10: Couche routage

Martin Quinson

Épisode précédent

- TCP offre la fiabilité, UDP est en mode DIY pour le reste
  - mécanisme de base avec deux buffers sender/recver
  - Etablissement en 3-ways pour éviter les paquets du passé
- Controle de flux (préserver le recepteur)
  - fenêtre annoncée dans flux en sens inverse
  - Problématique de remplissage des paquets (Naggle et delayed ACK)
  - Perte de paquets (triple ACK=NACK; timeout)
- Controle de congestion (usage au plus juste de la BW dispo)
  - Alternance entre SS (MIMD, si cwnd<ssthresh) et CA (AIMD, si cwnd>ssthresh)
  - triple ACK: ssthresh := cwnd/2; cwnd /= 2 et on repart en CA
  - timeout: ssthresh := cwnd/2; cwnd = 1 et on repart en SS

I) Introduction

- IP, c'est LE protocole d'internet.
  - L'objectif est de passer des paquets entre différents réseaux interconnectés
  - pour constituer un super-réseau formé de sous-réseaux indépendants
- Première réponse historique
  - protocole de transport de paquets IMP (1969, Arpanet), protocole NCP par dessus
  - Toutes les fonctionnalités dedans, très intégré, mais modèle rigide. Difficile d'interconnecter des réseaux hétérogènes
- Réponse d'Internet: protocole tout intégré en 1974 et bientôt découpé en TCP, UDP et IP.
  - Cela a valu le prix Turing 2004 à Vint Cerf et Robert Kahn
  - David C Clarke "The design philosophy of DARPA Internet Protocols" (1988) explique à posteriori les choix de design (notes de lecture "morning paper" ajouté sur le site du cours pour les curieux)

  - Objectifs principaux:
    * Usage multiplexé (pas de circuit établis)
    * Interconnexions de réseaux disparates existants
  - Objectifs secondaires:
    * Continuité de service malgré les pannes
    * Différents types de services et de réseaux: Tailles de paquets, adressage, modèle de service
    * Management/contrôle décentralisé et dynamique (arrivées/départs noeuds)
    * Objectif d'efficacité (contrôle de congestion, usage optimal des ressources malgré les variations perf)
  - Modèle de service choisi: datagram sans connexion en best-effort
    * C'est le plus petit dénominateur commun, mais faire plus est parfois contre-productif donc autant laisser les couches hautes faire ce qu'elles veulent

- Principe d'IP (annonce de plan)
  - Entêtes des paquets comme interface standardisée
    * élément fondamental du protocole entre émetteur/récepteur et récepteur/applications
    * l'implémentation n'est pas standardisée (moins rigide)
  - Adressage: chaque noeud a une adresse unique
  - Forwarding: quand un paquet arrive sur un routeur, il est renvoyé sur la bonne interface (la bonne carte réseau) en sortie, en fonction d'une table de routage.
  - Routage: calcul des tables de routage

II) Adressage IP

- En fait, c'est chaque carte réseau qui a une adresse. Donc un routeur avec 4 cartes réseaux a 4 adresses IP.
- IPv4: entier 32 bits, lu par paquet de 8 bits; IPv6: entier 128 bits, lu en hexadécimal
- On voudrait donner des adresses similaires aux ordinateurs proches dans le réseau afin de compacter les tables de routage
  - À la poste, on fait suivre par pays puis par région puis par ville puis ...
  - IP groupe les adresses "proches" par préfixe commun. Il faut différencier entre le préfixe "réseau" et le suffixe "ordinateur de ce réseau là"
- À l'origine, la frontière est directement dans l'adresse, en fonction du préfixe:

  - si ça commence par "0" (en binaire), c'est une adresse classe A, donc 7 bits pour encoder le réseau et 24 bits pour encoder la machine. On peut avoir 127 tels réseaux, de 2^{24} machines chaque 16,777,216.
  - si ça commence par "10", classe B. Donc 14 bits pour le réseau et 16 bits pour la machine. 2^{14} réseaux (16,384) de 2^{16} machines (65,536)
  - si ça commence par "110", classe C. Donc 24 bits pour réseau et 8 bits pour la machine. 2^{21} réseaux (2,097,152) de 2^8 machines (256)
- Mais il y a rapidement eu pénurie de réseaux (surtout les B)
  - Maintenant adresses CIDR (Classless InterDomain Routing), où l'on précise la taille du préfixe à part
  - Deux formes équivalentes: masque binaire (255.255.255.0) ou nombre de bits du réseau en postfix (/24)
  - Le préfix n'est pas limité à une taille multiple de 8. Adresse /17 pour un gros ISP
- Question: quelle est la plage d'adresse représentée par 128.138.207.160/27 ?
  - 128.138.207.160 = 10000000 10001010 11001111 101|00000 en binaire, et 27 bits vont jusqu'à la barre
  - 10000000 10001010 11001111 101|11111 en binaire (la plus grande machine), c'est 128.138.207.191
  - Donc la plage, c'est 128.138.207.160 - 128.138.207.191
- Question: quel est le masque dans le cas précédent?
  - 128.138.207.160/27 = 128.138.207.160/255.255.255.224
  - Autre: 195.176.181.11/30 = 195.176.181.11/255.255.255.252
    $\rightsquigarrow$ c'est une adresse dans la plage [195.176.181.8; 195.176.181.11]
    * $11_{\mathrm{d}} = 0000\ 1011_{\mathrm{b}}$ et $0000\ 1000_{\mathrm{b}} = 8_{\mathrm{d}}$
  - Autre: 192.168.0.3/24 = 192.168.0.3/255.255.255.0
    $\rightsquigarrow$ c'est une adresse dans la plage [192.168.0.0; 192.168.0.255]
- On regarde l'exemple d'adressage de sous-réseau sur le document
  - Réseau 1 est forcément 128.96.34.0 car il contient des petites valeurs et est de masque 128, donc le premier bit est un 0
  - Valeurs possibles pour Masque 2: ce qu'il faut pour encadrer le contenu
    * Contenu: $129_d = 1000\,0001_b$; $130_d = 1000\,0010_b$; $140_d = 1000\,1100_b$
    * 255.255.255.128 ($128_d = 1000\,0000_b$)
    * 255.255.255.192 ($192_d = 1100\,0000_b$)
    * 255.255.255.224 ($224_d = 1110\,0000_b$)
    * 255.255.255.240 ($240_d = 1111\,0000_b$)
  - Pour le réseau 3, on peut prendre simplement 128.96.33.0/24 (255.255.255.0)
- Il existe des adresses privées, spécifiques.

  - 127.0.0.1/32: c'est l'adresse de la machine locale, tout le temps
  - 192.168.0.0/16: réseau privé, utilisé pour à la maison derrière un NAT
- Habituellement, la première adresse de la plage est pour le réseau ..*.0
  - La dernière adresse de la plage est pour le broadcast
  - Donc on peut mettre 254 ordinateurs par réseau/24

III) Forwarding

III.1) Principe

- Principe: on prend le préfixe de longueur maximale parmi les routes existantes
- Le travail de base d'un routeur est donc de trouver le lien de sortie pour chaque adresse destination des paquets entrants
  - Il faut aller très vite (jusqu'à max 20 ns par paquet)
  - Si les préfixes étaient bien répartis, ça irait mais
    pénurie d'adresses $\rightsquigarrow$ fragmentation
    (qques tables de 25k lignes au coeur du réseau)
  - ASIC = processeurs spécialisés (+ des ruses algorithmiques hors sujet ici)
- Exercice 2 sur le document associé. Calculer l'interface de sortie
  - 123.4.1.69 $\rightarrow$ 1 (seulement dans 123.4.0.0/16, ligne 1)
  - 68.142.226.44 $\rightarrow$ 4 (seulement dans 0.0.0.0/1, avant dernière ligne)
  - 98.7.2.71 $\rightarrow$ 2 (dans le réseau 98.7.1.0/16, ligne 2)
  - 200.100.2.1 $\rightarrow$ 3 (dans le réseau énorme 128.0.0.0/1, ligne 4)
  - 128.138.207.167 $\rightarrow$ 4 (à la fois dans le réseau énorme et dans 128.138.0.0/16)
  - 123.4.20.11 $\rightarrow$ 2 (dans le réseau 123.4.20.0/24 et 123.4.0.0/16, donc on prend le long)
  - 123.4.21.10 $\rightarrow$ 1 (seulement dans 123.4.0.0/16)
- Rien ne dit que le routage est symétrique sur Internet, et calculer les tables optimales est difficile. On y revient

III.2) Autres mécanismes d'IP

- Fragmentation, quand le MTU du second réseau est plus petit
  - Il y a un champ dans les entêtes pour numéroter les sous-paquets
  - Réassemblage à destination ou en chemin si les paquets se croisent
  - Forcément, c'est moins efficace quand ça fragmente
- TTL, pour éviter les boucles (dues aux mauvaises configurations)
  - c'est en nombre de sauts, et le diamètre max d'internet est donc 256

- Protocole de contrôle IP: ICMP
  - Signalement des pbs de TTL ou de checksum
  - Permet de faire des ping et des traceroutes, mais aussi des DDos donc de moins en moins utilisé
- Pénurie d'IP publiques: NAT (network address translation)
  - On cache un réseau privé entier derrière une seule IP publique
  - Exemples: salle de TP ou tous les hôtes derrière un set-top-box = modem router
  - Multiplexing des machines masquées sur les ports de la machine publique

III.3) Protocoles associés (OPTIONNEL)

- Dans un réseau donné, il faut trouver l'adresse MAC du destinataire. C'est le protocole ARP
  - broadcast "Qui a l'IP bidule?"; réponse de l'intéressé
  - tout le monde met toutes les infos (y compris MAC demandeur) en cache avec timeout
  - car oui, tout le monde écoute tout sur un réseau éthernet ou wifi
- Pour obtenir une adresse IP dynamique, sans config: DHCP
  - client broadcast "Discover"
  - serveur broadcast "offer"
  - client broadcast "request" (pour gérer le cas où deux serveurs font des offres différentes)
  - serveur broadacst "ACK"

IV) Routage

IV.1) Introduction

- L'objectif est de calculer les tables de routages
  - Doit fonctionner de façon dynamique, décentralisée et à large échelle
- Le problème est très différent intra- ou inter-AS (autonomous system = ISP ou entreprise ou univ)
  - intra-AS: dans une organisation donnée
    - on a toutes les infos, relativement homogène
    - on vise l'optimal technique (usage des ressources et performance)
  - inter-AS: entre institutions ou entreprises concurrentes
    - configuration interne = secret commercial;
    - on vise l'optimal commercial (négociation financière)

IV.2) Routage intra-AS

- Le réseau est un graphe, chacun connait ses voisins (remember ARP)
- Latence comme métrique de distance additive
- Le problème est une recherche de plus court chemin, à faire en temps réel distribué (et dynamique)
- Deux approches possibles pour calculer le plus court chemin:
  - Vue locale du réseau, et calcul distribué (vecteur d'état)
  - Vue globale du réseau, et calcul local (état des liens)

IV.2.a) Algorithme par vecteurs d'états (Bellman-Ford 1956)

- Distance-vector routing en anglais
- Chaque routeur u maintient les informations suivantes pour toute destination v:
  - $D_u[v]$: Longueur du plus court chemin connu vers v
  - $N_u[v]$: Next hop, i.e. voisin à qui u doit envoyer les infos à destination de v
- Deux vecteurs de taille 10 dans l'exemple du doc (10 machines nommées)
- Algorithme
  - Valeur initiale de $D_u[v] = \infty$ si v n'est pas voisin de u, ou poids de l'arrête
  - À chaque itération, chaque routeur envoie son vecteur D à ses voisins
  - À la réception du vecteur de distance D_x envoyé par x + $c(u,x)$ coût de $\vec{ux}$, le routeur u décide de passer par x si c'est mieux que la route connue
    - $\forall v$, si $c(u,x)+D_x[v] < D_u[v]$ alors $N_u[v] = x$ et $D_u[v] = D_u[x]+D_x[v]$
    - Les infos reçus sont conservées. Utile quand les choses empirent
- Déroulé de l'algorithme sur l'exemple de la feuille (moitié gauche seulement)

nD↷	A	c	d	e	f
A	–	?∞	?∞	e2	?∞
c	?∞	–	d4	e3	?∞
d	?∞	c4	–	e5	f7
e	A2	c3	d5	–	f3
f	?∞	?∞	d7	e3	–

Situation initiale

nD↷	A	c	d	e	f
A	–	?∞	?∞	e2	?∞
c	?∞	–	d4	e3	**d11**
d	?∞	c4	–	e5	f7
e	A2	c3	d5	–	f3
f	?∞	**d11**	d7	e3	–

Broadcast de D_d vers c, e et f.

nD↷	A	c	d	e	f
A	–	**e5**	**e7**	e2	**e5**
c	**e5**	–	d4	e3	d11
d	**e7**	c4	–	e5	f7
e	A2	c3	d5	–	f3
f	**e5**	**e6**	d7	e3	–

Broadcast D_e vers A, c, d, f.

- Cet algorithme converge en au plus $O(diametre)$ étapes vers le routage parfait
- Mais les mauvaises nouvelles convergent mal. Si $e\vec{A}$ tombe en panne
  - e annonce $[\infty, ...]$
  - À la réception, c annonce $[11, ...]$ en passant par d
  - À la réception, d annonce $[15, ...]$ en passant par c
  - À la réception, c annonce $[19, ...]$ en passant par d
  - À la réception, d annonce $[23, ...]$ en passant par c
  - À la réception, c annonce $[27, ...]$ en passant par d
  - Ils vont mettre longtemps à revenir à ∞ :(
  - En pratique (protocole RIP), on compte en nombre de sauts, et $16 = \infty$
- Clivage de l'horizon
  - u n'annonce pas à v les distances des destinations x où $N_u[x] = v$
  - Converge plus vite quand un lien tombe, mais pas parfait
- Reverse poisonning: $u \to v$: $D_u[x] = \infty$ si $N_u[x] = v$
  - Converge encore plus vite, mais boucle à 3 éléments encore possible
  - BGP utilise un algorithme proche, mais chacun annonce les chemins complets qu'il prend pour les destinations, ce qui permet à l'interlocuteur de gérer les cycles
- Une variante de cet algo est utilisé en pratique par le protocole RIP

IV.2.b) Algorithme par état des liens (Dijkstra 1960)

- L'objectif est de donner à tous une vision complete de l'état du réseau, pour que chacun applique Dijkstra pour lui
- Chacun broadcast l'état de ses liens locaux de temps en temps;
  Comment faire pour que tout le monde reçoive tout?
- Idée 1: flooding.
  - Quand je reçois un tel message, je le fais suivre à tous mes voisins (sauf émetteur)
  - Sympa, mais explosion nb messages s'il y a des boucles

- Idée 2: Reverse Path Broadcast
  - Quand u reçoit de v les infos que x a broadcasté, u fait suivre ssi $N_u[x] = v$
  - (broadcast uniquement les infos qui parviennent sur le plus court chemin)
  - c fait suivre les infos de A seulement quand elles viennent de e (pas si elles arrivent de d)
  - Sympa, sauf que je ne connais justement pas le plus court chemin avant que tout le monde ait tout broadcasté
- Idée 3: Flood contrôlé par numéro de séquence
  - À l'émission, chaque paquet broadcasté a un numéro de séquence, qui augmente pour le broadcast du prochain
  - En local, chacun stocke n_v, le dernier numéro de séquence reçu de v
  - À la réception d'un paquet de v numéroté N, je le broadcast ssi $N > n_v$, puis mise à jour $n_v \leftarrow N$
  - (je fais suivre les paquets seulement la première fois que je les vois)
- Ensuite, l'algo de Dijkstra est similaire à Bellman-Ford mais en local et sans pb
  - D_u : distance pour aller de là où je suis jusqu'à u
    $p[v]$: précédent de v, là par où j'arrive à v en venant de là où je suis
  - Initialisation de toutes les distances à ∞, sauf les voisins du noeud local
    - ensemble todo contenant tous les noeuds sauf le noeud local
  - À chaque tour de boucle, sélectionne le noeud $u \in todo$ tq D_u est minimale
    - pour tous les voisins de u dans $todo$, notés v
      - si $D_u + c(u, v) < D_v$ alors $D_v = D_u + c(u, v)$ et $p_v = u$
  - Quand j'ai fini l'exécution (quand $todo = \emptyset$), j'utilise récursivement les p_x pour savoir par où sortir pour aller à un noeud donné
- Exemple d'exécution de Dijkstra pour le noeud d

A		B		c		**d**		e		f		g		h		i		j		
D	p	D	p	D	p	**D**	**p**	D	p	D	p	D	p	D	p	D	p	D	p	
∞	?	∞	?	4	d	–	–	5	d	7	d	3	d	∞	?	∞	?	∞	?	init
						–	–							15	g	9	g	10	g	g
						–	–													c
7	e					–	–													e
						–	–													A
						–	–							14	f					f
		12	i			–	–													i
						–	–							11	g					j
						–	–													h, B

- Une variante de cet algo est utilisé en pratique par le protocole OSPF

IV.3) Routage inter-AS

- C'est très différent d'intra-AS car c'est un problème économique et non un pb technique
  - Les ISP ne veulent pas communiquer l'état exact de leur réseau interne (trust issue)
  - On peut vouloir forwarder vers un réseau moins performant mais moins cher
- Link state/Dijstra nécessite info complete, pb extensibilité et impossible de moduler sa politique
- Vecteur distance/Bellman ne permet pas de moduler sa politique, converge lentement
- Algo de vecteur de chemins à la place.
  - Comme un vecteur de distance, mais on propage le chemin complet qu'on prend pour chaque destination
  - Plus de mémoire et taille message, mais pas plus de calculs
- Protocole BGP (Border Gateway Protocol) utilisé en pratique
  - Chaque AS annonce les autres AS qu'il est capable d'atteindre, avec info de distance (et la route des AS utilisées pour y aller, donc)
  - Pas de boucles possibles puisque routes complètes annoncées (sauf si churn sévère)
  - Chaque AS peut mentir par omission et ne pas annoncer certaines routes pour lesquelles il sait router mais ne souhaite pas prendre en charge les données des autres
  - Ca passe bien à l'échelle, mais atteignabilité pas garantie : si personne veut faire passer le trafic des autres, ça marche forcément pas

- Exemple sur le réseau à router du doc.
  - 1: e annonce savoir aller à A directement. e:(A,1,$\emptyset$)
  - 2: d et j ne font pas suivre l'info de e (trop cher?)
  - 2: f fait suivre l'info (A,2,{e})
  - 3: d:(A,3,{e,f}) et h:(A,3,{e,f})
  - 4: g:(A,4,{e,f,d})
  - 5: i peut choisir entre (A,4,{e,f,h}) et (A,5,{e,f,d,g})
- Si e annonce ensuite ne pas pouvoir rejoindre A, d et f ne vont pas partir en boucle puisque chacun sait que l'autre comptait sur le lien qui vient de disparaître

C'est fini

- C'est la fin du module. Lundi prochain, partiel.
- D'autres modules vous demandent de savoir lire du C (arcsys, C++), mais si vous ne cherchez pas à poursuivre, vous ne referez jamais plus de réseau. Avant l'agreg :)