [9]. À l'expiration de la temporisation, la route n'est plus valide. Néanmoins, elle est conservée dans la table pour un court moment, le temps que les voisins soient prévenus que la route à été abandonnée. À l'expiration du temporisateur de ramassage des déchets, la route est finalement supprimée des tables.

La temporisation est initialisée quand une route est établie, et à chaque fois qu'un message de mise à jour est reçu pour la route. Si 180 secondes s'écoulent depuis le dernier moment où la temporisation a été initialisée, la route est considérée avoir dépassé sa période de validité, et le processus de suppression que nous sommes sur le point de décrire est démarré à cet effet.

Les suppressions peuvent se produire pour une des deux raisons suivantes :

1. la temporisation expire
2. la métrique est fixée à 16 du fait de la réception d'une mise à jour depuis la passerelle courante

(Voyez la section 3.4.2 pour une discussion sur le traitement des mises à jour provenant d'autres passerelles.) Dans chacun des cas, les événements suivants se produisent :

– Le temporisateur de ramassage des déchets est fixé à 120 secondes.
– La métrique de la route est fixée à 16 (infini). Cela provoque l'abandon de la route.
– Un drapeau est défini ; il note que cette entrée a été modifiée, et le processus de sortie reçoit un signal lui enjoignant de déclencher une réponse.

Jusqu'au moment où le temporisateur de ramassage des déchets expire, la route est incluse dans toutes les mises à jour envoyées par cet hôte, avec une métrique de 16 (infini). Quand le temporisateur de ramassage des déchets expire, la route est supprimée des tables.

Si une nouvelle route vers ce réseau est établie alors que le temporisateur de ramassage des déchets est en cours de fonctionnement, la nouvelle route remplacera celle qui est sur le point d'être effacée. Dans ce cas, le temporisateur de ramassage des déchets doit être réinitialisé.

Voyez la section 3.5 pour une discussion sur un délai qui est requis dans l'exécution de mises à jour déclenchées. Bien que les implémentations de ce délai requièrent un temporisateur, il est plus naturel d'en discuter dans la section 3.5 qu'ici.

3.4 Traitement de l'entrée

Cette section décrira le traitement des datagrammes reçus sur le port UDP 520. Avant de traiter les datagrammes en détail, certaines vérifications de format générales doivent être faites. Elles dépendent du champ nº de version du datagramme de la façon suivante :

[9] garbage-collection timer

0 Les datagrammes dont le numéro de version est 0 doivent être ignorés. Ils proviennent d'une version précédente du protocole, dont le format de paquet était spécifique à la machine.

1 Les datagrammes dont le numéro de version est 1 doivent être traités comme décrit dans le reste de cette spécification. Tous les champs qui sont étiquetés plus haut « doit être nul » doivent être vérifiés. Si l'un de ces champs contient une valeur non nulle, le message entier doit être ignoré.

>1 Les datagrammes dont le numéro de version est supérieur à 1 doivent être traités comme décrit dans le reste de cette spécification. Tous les champs qui sont étiquetés plus haut « doit être nul » doivent être ignorés. De futures versions du protocole pourraient placer des données dans ces champs. Les implémentations de la version 1 doivent ignorer ces données supplémentaires et ne traiter que les champs spécifiés dans ce document.

Après avoir vérifié le numéro de version et effectué toutes les vérifications préliminaires, le traitement dépendra de la valeur du champ de commande.

3.4.1 Request

La commande *request* est utilisée pour demander une réponse contenant tout ou partie de la table de routage de l'hôte. [Notez que le terme « hôte » est utilisé indifféremment pour désigner un hôte ou une passerelle ; dans la plupart des cas, il serait inhabituel qu'un hôte non-passerelle envoie des messages RIP.] Normalement, les requêtes sont envoyés par diffusion, à partir du port UDP source 520. Dans ce cas, les processus silencieux ne répondent pas à la requête. Les processus silencieux sont par définition des processus qui ne devraient normalement pas voir d'information de routage. Néanmoins, il peut y avoir des situations impliquant la surveillance de passerelles où l'on souhaite que même un processus silencieux puisse examiner la table de routage. Dans ce cas, la requête devrait être envoyée depuis un nº de port UDP différent de 520. Si une requête provient du port 520, les processus silencieux ne répondent pas. Si la requête provient de tout autre port, les processus doivent répondre même s'ils sont silencieux.

La requête est traitée entrée par entrée. S'il n'y a pas d'entrée, aucune réponse n'est fournie. Il y a un cas spécial : s'il y a exactement une entrée dans la requête, avec un identificateur de famille d'adresses de valeur nulle (signifiant « non spécifié »), et une métrique de valeur infinie (c.-à-d. 16 pour les implémentations actuelles), il s'agit d'une requête d'envoi de l'entièreté de la table de routage. Dans ce cas, un appel est fait au processus de sortie pour envoyer la table de routage au port requis.

Mis à part ce cas spécial, le traitement est assez simple. Parcourez la liste des entrées de la requête une par une de haut en bas. Pour chaque entrée, recherchez la destination dans la base de données de routage de l'hôte. S'il y a une route, placez la métrique de cette route dans le champ métrique à l'intérieur du datagramme. S'il n'existe pas de route vers la destination spécifiée, placez l'infini (c.-à-d. 16) dans le champ métrique du datagramme. Une fois que toutes les entrées ont été remplies, fixez la commande à ' response ' et renvoyez le datagramme au port d'où il provenait.

Notez qu'il y a une différence de traitement si la requête est relative à un groupe de destinations spécifié, ou à une table de routage entière. Si la requête demande une table

d'hôtes complète, un traitement de sortie normal est effectué. Cela inclut l'horizon partagé (voyez la section 2.2.1) et le masquage de sous-réseaux (section 3.2), de sorte que certaines entrées de la table de routage ne seront pas montrées. Si la requête réclame des entrées spécifiques, elles sont recherchées dans la table de routage et l'information est retournée. Aucun traitement d'horizon partagé n'est effectué, et les sous-réseaux sont retournés si c'est requis. Nous prévoyons que ces requêtes vont être utilisées dans différents contextes. Quand un hôte démarre pour la première fois, il diffuse ses requêtes sur chaque réseau connecté en demandant une table de routage complète. En général, nous supposons que les tables de routage complètes vont être employées pour mettre à jour la table de routage d'un autre hôte. Pour cette raison, l'horizon partagé et tous les autres filtrages doivent être utilisés. Les requêtes pour des réseaux spécifiques ne sont faites que par des logiciels de diagnostic, et ne sont pas utilisées pour le routage. Dans ce cas, le requérant voudrait connaître le contenu exact de la base de données de routage, et ne voudrait pas qu'on lui cache la moindre information.

3.4.2 Response

Des réponses peuvent être reçues pour plusieurs raisons différentes :
- réponse à une question spécifique
- mises à jour régulières
- mises à jour déclenchées provoquées par un changement de métrique

Le traitement est identique quelle que soit la façon dont les réponses ont été générées.

Puisque le traitement d'une réponse peut mettre à jour la table de routage de l'hôte, la validité de la réponse doit être vérifiée avec soin. La réponse doit être ignorée si elle ne provient pas du port 520. L'adresse IP source devrait être examinée pour voir si le datagramme provient d'un voisin valide. La source du datagramme doit se situer sur un réseau directement connecté. Cela vaut également la peine de vérifier si la réponse provient de l'une des adresses propres de l'hôte. Les interfaces situées sur des réseaux à diffusion peuvent recevoir des copies de leur propre diffusion immédiatement. Si un hôte traite sa propre sortie comme une nouvelle entrée, une certaine confusion en résultera certainement, et de tels datagrammes doivent être ignorés (sauf dans les circonstances formulées dans le prochain paragraphe).

Avant de traiter réellement une réponse, il peut être utile d'utiliser son existence comme entrée pour un processus gardant une trace du statut de l'interface. Comme mentionné plus haut, nous invalidons une route quand nous n'avons pas reçu de nouvelles de sa passerelle depuis une certaine période de temps. Cela fonctionne bien pour les routes provenant d'une autre passerelle. Il est également souhaitable de savoir quand l'un de nos réseaux directement connectés est tombé en panne. Ce document ne spécifie aucune méthode particulière pour faire cela, car de telles méthodes dépendent des caractéristiques du réseau et de l'interface réseau attachée. Néanmoins, de telles méthodes impliquent souvent d'écouter les datagrammes arrivant sur cette interface. Des datagrammes arrivants peuvent être utilisés comme une indication que cette interface fonctionne. Néanmoins, il y a lieu de faire attention, car il est possible qu'une interface tombe en panne de manière telle que des datagrammes d'entrée sont reçus, mais les datagrammes de sortie ne sont jamais envoyés avec succès.

Maintenant que le datagramme dans son ensemble a été validé, traitez ses entrées une à une. À nouveau, commencez en faisant la validation. Si la métrique est plus grande que

l'infini, ignorez l'entrée. (Cela devrait être impossible, si l'autre hôte fonctionne correctement. Des métriques incorrectes et autres erreurs de format devraient probablement provoquer des alertes ou être enregistrées.) Regardez ensuite l'adresse destination. Vérifiez l'identificateur de famille d'adresses. S'il ne possède pas une valeur attendue (p.ex. 2 pour les adresses Internet), ignorez l'entrée. Détectez maintenant différents types d'adresses inappropriées. Ignorez l'entrée si l'adresse est de classe D ou E, ou si elle est située sur le réseau 0 (sauf pour 0.0.0.0, si les routes par défaut sont acceptées), ou sur le réseau 127 (le réseau loopback [10]). Testez également si c'est une adresse de diffusion, c.-à-d. dont la partie hôte n'est composée que de chiffres 1 sur un réseau qui supporte la diffusion, et ignorez de telles entrées. Si l'implémenteur a choisi de ne pas supporter les routes d'hôtes (voir la section 3.2), vérifiez si la partie hôte de l'adresse est non nulle ; si c'est le cas, ignorez l'entrée.

Rappelez vous que le champ ' adresse ' contient un certain nombre d'octets inutilisés. Si le n° de version du datagramme est 1, ils doivent être examinés. Si l'un d'entre eux n'est pas nul, l'entrée doit être ignorée. (Beaucoup de ces cas indiquent que l'hôte initiateur du message ne fonctionne pas correctement. Par conséquent, une certaine forme d'enregistrement d'erreur ou d'alerte devrait être mise en place.)

Mettez à jour la métrique en ajoutant le coût du réseau par lequel le message est arrivé. Si le résultat est supérieur à 16, utilisez 16, c.-à-d.

$$\textit{métrique} = \min(\textit{métrique} + \textit{coût}, 16)$$

Examinez maintenant l'adresse pour voir s'il y a déjà une route la rejoignant. En général, si ce n'est pas le cas, il faut en ajouter une. Néanmoins, il y a plusieurs exceptions. Si la métrique est infinie, n'ajoutez pas d'entrée. (On pourrait en mettre à jour une existante, mais on n'ajoute pas de nouvelles entrées possédant une métrique infinie.) Il faut éviter d'ajouter des routes vers des hôtes si l'hôte fait partie d'un réseau ou sous-réseau pour lequel nous disposons d'une route au moins aussi bonne. Si aucune de ces exceptions ne s'applique, ajoutez une nouvelle entrée dans la base de données de routage. Ceci inclut les actions suivantes :

- Définir la destination et la destination à partir du datagramme.
- Fixer la passerelle à l'hôte à l'origine du datagramme.
- Initialiser la temporisation pour la route. Si le temporisateur de ramassage des déchets est en cours de fonctionnement pour cette route, arrêtez-le. (Voyez la section 3.3 pour une discussion sur les temporisateurs.)
- Définir le drapeau de changement de route et aviser le processus de sortie de déclencher une mise à jour (voyez 3.5).

S'il y a une route existante, comparez d'abord les passerelles. Si ce datagramme provient de la même passerelle que la route existante, réinitialisez la temporisation. Ensuite, comparez les métriques. Si le datagramme provient de la même passerelle que la route existante et que la nouvelle métrique est différente de l'ancienne, ou si la nouvelle métrique est plus petite que l'ancienne, effectuez les actions suivantes :

- Adoptez la route provenant du datagramme, c.-à-d. placez-y la nouvelle métrique, et affectez la passerelle au nom d'hôte duquel provenait le datagramme.
- Initialisez la temporisation pour la route.

[10] boucle locale

– Établissez le drapeau de changement de route, et signalez au processus de sortie de déclencher une mise à jour (voir 3.5).
– Si la nouvelle métrique est 16 (infini), le processus d'effacement est entamé.

Si la nouvelle métrique est 16 (infini), cela démarre le processus d'effacement de la route. La route n'est alors plus utilisée pour router les paquets, et le temporisateur d'effacement est démarré (voir section 3.3). Notez qu'un effacement n'est entamé que si la métrique est d'abord fixée à 16. Si la métrique valait déjà 16, alors une nouvelle suppression n'est pas entreprise. (Entamer un effacement déclenche un temporisateur. L'ennui est que nous ne voulons pas réinitialiser le temporisateur toutes les 30 secondes, car de nouveaux messages arrivent avec une métrique infinie.)

Si la nouvelle métrique est identique à l'ancienne, le plus simple est de ne rien faire de plus (au-delà de réinitialiser la temporisation, comme spécifié plus haut). Néanmoins, le `routed` 4BSD utilise une heuristique supplémentaire ici. Normalement, il est absurde de passer à une route de même métrique que la route existante mais avec une passerelle différente. Néanmoins, si la route existante montre des signes de dépassement possible du délai, il peut être préférable de passer immédiatement à une route alternative de même valeur, plutôt que d'attendre que la temporisation se produise. (Voyez la section 3.3 pour une discussion sur les temporisations.). Par conséquent, si la nouvelle métrique est identique à l'ancienne, `routed` examine l'état de la temporisation pour la route existante. Si on est au moins à mi-chemin de l'expiration de sa période de validité, `routed` passe à la nouvelle route, c.-à-d. que la passerelle est remplacée par la source du message actuel. Cette heuristique est optionnelle.

Toute entrée échouant à ces tests est ignorée, car elle n'est pas meilleure que la route actuelle.

3.5 Traitement de la sortie

Cette section décrit le traitement utilisé pour créer des messages de réponse contenant tout ou partie de la table de routage. Ce traitement peut être déclenché de l'une des manières suivantes :

– par traitement de l'entrée quand une requête est rencontrée. Dans ce cas, le message résultant n'est envoyé qu'à une seule destination.
– par la mise à jour régulière du routage. Toutes les 30 secondes, une réponse contenant la table de routage entière est envoyée à chaque passerelle voisine (Voir section 3.3)
– par des mises à jour déclenchées. À chaque fois que la métrique d'une route est modifiée, une mise à jour est déclenchée. (La mise à jour peut être retardée ; voyez plus bas).

Avant de décrire la façon dont un message est généré pour chaque réseau directement connecté, nous commenterons la façon dont les destinations sont choisies pour les deux derniers cas. Normalement, quand une réponse doit être envoyée à toutes les destinations (c.-à-d. soit la mise à jour régulière, soit une mise à jour déclenchée est en cours de préparation), une réponse est envoyée à l'hôte situé à l'autre bout de chaque liaison point-à-point, et une réponse est diffusée sur tous les réseaux connectés supportant la diffusion. Ainsi donc, une réponse est préparée pour chaque réseau directement connecté et envoyée à l'adresse correspondante (de la destination ou de diffusion). Dans la plupart des cas, cela atteint toutes les passerelles voisines. Néanmoins, il y a certains cas où cela peut ne pas être assez bien. Cela peut impliquer un réseau qui ne supporte pas la diffusion (p.ex. ARPANET), ou une

situation impliquant des passerelles stupides. Dans de telles circonstances, il peut être nécessaire de spécifier une liste réelle de passerelles et hôtes voisins, et d'envoyer explicitement un datagramme à chacun d'entre eux. Il revient à l'implémenteur de décider si un tel mécanisme est nécessaire, et le cas échéant de définir comment la liste est spécifiée.

Les mises à jour déclenchées requièrent un traitement spécial pour deux raisons. Premièrement, l'expérience montre que les mises à jour déclenchées peuvent provoquer des charges excessives sur des réseaux de capacité limitée ou comportant trop de passerelles. Le protocole requiert donc que les implémenteurs prennent des dispositions pour limiter la fréquence des mises à jour déclenchées. Après l'émission d'une mise à jour déclenchée, un temporisateur devrait être démarré pour un temps aléatoire compris entre 1 et 5 secondes. Si d'autres changements susceptibles de déclencher des mises à jour se produisent avant que le temporisateur n'expire, une simple mise à jour est déclenchée quand le temporisateur expire, et le temporisateur est ensuite fixé à une autre valeur aléatoire comprise entre 1 et 5 secondes. Une mise à jour déclenchée peut être supprimée si une mise à jour régulière est prévue avant que la mise à jour déclenchée ne soit envoyée.

Deuxièmement, les mises à jour déclenchées n'ont pas besoin d'inclure la table de routage entière. En principe, seules les routes qui ont été modifiées doivent être incluses. Les messages générés faisant partie d'une mise à jour déclenchée doivent donc au moins inclure les routes dont le drapeau de changement de route est défini. Ils peuvent inclure des routes supplémentaires, ou toutes les routes à la discrétion de l'implémenteur ; néanmoins, quand des mises à jour de routage complètes requièrent de multiples paquets, l'envoi de toutes les routes est fortement découragé. Quand une mise à jour déclenchée est traitée, les messages devraient être générés pour chaque réseau directement connecté. Le traitement de l'horizon partagé est effectué lors de la génération de mises à jour déclenchées aussi bien que lors des mises à jour normales (voir plus bas). Si, après le traitement de l'horizon partagé, une route modifiée devait apparaître identique à son état antérieur, la route ne doit pas être envoyée ; si, en conséquence, aucune route ne doit être envoyée, la mise à jour peut être omise sur ce réseau. (Si une route n'a subi qu'un changement de métrique, ou utilise une nouvelle passerelle située sur le même réseau que l'ancienne passerelle, la route sera envoyée sur le réseau de la vieille passerelle avec une métrique infinie à la fois avant et après le changement.) Une fois que toutes les mises à jour déclenchées ont été générées, les drapeaux de changement de route devraient être réinitialisés.

Si le traitement de l'entrée est autorisé alors que la sortie est en train d'être générée, un inter-verrouillage approprié doit être mis en œuvre. Les drapeaux de changement de route ne devraient pas être modifiés à la suite du traitement de l'entrée quand un message de mise à jour déclenchée est en cours de génération.

La seule différence entre une mise à jour déclenchée et les autres messages de mise à jour est la possible omission des routes qui n'ont pas changé. Les mécanismes restants, qui sont sur le point d'être décrits, doivent tous s'appliquer aux mises à jour déclenchées.

Voici la façon dont un datagramme de réponse est généré pour un réseau directement connecté particulier :

L'adresse IP source doit être celle de l'hôte émetteur sur ce réseau. C'est important parce que l'adresse source est placée dans les tables de routage d'autres hôtes. Si une adresse source incorrecte est employée, d'autres hôtes peuvent être incapables de router les datagrammes. Parfois, les passerelles sont configurées avec plusieurs adresses IP sur une même interface

physique. Normalement, cela signifie que plusieurs réseaux IP logiques sont transportés sur un même médium physique. Dans de tels cas, un message de mise à jour séparé doit être envoyé à chaque adresse, avec cette adresse comme adresse IP source.

Spécifiez le numéro de version de la version actuelle de RIP. (La version décrite dans ce document est la 1.) Fixez la commande à ' response '. Fixez les octets marqués « doit être nul » à zéro. Maintenant, commencez à remplir les entrées.

Pour remplir les entrées, parcourez de haut en bas toutes les routes de la table de routage interne. Rappelez-vous que la taille maximale d'un datagramme est de 512 octets. Quand il n'y a plus d'espace pour le datagramme, envoyez le message actuel et démarrez-en un nouveau. Si une mise à jour déclenchée est en cours de génération, seules les entrées dont les drapeaux de changement de route sont spécifiés doivent être incluses.

Voyez la description dans la Section 3.2 pour une discussions sur les problèmes soulevés par les routes vers des sous-réseaux ou d'hôtes. Les routes vers des sous-réseaux n'auront aucune signification à l'extérieur du réseau, et doivent être omises si la destination n'est pas sur le même réseau composé de sous-réseaux ; elles devraient être remplacées par une unique route vers le réseau duquel font partie les sous-réseaux. De la même façon, les routes d'hôtes doivent être éliminées si elles sont synthétisées par une route de réseau, comme décrit dans la discussion de la Section 3.2.

Si la route passe ces tests, alors les destination et métrique sont placées à l'intérieur de l'entrée dans le datagramme de sortie. Les routes doivent être incluses dans le datagramme même si leur métrique est infinie. Si la passerelle de la route est située sur le réseau pour lequel le datagramme est préparé, la métrique de l'entrée est fixée à 16, ou bien l'entrée entière est omise. L'omission de l'entrée revient simplement à faire de l'horizon partagé. L'inclusion d'une entrée de métrique égale à 16 est de l'horizon partagé avec empoisonnement. Voyez la Section 2.2 pour une discussion plus complète sur ces alternatives.

3.6 Compatibilité

Le protocole décrit dans ce document est destiné à interopérer avec `routed` et d'autres implémentations de RIP. Néanmoins, un point de vue différent est adopté par rapport au moment de l'incrémentation de la métrique que celui choisi dans la plupart des implémentations précédentes. En utilisant la perspective précédente, la table de routage interne a une métrique de 0 pour tous les réseaux directement connectés. Le coût (qui est toujours de 1) est ajouté à la métrique quand la route est envoyée dans un message de mise à jour. En contraste, dans ce document, les réseaux directement connectés apparaissent dans la table de routage interne avec des métriques égales à leur coût ; les métriques ne valent pas nécessairement 1. Dans ce document, le coût est ajouté aux métriques quand des routes sont reçues dans des messages de mise à jour. Les métriques de la table de routage sont envoyées dans des messages de mise à jour sans modification (à moins que cela ne soit altéré par l'horizon partagé).

Ces deux points de vue résultent en une émission de messages de mise à jour identiques. Les métriques présentes dans la table de routage diffèrent d'une constante « 1 » dans les deux descriptions. En fait, il n'y a donc aucune différence. Le changement a été effectué car les nouvelles descriptions facilitent le traitement des situations où différentes métriques sont utilisées sur des réseaux directement attachés.

Les implémentations qui ne supportent que des coûts de réseaux unitaires ne doivent pas être modifiées pour se conformer au nouveau style de présentation. Néanmoins, elles doivent de toute manière suivre les autres prescriptions fournies dans ce document.

4 Fonctions de contrôle

Ces section décrit les contrôles administratifs. Ils ne font pas partie du protocole en tant que tels. Néanmoins, l'expérience engrangée avec les réseaux existants suggère qu'ils sont importants. Puisqu'ils ne font pas nécessairement partie du protocole, ils sont considérés optionnels. Néanmoins, nous recommandons fortement qu'au moins quelques uns d'entre eux soient inclus dans chaque implémentation.

Ces contrôles sont prévus principalement pour permettre à RIP d'être connecté à des réseaux dont le routage peut être instable ou sujet aux erreurs. Voici quelques exemples :

Il est parfois désirable de limiter les hôtes et passerelles depuis lesquels on peut accepter des informations. À l'occasion, des hôtes ont été mal configurés d'une façon telle qu'ils commencent à envoyer des informations inappropriées.

Un certain nombre de sites limitent le groupe de réseaux qu'ils permettent dans les messages de mise à jour. Une organisation A pourrait disposer d'une connexion vers une organisation B qu'elles utilisent pour une communication commune directe. Pour des raisons de sécurité ou de performance, A pourrait ne pas vouloir permettre l'utilisation de cette connexion par d'autres organisations. Dans de tels cas, A ne devrait pas inclure les réseaux de B dans les mises à jour que A envoie à des parties tierces.

Voici quelques contrôles typiques (notez néanmoins que le protocole RIP ne les requiert pas au même titre que d'autres contrôles) :

- une liste de voisins - l'administrateur réseau devrait pouvoir définir une liste de voisins pour chaque hôte. Un hôte n'accepterait de messages de réponse que des hôtes présents dans sa liste de voisins.
- permettre ou interdire des destinations spécifiques - l'administrateur réseau devrait pouvoir spécifier une liste d'adresses destination à autoriser ou à interdire. La liste serait associée à une interface particulière dans les sens entrant et sortant. Seuls les réseaux permis seraient mentionnés dans les messages de réponse sortants ou traités parmi les messages de réponse entrants. Si une liste d'adresses autorisées est spécifiée, toutes les autres adresses sont interdites. Si une liste d'adresses interdites est spécifiée, toutes les autres adresses sont permises.

Bibliographie

[1] R.E Bellman, *Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.

[2] D.P. Bertsekas, R.G. Gallaher, *Data Networks*, Prentice-Hall, Englewood Cliffs, N.J., 1987.

[3] R. Braden, J. Postel, *Requirements for Internet Gateways*, USC/Information Sciences Institute, RFC-1009, juin 1987.

[4] D.R. Boggs, J.F. Shoch, A.E. Taft, R.M. Metcalfe, *Pup : An Internetwork Architecture*, IEEE Transactions on Communications, avril 1980.

[5] D.D Clark, *Fault Isolation and Recovery*, MIT-LCS, RFC-816, juillet 1982.

[6] L.R. Ford Jr., D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, N.J., 1962.

[7] Xerox Corp., *Internet Transport Protocols*, Xerox System Integration Standard XSIS 028112, décembre 1981.