Protocoles Réseau V

Juliusz Chroboczek

9 octobre 2018

1 Préliminaires

1.1 Tables de routage

Pour chaque paquet qu'elle manipule, la couche réseau effectue une décision de routage qui dépend de la destination d'un paquet. Pour cela, elle consulte une structure de données — la *table de routage*, qui conceptuellement est une fonction qui à toute adresse IP de destination affecte un voisin, c'est-à-dire une paire $(\text{interface}, \text{IP})$:

$$\text{D} \quad \mapsto \quad (\text{if}, \text{nh})$$

Il n'est bien-sûr pas raisonnable de représenter une table de routage de façon exhaustive — en IPv4 il faudrait plusieurs gigaoctets de mémoire, sans même parler d'IPv6. La table de routage est représentée par une structure qui à un préfixe associe soit un voisin soit une interface ; dans ce dernier cas, le voisin associé est directement connecté, et l'adresse du voisin est égale à celle de la destination :

$$\begin{array}{lll} \text{P} & \mapsto & (\text{if}, \text{nh}) \\ \text{P} & \mapsto & \text{if} \end{array}$$

Par exemple, la table de routage suivante envoie les paquets du /24 à une destination directement connectée, et les autres à un routeur par défaut :

$$\begin{array}{lll} 203.0.113.0/24 & \mapsto & \text{eth0} \\ 0.0.0.0/0 & \mapsto & (\text{eth0}, 203.0.113.1) \end{array}$$

La table de routage est une structure de données ambigüe : deux préfixes peuvent se recouvrir, et donc plusieurs entrées peuvent s'appliquer à la même adresse de destination. Cependant, si deux préfixes distincts ont une intersection non-nulle, alors l'un est inclus dans l'autre — il y en a donc un qui est plus spécifique que l'autre, et c'est celui-ci qui s'applique. On appelle cette règle la *règle du préfixe le plus long (longest-prefix rule)*.

La table de routage peut être construite manuellement, mais en pratique elle est construite automatiquement, par un *protocole de routage*.

1.2 Modèles de lien

Dans le *modèle par préfixe* ou *modèle Internet* d'un réseau IP, on affecte un préfixe distinct à chaque lien, et toutes les interfaces connectées à ce lien ont des adresses IP contenues dans ce préfixe ; les nœuds eux-mêmes n'ont pas d'adresses IP. Le protocole de routage ne manipule pas d'adresses individuelles, mais des préfixes, et c'est ARP ou ND qui est chargé du routage au sein d'un lien.

Il existe un autre modèle, le *modèle mesh*. Dans ce modèle, chaque nœud se voit attribuer une adresse (les interfaces n'en ont pas), et le protocole de routage manipule des adresses individuelles. Clairement, le modèle mesh annonce plus d'informations à travers le protocole de routage, mais il permet la mobilité des nœuds sans renuméroter. En pratique, un protocole de routage capable de manipuler des prefixes de longueur arbitraire est aussi capable de manipuler des routes de hôtes (/32 ou /128), et donc de fonctionner dans le modèle mesh (filaire — les meshes sans fil demandent d'être capable de gérer des liens non transitifs, ce que le plupart des protocoles de routage ne savent pas faire).

Le modèle mesh est plus simple que le modèle par préfixes, et pour alléger les notations c'est dans celui-ci que je présente les protocoles de routage. Cependant, tous les algorithmes que je présente se généralisent au modèle Internet.

2 L'algorithme d'accessibilité naïf

L'algorithme de routage le plus simple — et qui ne fonctionne pas en général — a été utilisé dans le protocole EGP. C'est un parcours de graphe distribué qui n'a pas vraiment de nom, je l'appelle l'*algorithme d'accessibilité naïf*.

2.1 Première formulation

Chaque routeur X maintient la table de routage $R(X)$, qui à chaque destination S associe un voisin nh. On note $nh_S(X)$ la valeur associée à S dans le routeur X, c'est-à-dire le voisin qu'utilise X pour faire suivre les paquets destinés à S.

Initialement, la table de routage ne contient que l'entrée $\mathrm{nh}_Y(Y) = Y$. Périodiquement, un routeur Y envoie à tous ses voisin une *annonce* qui contient l'ensemble $\mathrm{dom}(R(X))$, c'est-à-dire l'ensemble des nœuds que X sait joindre.

Lorsqu'un routeur X reçoit une annonce $R(Y)$ de la part d'un voisin Y, pour chaque $S \in R(Y)$,

- si $S \notin R(X)$, alors $\mathrm{nh}_S(X) := Y$;
- sinon rien.

Exemple de convergence On considère la topologie de la figure 1, et on s'intéresse à la convergence du protocole ci-dessus. L'algorithme peut évoluer par exemple de la façon suivante :

S	(S,S)	(S,S),(A,A)	(S,S),(A,A)	(S,S),(A,A),(B,A)	⋯
A	(A,A)	(A,A)	(S,S),(A,A),(B,B)	(S,S),(A,A),(B,B)	⋯
B	(B,B)	(A,A),(B,B)	(A,A),(B,B)	(S,A),(A,A),(B,B)	⋯
C	(C,C)	(A,A),(C,C)	(A,A),(B,B),(C,C)	(S,A),(A,A),(B,B)	⋯

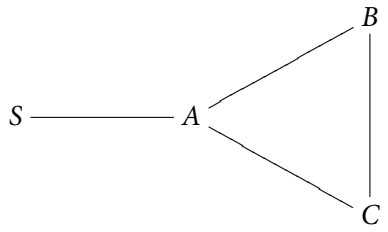

Figure 1 — Une topologie

2.2 Deuxième formulation

La formulation de l'algorithme d'accessibilité naïf ci-dessus est un peu compliquée — elle parle de listes de voisins, ce qui n'est pas forcément facile à manipuler. Une façon de raisonner que je préfère consiste à fixer une destination S, et de considérer l'algorithme comme s'exécutant simultanément pour tous les S du réseau. Comme S est fixé, on peut écrire $\mathrm{nh}(X)$ pour $\mathrm{nh}_S(X)$.

- Initialement $\mathrm{nh}(S) = S$, $\mathrm{nh}(X) = \bot$.
- Assez souvent, si $\mathrm{nh}(Y) \neq \bot$ alors Y envoie une annonce pour S.
- Lorsque X reçoit une annonce pour S de la part d'Y, alors :
  - si $\mathrm{nh}(X) = \bot$, alors $\mathrm{nh}(X) := Y$;
  - sinon rien.

Dans la topologie de la figure 1, l'algorithme peut évoluer de la façon suivante :

S	nh = S	nh = S	nh = S	nh = S
A	nh = $\bot$	nh = S	nh = S	nh = S
B	nh = $\bot$	nh = $\bot$	nh = A	nh = A
C	nh = $\bot$	nh = $\bot$	nh = A	nh = A

2.3 *Timeout*

Considérons ce qui se passe si le lien entre A et C casse (on débranche un cable Ethernet, ou alors les conditions de propagation radio changent). La topologie est alors celle de la figure 2.

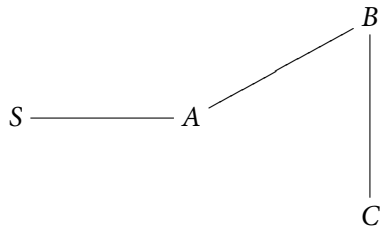

Figure 2 — Une autre topologie

On voudrait que le routeur C reroute les données destinées à S à travers B ; mais le protocole défini précédemment n'a aucun mécanisme pour changer une route qui ne marche plus. La solution habituelle consiste à éliminer une route à travers un voisin qu'on n'a pas entendu depuis un moment.

– Si X n'entend plus Y pendant un temps t_{hold}, alors $\text{nh}(S) := \bot$.

La valeur t_{hold} est typiquement choisie pour réagir à la perte de deux ou trois paquets de suite (des algorithmes plus raffinés sont parfois utilisés, voyez le paragraphe 4.4 plus bas).

Avec cette amélioration, le comportement dans la topologie de la figure 2 est le suivant :

S	nh = S	nh = S	nh = S
A	nh = S	nh = S	nh = S
B	nh = A	nh = A	nh = A
C	nh = A	nh = $\bot$	nh = B

Bien évidemment, il est possible d'optimiser le protocole pour éviter l'étape où C n'a pas de route vers S, au prix d'une consommation de mémoire plus élevée ; voyez le paragraphe 4.4.

Si le lien retourne à l'état précédent (on revient à la topologie de la figure 1), le protocole naïf continue à router à travers B. Nous verrons que ce n'est pas le cas du protocole à vecteur de distances (paragraphe 4).

2.4 Boucles de routage persistantes

Supposons maintenant que c'est le lien entre S et A qui casse : la topologie est celle de la figure 3.

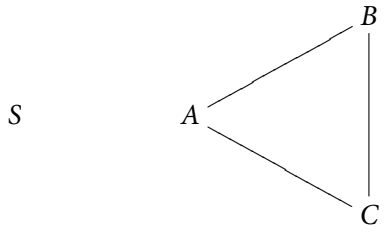

Figure 3 — Encore une topologie

Le protocole peut évoluer de la façon suivante :

S	nh = S	nh = S	nh = S
A	nh = S	nh = ∞	nh = B
B	nh = A	nh = A	nh = A
C	nh = A	nh = $\bot$	nh = B

Lorsque A expire le voisin S, B est encore en train d'annoncer que S est accessible ; A bascule donc sur B et crée donc une boucle de routage. A et B annoncent alors tous deux l'accessibilité de S, et confirment donc mutuellement leur accessibilité : la boucle ne sera jamais éliminée, elle est *persistante*. Si la technique de l'horizon scindé (paragraphe 4.4) permet d'éliminer de telles boucles à deux, elle ne fait rien pour éliminer la boucle analogue à trois.

Digression historique Bien qu'il ne fonctionne pas en général, le protocole d'accessibilité naïf a été utilisé par EGP, le deuxième protocole de routage externe de l'Internet. À l'époque, l'Internet consistait d'une *backbone* (routée en états de lien), le NFSnet, et les autres systèmes autonomes étaient directement connectés à la *backbone*. Même dans cette topologie dégénérée, EGP causait des problèmes.

Digression historique dans la digression historique Le premier protocole de routage externe de l'Internet s'appelait GGP, et était un protocole à vecteur de distances pur (paragraphe 4). Le troisième protocole de routage externe, encore utilisé aujourd'hui, s'appelle BGP, et c'est un protocole à vecteur de chemins (paragraphe 7).

3 Métriques

Une *métrique* est une valeur qu'on affecte à une route (une suite de liens) et qui spécifie la désirabilité de cette route : plus la métrique est faible, et plus le lien est désirable. Une métrique est généralement calculée en affectant une valeur à chaque lien, son *coût* ; la métrique d'une route est alors la somme des coûts des liens qui la composent. Un protocole de routage calcule généralement les chemins de métrique minimale[1]. On note c_{XY} le coût du lien allant de X à Y ; s'il n'y a pas de lien direct entre X et Y, $c_{XY} = \infty$. Les liens ne sont pas symétriques en général : il se peut que $c_{XY} \neq c_{YX}$ (pensez à un lien radio avec des émetteurs de puissance différente).

La métrique la plus simple est la métrique du nombre de sauts *(hop-count metric)*, qui est définie par $c_{XY} = 1$; la métrique d'une route est alors le nombre de liens qui la composent, et le protocole de routage choisit les routes ayant le moindre nombre de liens. Cette métrique très simple fonctionne relativement bien dans des réseaux filaires où tous les liens sont à peu près identiques. Par contre, elle fonctionne mal dans des réseaux sans fil, où elle a tendance à choisir des liens longs, et donc de mauvaise qualité.

Le plus souvent, les implémentations des protocoles de routage utilisent une métrique *administrativement définie* (ou *statiquement définie*) : les routeurs utilisent par défaut une métrique constante (ou alors qui dépend de la nature du lien — un Ethernet à un Gigabit a une métrique plus petite que celle d'une ligne ADSL) qui est paramétrable. L'administrateur humain observe le réseau, et ajuste manuellement les métriques jusqu'à ce que le traffic de données suive les bons chemins.

Dans les réseaux sans fil, il semblerait naturel de baser la métrique sur des données de couche physique (la puissance du signal reçu, le rapport signal-bruit, etc.). Ces techniques ne semblent pas très bien marcher — on n'a pas encore réussi à trouver une donnée de couche physique qui soit un bon prédicteur du taux de perte. En pratique, on utilise plutôt des données de couche lien, notamment le taux moyen de perte de paquets (qui s'avère être un excellent prédicteur du taux de perte de paquets) et le débit.

Une autre mesure parfois utilisée pour construire une métrique est la latence du lien, qu'on peut mesurer soit à la couche lien soit à la couche réseau (ce qui ne donne pas forcément le même résultat du fait de la présence de tampons supplémentaires à la couche réseau).

1. Ce qui est différent de l'arbre couvrant de poids minimal.

Ces techniques de calcul dynamique de la métrique ont tendance à causer une boucle de rétroaction *(feedback loop)* : plus un lien est désirable, et plus il devient congestionné, ce qui augmente son taux de perte ou sa latence, et du coup le rend moins désirable. Une telle boucle de rétroaction peut causer des oscillations du routage, et il faut donc employer des techniques qui en limitent la fréquence (hysteresis).

Espace de métriques Les métriques vivent dans un espace borné — elles sont codées sur un nombre fini de bits, et la valeur maximale est interprétée comme l'infini. Par exemple, le protocole RIP utilise des métriques de 4 bits ($\infty = 15$), tandis que OSPF utilise des métriques de 16 bits ($\infty = 65535$).

Plus généralement, les coûts et métriques peuvent appartenir à un espace arbitraire, pas forcément un ensemble d'entiers, équippé d'une opération d'addition (un coût ajouté à une métrique donne une métrique) et d'un ordre, ce qui peut aussi être représenté comme un semi-anneau (voyez la digression dans la digression du paragraphe 4.3).

4 Vecteur de distances

Le protocole à *vecteur de distances*, ou *Bellman-Ford distribué*, est un raffinement du protocole d'accessibilité simple qui distribue des métriques au lieu d'une simple information d'accessibilité. En plus de son *next-hop* sélectionné $\mathrm{nh}_S(X)$, chaque nœud maintient une estimation $\mathrm{d}_S(X)$ de la métrique de la route sélectionnée. Comme on a fixé S, on peut se permettre d'écrire $\mathrm{nh}(X)$ et $\mathrm{d}(X)$ pour $\mathrm{nh}_S(X)$ et $\mathrm{d}_S(X)$ respectivement.

- Initialement, $\mathrm{d}(S) = 0$ et $\mathrm{d}(X) = \infty$.
- Assez souvent, si $\mathrm{d}(Y) < \infty$, Y annonce $\mathrm{d}(Y)$ à ses voisins.
- Quand X reçoit $\mathrm{d}(Y)$:
  - si $\mathrm{nh}(X) = Y$, alors $\mathrm{d}(X) := c_{XY} + \mathrm{d}(Y)$;
  - si $c_{XY} + \mathrm{d}(Y) < \mathrm{d}(X)$, alors $\mathrm{d}(X) := c_{XY} + \mathrm{d}(Y)$ et $\mathrm{nh}(X) := Y$.
- *Timeout* : si $nh(X) = Y$, et X n'entend plus Y , $\mathrm{d}(X) := \infty$ et $\mathrm{nh}(X) := \perp$.

4.1 Exemples

Dans la topologie de la figure 1, le protocole peut évoluer de la façon suivante :

S	0	0	0	0
A	∞	1, nh = S	1, nh = S	1, nh = S
B	∞	∞	2, nh = A	2, nh = A
C	∞	∞	2, nh = A	2, nh = A

Lorsque le lien entre A et C casse (figure 2), le protocole évolue de la façon suivante :

S	0	0	0
A	1, nh = S	1, nh = S	1, nh = S
B	2, nh = A	2, nh = A	2, nh = A
C	2, nh = A	∞	3, nh = B

Enfin, lorsqu'on revient à la topologie de la figure 1, le comportement est le suivant :

S	0	0
A	1, nh = S	1, nh = S
B	2, nh = A	2, nh = A
C	3, nh = B	2, nh = A

Dès la première annonce envoyée par A et reçue par C, ce dernier bascule sur la route optimale.

4.2 Comptage à l'infini

Que se passe-t-il lorsque c'est le lien entre S et A qui casse (topologie de la figure 3) dans le protocole à vecteurs de distance ? Une évolution possible est la suivante :

S	0	0	0	0	0	0	⋯
A	1, nh = S	∞	3, nh = B	3, nh = B	3, nh = B	3, nh = B	⋯
B	2, nh = A	2, nh = A	2, nh = A	3, nh = C	3, nh = C	3, nh = B	⋯
C	2, nh = A	2, nh = A	2, nh = A	2, nh = A	∞	4, nh = A	⋯

Les routeurs forment une boucle, à deux ou trois, selon les circonstances ; à chaque annonce, la métrique augmente de 1. On dit que les routeurs « comptent à l'infini ». La boucle est brisée lorsque la métrique atteint l'infini.

De ce fait, l'algorithme naïf à vecteurs de distance n'est applicable que pour les petites valeurs de l'infini. Par exemple, le protocole RIP utilise une métrique où l'infini vaut 15 (la métrique est codée sur 4 bits).

4.3 Digression : l'algorithme de Bellman et Ford

L'algorithme de Bellman et Ford (BF) est un algorithme classique de calcul des plus court chemin dans un graphe. Il en existe plusieurs formulations, je vous propose la suivante.

Soit A la matrice d'adjacence d'un graphe (A_{ij} est c_{XY}, où i et j sont les indices de X et Y respectivement). L'algorithme opère sur une matrice M, où M_{ij} est l'estimation courante de la distance entre les nœuds d'indices i et j (M_{ij} est $\mathrm{d}_Y(X)$).

Initialement, M est la matrice qui a 0 sur la diagonale et ∞ ailleurs :

$$M = \begin{bmatrix} 0 & \infty & \cdots & \infty \\ \infty & 0 & \cdots & \infty \\ \vdots & & \ddots & \vdots \\ \infty & \infty & \cdots & 0 \end{bmatrix}$$

À chaque étape, pour tout i et j, on fait

$$M_{ij} := \min_k (A_{ik} + M_{kj}).$$

L'algorithme termine après n étapes, où n est le nombre de nœuds dans le graphe. S'il existe encore i, k, j tels que $A_{ik} + M_{kj} < M_{ij}$, le graphe contenait un cycle de poids négaitif ; sinon M_{ij} contient

le poids du chemin de poids minimal du nœud *i* au nœud *j*. (On peut interrompre l'itération dès que plus rien ne change, ce qui dans un graphe sans cycles de poids négatif arrivera au bout d'un nombre d'étapes égal au plus au diamètre.)

L'algorithme de Bellman-Ford diffère de l'algorithme à vecteurs de distance de deux façons. Tout d'abord, dans Bellman-Ford les entrées de la matrice M ne peuvent que décroître, à la différence du vecteur de distances, où une métrique peut croître lorsque le *next-hop* sélectionné renie la métrique précédemment annoncées : BF ne gère pas les topologies qui changent dynamiquement. Ensuite, BF exécute chaque étape du calcul de façon synchrone, alors que l'algorithme à vecteurs de distance ne fait pas d'hypothèses sur l'ordre dans lequel les annonces sont envoyées.

L'algorithme à vecteurs de distances est parfois appelé *Bellman-Ford distribué*.

Digression dans la digression Notons $\otimes$ l'opération $+$, et $\oplus$ l'opération du minimum. Une étape de calcul de Bellman-Ford s'écrit alors :

$$M_{ij} := \bigoplus_k (A_{ik} \otimes M_{kj}).$$

On remarque alors qu'une étape de Bellman-Ford est une multiplication de matrices dans le semi-anneau $(+, \min)$:

$$M := A \otimes M.$$

On peut montrer que l'algorithme de Bellman-Ford s'applique à tout semi-anneau distributif à gauche. Voyez Seweryn Dynerowicz et Tim Griffin, *On the Forwarding Paths Produced by Internet Routing Algorithms*, 2013.

4.4 Optimisations

L'algorithme à vecteurs de distance a la propriété que même sa version naïve fonctionne en pratique, tout au moins dans des réseaux relativement stables et de taille modérée — le protocole RIP réalise, à peu de chose près, le protocole à vecteurs de distance naïf. Cependant, il existe un certain nombre d'optimisations qui permettent d'améliorer sa vitesse de convergence et la quantité d'informations de routage distribuées. À part le temps de silence, qui a d'autres problèmes, ces optimisations n'empêchent pas le comptage à l'infini — pour cela, il faut utiliser un algorithme sans boucles (paragraphe 8).

Horizon scindé L'algorithme du vecteur de distances annonce les métriques de toutes les routes qu'il connaît sur toutes les interfaces. L'*horizon scindé* est une amélioration qui consiste à éviter d'annoncer une route sur l'interface sur laquelle elle a été apprise ; dans l'algorithme du paragraphe 4, on remplace la deuxième ligne par :

– Assez souvent, si $\mathrm{d}(Y) < \infty$, Y annonce $\mathrm{d}(Y)$ sur toutes ses interfaces sauf celle sur laquelle pointe $\mathrm{nh}(Y)$.

L'horizon scindé est basé sur l'observation que si la route vers Y a été apprise sur une interface I, alors tous les nœuds du lien correspondant ont déjà reçu une annonce de métrique inférieure ; il

est donc inutile d'envoyer l'annonce. Il a pour effet d'éviter le comptage à l'infini à deux (mais pas à trois ou plus). Un autre effet bénéfique est qu'il diminue la taille des annonces.

L'horizon scindé n'est correct que sur des liens transitifs (où la relation « est voisin de » est transitive), ce qui n'est pas forcément le cas des liens radio.

Horizon scindé avec poison inverse *L'horizon scindé avec poison inverse* est une modification de l'horizon scindé où au lieu de supprimer les annonces sur l'interface où pointe la route, on fait des annonces de métrique infinie (des rétractions). Il a pour avantage de rompre une boucle de routage à deux sans nécessiter un *timeout* ; son défaut est qu'il augmente la taille des annonces.

Temps de silence Le *temps de silence (hold time)* est une modification à l'algorithme qui consiste à maintenir un temps de silence après que la métrique d'une route a augmenté. Si le temps de silence est supérieur au temps t_{hold} pendant lequel une route est maintenue en l'absence d'annonces, le *hold time* permet d'éviter la *race condition* qui mène au comptage à l'infini, et donc d'éviter les boucles. Il est rarement employé en pratique, car il ralentit dramatiquement la reconvergence du réseau.

Accessibilité explicite Le protocole à vecteur de distances surcharge les annonces : le fait de recevoir une annonce d'un voisin Y indique non seulement la présence d'une route mais aussi le fait que Y est encore accessible. Afin d'accélérer la convergence, il est utile d'avoir des messages d'accessibilité explicites, souvent appelés messages *Hello*, qui ne servent qu'à indiquer que Y est encore accessible. Les *Hello* sont de petits messages, il est donc possible de les envoyer plus souvent que les annonces ; lorsqu'un nombre suffisant de *Hellos* ont été perdus, Y est déclaré inaccessible, et toutes les routes qui passent par Y sont perdues sans attendre qu'elles expirent.

Accessibilité symétrique Le fait que le nœud X reçoit des annonces ou les *Hellos* de son voisin Y indique que X entend Y, ce qui n'implique pas forcément que Y entend X (pensez aux liens radio, ou aux *firewalls* mal configurés). Or, pour que X puisse router à travers Y, c'est l'accessibilité inverse qui est importante : il faut que Y entende X. Pour cette raison, les protocoles modernes combinent le sous-protocole d'accessibilité explicite avec un protocole d'accessibilité inverse, où le nœud Y indique explicitement à X qu'il entend les *Hellos* de X. Si cette indication n'est pas reçue par X, Y est jugé inaccessible même si X entend les *Hellos* de X.

Annonces déclenchées Le protocole à vecteur de distances naïf envoie des annonces de façon périodique ou pseudo-périodique, sans considérer les changements de table de routage. Pour accélérer la convergence, un nœud peut envoyer une annonce dès qu'il a détecté un changement de sa table de routage. Une telle annonce est dite *déclenchée* (*triggered update* par opposition à *scheduled update*).

Les annonces déclenchées améliorent dramatiquement les performances d'un protocole à vecteur de distances, par exemple en réduisant le temps nécessaire pour compter à l'infini de plusieurs minutes à quelques secondes. Cependant, elles sont difficiles à implémenter, et une implémentation trop naïve peut causer des *bursts* importants de traffic de routage.

5 Protocoles à états de lien

L'algorithme dominant dans le domaine du routage interne s'appelle l'*algorithme à états de lien*. L'état de lien consiste de trois sous-protocoles :

1. calcul de la topologie locale ;
2. inondation fiable ;
3. calcul de l'arbre des plus courts chemins et de la table de routage.

Calcul de la topologie locale Chaque routeur calcule l'ensemble de ses voisins symétriques en échangeant des paquets *Hello* avec eux. Par exemple, dans la topologie de la figure 1, le routeur B calcule la topologie locale $\{(B, A), (B, C)\}$.

Inondation fiable Lors de l'inondation fiable, chaque routeur inonde à travers tout le domaine de routage sa topologie locale ; à chaque fois que cette dernière change, il inonde de nouveau (l'algorithme d'inondation fiable est donc capable de mettre à jour une donnée précédemment inondée). Le résultat de l'inondation fiable est que tous les routeurs ont la même *base de données d'états de lien, (link-state database, LSDB)*. Par exemple, dans la topologie de la figure 1, la LSDB dupliquée par tous les routeurs a la forme $\{(S, A), (A, S), (A, B), (A, C), \ldots\}$).

Calcul des plus courts chemins La LSDB contient l'union de toutes les topologies locales, et donc la topologie globale du domaine de routage. Chaque routeur X calcule l'arbre des plus courts chemins enraciné en X (par exemple à l'aide de l'algorithme de Dijkstra). À chaque fois qu'une nouvelle entrée de LSDB est inondée, ce calcul est refait.

Cet arbre des plus courts chemins est ensuite tronqué juste après la racine pour obtenir la table de routage. Par exemple, dans la topologie de la figure 1, S calcule l'arbre (S, SA, SAB, SAC), ce dont il déduit la table de routage $(S \mapsto S, A \mapsto A, B \mapsto A, C \mapsto A)$.

5.1 Fragilité de l'état de liens naïf

L'algorithme naïf à états de liens, décrit au paragraphe précédent, dépend de trois miracles :

1. les bases de données de liens sont synchronisées ;
2. l'ensemble des plus courts chemins est un arbre ;
3. le recollage des arbres tronqués produit des routes sans boucles.

Le premier miracle est dû à l'implémentation soignée de l'inondation fiable, ainsi qu'à la limitation de la taille des domaines d'inondation (voir *aires* ci-dessous). Les deux autres miracles dépendent du fait que l'algèbre de routage est distributive.

5.2 Découpage en aires

Comme mentionné ci-dessus, l'algorithme naïf à état de liens est fragile. Les protocoles de routage les plus utilisés pour le routage interne, OSPF et IS-IS, le rendent plus robuste en découpant le domaine de routage en *aires*.

Dans un réseau OSPF, chaque routeur appartient à une ou plusieurs *aires*. Chacune des aires est de taille raisonnable (1000 routeurs est raisonnable, 10 000 ne l'est pas), ce qui permet à l'inondation fiable de converger rapidement, et à l'algorithme de Dijkstra de terminer en un tamps raisonnable.

Le routage entre les aires se fait en vecteur de distances. Il y a une restriction importante sur la topologie des aires : toute aire est voisine de l'aire 0, dite *backbone* ; cette restriction fait que le comptage à l'infini n'a pas lieu.

6 Routage hiérarchique

Originellement, l'Internet tout entier constituait un seul domaine de routage, routé avec un protocole à vecteur de distances (GGP). Lorsqu'un routeur se plantait à San Francisco, le changement de métrique induit était propagé globalement, et un routeur dans le Massachusetts devait mettre à jour sa métrique.

Depuis les années 1980, l'Internet est divisé en *systèmes autonomes (AS)*, des ensembles de routeurs administrés par une seule entité (par exemple, AS1307 s'appelle FR-U-JUSSIEU-PARIS). Le routage est hiérarchique : les AS exécutent un protocole de routage *externe* entre eux, et chaque AS exécute un protocole de routage *interne* en son sein. (Une façon de voir les choses est que les aires d'un protocole de routage interne à état de liens introduisent un troisième niveau de hiérarchie. On pourrait même dire que ARP/ND constitue un quatrième niveau.)

Chaque AS choisit le protocole de routage interne qu'il utilise; par contre, la communication entre AS est normalisée, et utilise le protocole BGP. Un changement de topologie à l'intérieur d'un AS n'est pas annoncée à l'extérieur de l'AS ; pour faire une annonce globale, il faut que la topologie externe change (par exemple du fait de la panne d'un *routeur de frontière*).

7 BGP : vecteur de chemins

BGP, le protocole utilisé pour le routage externe, utilise une variante du vecteur de distances qui s'appelle le *vecteur de chemins (path vector)*. En vecteur de chemins, un routeur n'annonce pas une métrique à ses voisins, comme en vecteur de distances, mais le chemin complet qu'a suivi l'annonce avant d'atteindre le récepteur. Dans la topologie de la figure 1, par exemple, le routeur S annonce S, le routeur A annonce $A \cdot S$, le routeur B annonce $B \cdot A \cdot S$.

Pour éviter les boucles, chaque routeur, avant d'accepter une annonce, vérifie s'il est présent dans le chemin annoncé ; si c'est le cas, il ignore l'annonce — l'accepter causerait une boucle. Une annonce est préférable à une autre si le chemin qu'elle porte est plus court : c'est la longueur du chemin qui sert de métrique.

Dans le style du paragraph 4, l'algorithme s'écrit comme suit :

- Initialement, $\text{path}(S) = \varepsilon$ et $\text{path}(X) = \bot$.
- Assez souvent, si $\text{path}(Y) \neq \bot$, Y annonce $Y \cdot \text{path}(Y)$ à ses voisins.
- Quand X reçoit un chemin p de Y :
  - si $\text{nh}(X) = Y$, alors $\text{path}(X) := p$;
  - si $|p| < |\text{path}(X)|$, alors $\text{path}(X) := p$.

– *Timeout* : si $\text{path}(X) = Y\cdots$, et X n'entend plus Y, alors $\text{path}(X) := \bot$.

Le vecteur de chemins évite les boucles[2] de façon efficace et simple. Son principal défaut est que les annonces ont une taille importante. BGP contourne ces problèmes en utilisant un transport fiable (TCP) et en n'envoyant des annonces que lorsque l'ensemble de routes a changé. En outre, comme BGP est un protocole de routage externe, il annonce la suite d'AS suivie par l'annonce plutôt que la suite de routeurs, ce qui réduit la taille du chemin et évite de reannoncer lorsqu'un reroutage a eu lieu à l'interieur d'un AS.

8 Vecteur de distances sans boucles

On peut voir le vecteur de chemins est un cas particulier d'algorithme à vecteur de distances sans boucle. Ces algorithmes sont paramétrisés par une *condition de faisabilité* : une annonce n'est acceptée que si elle satisfait la condition.

Si on note F_Y l'information de faisability annoncée par Y, et feasible la condition de faisabilité, un tel algorithme peut s'ecrire dans le style du paragraphe 4 :

– Initialement, $\text{d}(S) = 0$ et $\text{d}(X) = \infty$.
– Assez souvent, si $\text{d}(Y) < \infty$, Y annonce $(\text{d}(Y), F_Y)$ à ses voisins.
– Quand X reçoit $\text{d}(Y), F_Y$ et $\text{feasible}(F_Y, \text{d}(Y))$:
  – si $\text{nh}(X) = Y$, alors $\text{d}(X) := c_{XY} + \text{d}(Y)$;
  – si $c_{XY} + \text{d}(Y) < \text{d}(X)$, alors $\text{d}(X) := c_{XY} + \text{d}(Y)$ et $\text{nh}(X) := Y$.
– *Timeout* : si $nh(X) = Y$, et X n'entend plus Y , $\text{d}(X) := \infty$ et $\text{nh}(X) := \bot$.

Dans le cas de l'algorithme à vecteur de chemins, F_Y est le chemin suivi par l'annonce, et $\text{feasible}(F_Y, \text{d}(Y)) \equiv X \notin F_Y$.

Dans le protocole EIGRP (utilisant l'algorithme DUAL) ainsi que dans le protocole Babel, chaque routeur X maintient une *distance de faisabilité* $\text{fd}(X)$ définite comme étant le minimum de $\text{d}(X)$ au cours du temps :

$$\text{fd}(X) = \min_{t<\text{now}} \text{d}(X).$$

La condition de faisabilité est satisfaite si la métrique portée par l'annonce est strictement inférieure à la distance de faisabilité du récepteur :

$$\text{feasible}(\text{d}(Y)) \equiv \text{d}(Y) < \text{fd}(X).$$

La condition de faisabilité d'EIGRP et Babel est conservative : elle élimine parfois des routes qui pourtant ne causent pas une boucle. Ces protocoles ont donc besoin d'un mécanisme qui évite les famines ; EIGRP et Babel utilisent des mécanismes très différents.

2. En fait, une boucle transitoire peut avoir lieu, mais elle disparaît dès qu'une annonce a fait le tour de la boucle.