Exposé 1/ [redacted] : IPv4 - IPv6

6rd : pour faire de l'ipv6 rapidos sur ton réseau IPV4 d'opérateur, parce qu'en fait c'est gros, et que t'as plein de trucs en IPv4, donc tu peux pas juste faire des équipements d'interco v6 : tu prends des passerelles 6to4 (Communication en v6 et sur v4 en inversement, construit en 2002::. Dessus y'a un problème, parce que tout le monde peut récupérer tout le flux de tout le monde, et si il est pas retransmis, aux chiottes l'interco), et à la place de prendre un préfixe en 2002, toi, opérateur, t'as obtenu un vrai préfixe et tu fais +. ça juste marche !

Sécurisation des adresses : basiquement on fait des adresses IPv6 autoconfigurées (SLAAC) en utilisant l'adresse MAC, en gros, dans l'adresse IPv6. Sauf que c'est un leak, ça donne des infos sur la machine et un moyen d'identifier/profiler les gens. Pour faire mieux, on peut alors utiliser d'autres méthodes comme la génération d'identifiants aléatoires stables ou non en remplacement de l'adresse MAC. Dans tous les cas, il faut une fonction DAD (Duplicate Address Detection) pour être sûr qu'une adresse générée n'existe pas déjà.

Exposé 2/ [redacted] : Amélioration des perfs de datacenters

On cherche à gagner en efficacité tout en ayant un taux de perte ridiculement faible. Les topologies habituelles (hiérarchiques à 3 niveaux) sont pas assez performantes et coûtent cher donc on en met en place d'autres : fat tree (bonne scalabilité, coût faible), flattened butterfly (moins ede liens que fat tree) et dragonfly (perfs dépendant beaucoup des capacités des commutateurs, donc potentiellement plus cher). En plus de ces typologies régulières qui minimisent les coûts, on teste divers protocoles de routage ainsi que des mécanismes de load balancing. Habituellement on utilise du Shortest Path, pas adapté ici à cause des liens de même coût. Morale de l'histoire : le partage de charge par paquets est plus fiable que le partage par flux qui peut varier en efficacité avec la taille des flux. Pareil pour le routage adaptatif, source d'instabilité dans le réseau, donc pas adapté ici.

À propos des protocoles de routage : on teste SPRm, ECMP et VBR.

- SPRm = SPR avec en plus un critère sur le nœud de degré minimal.
- ECMP génère de table de routage de tous les chemins de même coût vers un ensemble de destinations. Il ne reste qu’à load balancer les paquets entrants vers des chemins différents pour alléger la charge et gagner en débit.
- VBR : sélection random parmi tous les noeuds du réseau de data center d’un noeud intermédiaire vers lequel envoyer les paquets, et utiliser le chemin le plus court vers ce noeud. Une fois arrivé au noeud,

le paquet emprunte le chemin le plus court vers la destination finale.

Après avoir testé chacun de ces protocoles, on constate que les variantes standards proposent toutes un débit inférieur à leurs variantes load balancées (LB-SPRm multiplie par 25 le débit proposé par SPRm sur une topologie fat tree).

Exposé 3/ RIFT: Routing in Fat Trees

La propriété singulière de RIFT est d'inonder uniquement les informations d'état de liaison "plates" vers le nord de sorte que chaque niveau comprend la topologie complète des niveaux au sud de celui-ci. Ces informations ne sont plus jamais inondées d'est en ouest ni de retour vers le sud. Dans la direction sud, le protocole fonctionne comme un protocole de vecteur de chemin "unidirectionnel" ou plutôt un vecteur de distance avec un horizon fractionné implicite alors que l'information ne se propage que d'un bond vers le sud et est "republiée" par les nœuds au niveau inférieur suivant. Cependant, nous utilisons également les inondations dans la direction sud pour éviter la nécessité de créer une mise à jour par voisin. Nous laissons de côté la direction Est-Ouest pour le moment.

Ces contraintes de flux d'informations créent une propagation "fluide" des informations où les nœuds ne reçoivent pas les mêmes informations de plusieurs fronts, ce qui les obligerait à effectuer un calcul diffus pour briser les mêmes informations d'accessibilité arrivant sur des liens arbitraires et, en fin de compte, forcer hop-by- transfert de saut sur les plus courts chemins uniquement.

Pour tenir compte de la division des informations "nord" et "sud", la base de données d'état des liens est divisée en TIE "représentation nord" et "représentation sud", alors qu'en termes plus simples les N-TIE contiennent une description de la topologie de l'état des liens des niveaux inférieurs et et les S-TIE transportent simplement des routes par défaut. Cette vision simplifiée sera affinée progressivement dans les sections suivantes tout en introduisant des procédures de protocole visant à répondre aux exigences décrites

Exposé 4/ Virtual Chassis Fabric Ethernet et VSS

(Juniper) Virtual Chassis : Interconnexion de switchs en anneau, manageable comme un seul switch

Virtual Chassis Fabric : Virtual Chassis organisé en architecture Spine&Leaf

Réseau Clos, intérêts :

- Simplicité pour l'administrateur
- Redondance et tolérance aux pannes
- Nombre de sauts fixe entre les équipements donc méthode de routage déterministes
- Facilité d'évolution

VSS (Cisco) : fusion logique de deux switchs, fonctionnement en mode Maître/Esclave. Les deux switchs sont actifs (même l'esclave). Ex : le primaire porte la configuration et les fonctions de virtualisation et le secondaire continue de switcher et de répondre aux requêtes ARP. Merci pour votre attention.

Exposé 5/ Architecture Juniper des DataCenter de Google

1) L'architecture Clos qu'est-ce que c'est ?

La topologie

cf cours

Les avantages

- Pas de noeud d'étranglement
- Redondance : résiste aux pannes
- Grand débit avec équipements standards et pas chère
- Evolutif
- Load Balancing + QoS (vPC)
- Latence égale

2) Les évolutions physiques du Data Center

- rien de très utile pour l'exam

3) Le Protocole de Routage Firepath

Pourquoi pas OSPF ?

- Contexte Datacenters :
    - Que du multipath à coûts égaux
    - Pas de dynamicité de l'infrastructure
- OSFP :
    - Pas de multipath si coûts égaux
    - Décentralisé donc Overhead sur le réseau

- Architecture dynamique

=> OSPF non adapté aux DC

Les points clés de Firepath Protocol

- Système de routage Centralisé :
  - Control Plane Network (CPN)
  - Basé sur l'état des liens
  - Une Base de Données des états centralisée
- Comment ca marche ? (les étapes)
  1. Tous les switch Clients sont lancés avec une config de base (cf plus tard)
  2. Chaque switch clients récupèrent les infos de ses ports grâce à IFM
  3. Communiquent les états au Master => Constitue une DataBase "link state database (LSD)"
  4. Update : Le Master envoie la DataBase aux clients (Via UDP multicast sur CPN)
  5. Les clients actualisent leur routes Shortest Path basée sur cette update via Equal-Cost Multi-Path (ECMP)
  6. Puis périodiquement le master envoie des messages KeepAlive avec le numéro de version de sa DataBase pour la syncro

Exposé 6/ [redacted] : Haute disponibilité dans les DC Google

Les réseaux évoluent car on apprend au fur et a mesure de nos erreurs. Google classifie les types d'erreur qui surviennent dans leur data center pour mieux les anticiper et les résoudre au plus vite. Une erreur peut en entrainer d'autres d'ou l'intérêt d'identifier la root cause.

Il y a différents concept pour assurer la haute disponibilité d'un DC:

- Le defense in depth

En gros c'est la redondance des équipements et la redondance inter réseaux. Si un réseau B4 tombe, on passe par un réseau B2. Pour rappel B4 c'est du SDN et B2 c'est des archis traditionelles non virtualisées.

- le maintien de la consistence inter plan

S'assurer que le plan de données soit toujours à jour selon le plan de contrôle, utiliser des invariants dans le réseaux, faire du monitoring...

- le fail open

Quand quelque chose dans le réseau cesse de fonctionner, s'assurer que ça ait un impact minimal sur le reste du réseau = cloisonnement de sous réseau, laisser le plan de données en fonctionnement lorsque le plan de contrôle est en echec...

- Le recover fast

qui signifie juste "faire en sorte de corriger tout au plus vite" ça passe par des analyses de root cause et du monitoring Difficile de détailler plus, c'était surtout du vent. J'apprécie ton honnêteté !

Exposé 7/ [redacted] : QCN Algorithme de notification de congestion

1) Quantized Congestion Notification

Critères nécessaires au QCN :

- Stabilité (besoin que le buffer ne varie pas trop).
- Réactivité (l'algorithme doit être réactif pour éviter les congestions)
- Equité (les flux doivent avoir une part égale du débit)
- Facilité d'implémentation (nécessité d'implémenter ça au niveau hardware, donc pas besoin de faire ça au niveau logiciel)

Sert à notifier la congestion au niveau d'un routeur qui renvoie alors un feedback.

Si son buffer est trop rempli il renvoie un feedback à la source (qu'on appelle le Rate Limiter) qui diminue alors son débit d'envoi.

Définitions :

- Current Rate (débit actuel)
- Target Rate (débit objectif à atteindre)
- Bit Counter (une fois qu'on a reçu une certaine quantité de données le compteur indique qu'on a terminé un cycle)
- Time Counter (une fois qu'on a attendu un certain temps le compteur indique qu'on a terminé un cycle)

Une fois le feedback reçu on diminue le débit, puis on passe en fast recovery au bout d'un moment (même principe que tcp). On double le target rate au fur et à mesure (il y a en tout 5 cycles, par Bit Counter ou Time Counter). Si un des 2 compteurs a terminé son cycle on passe en Active Increase (on augmente encore plus le débit). Si les 2 compteurs ont terminé leurs cycles on passe en Hyper Active Increase

(toujours + de débit).

Les compteurs servent car on a un mécanisme de notification de congestion, mais pas de mécanisme pour notifier lorsque le système est prêt à ré-augmenter le débit auquel il reçoit.

2) Averaging principle

Cette partie sert à montrer que l'algorithme est viable même si on ne sait pas définir le RTT (qui introduit de la latence dans le système), et donc fait que le système de congestion aurait peut-être du retard (on notifie une congestion qui n'existe plus).

Avec le averaging principle on peut estimer le RTT potentiel dans le système et aussi estimer le fonctionnement de l'algorithme et prouver qu'il fonctionne bien.

La méthode de simulation reprend le Current Rate et le Target Rate (on réévalue ces deux rate au fur et à mesure pour la simulation).

3) Conclusion

L'algorithme QCN serait à implanter au niveau hardware et pourrait être utilisable mais n'en est encore qu'au niveau théorique

Exposé 8/ [redacted] : Ethernet pour centres de données

1) DCB

Dans les centres de données (data center en anglais) on veut des flux qui sont bien tolérants aux gros débits, à de très multiples émetteurs, qui peuvent être source de plusieurs liens, et aux rafales (burst en anglais), ainsi que d’avoir des protocoles et des algorithmes simples car on doit utiliser beaucoup de matériel, on le préfère donc bon marché. Aussi, les flux entre 10 kb et 100 Mb sont les plus cruciaux, et sont directement concernés par ce qu’on va présenter par la suite, car nombreux.

On utilise un protocole sur Ethernet qui envoie des trames de pause à l’émetteur en cas de congestion, ainsi pas de suppression de paquet, donc pas de « slow start » (selon l’expression consacrée) ni de contrôle de congestion. Pour des données critiques, utilisé entre les centres de traitement et les centres de stockage, on aimerait l’étendre à plusieurs architectures.

2) Les problèmes de DCB

Cette absence de contrôle de congestion causé par l’absence de suppression de paquet impliquent quelques problèmes.

- Remplissage de file : causé par l’absence de contrôle de congestion
- Problème en tête de file (en anglais Head of Line blocking ou HoL) : envoi de trames de pause vers des émetteurs qui émettent plusieurs flux, qui vont donc arrêter toute transmission même si des liens qui ne sont pas impliqués dans une congestion.
- Iniquité (Unfairness en anglais): Tourniquet (round robin en anglais) sur les ports et non sur les flux, si plusieurs flux arrivent sur le même port, ils seront défavorisés par rapport à un flux qui occupe seul son port.
- Interblocage (deadlocks en anglais) : boucle de flux qui s’attendent, et qui sont par ce fait condamnés à ne jamais arriver à leurs fins.

3) TCP Bolt

On va pour résoudre ces problèmes, utiliser TCP Bolt, qui consiste en utiliser DCTCP et une fenêtre de départ qui est le produit délai débit = RTT * Débit. DCTCP utilise le bit ECN, qui sert dans les versions classiques de TCP à signaler une congestion au destinataire, qui ensuite le signale à l’émetteur. Toutefois, on ne connaît pas l’importance de cette congestion… Le mécanisme de DCTCP permet au destinataire d’envoyer à l’émetteur un changement d’état (passage de congestion à non congestion, ou l’inverse). Ainsi l’émetteur sait précisément combien de paquets sont impliqués dans une congestion, et peut ajuster sa fenêtre d’envoi.

La raison de la fenêtre de débit est de terminer plus rapidement l’intervalle de flux qui sont impliqués dans les data centers, en particulier ceux de 10kb à 1Mb.

On obtient des temps de terminaison de flux réduits de 50 à 70 % en moyenne, jusqu’à 90 %, car TCP Bolt résout tout les problèmes si ce n’est l’interblocage, et donc augmente significativement les perf. Les deux premiers par DCTCP et son contrôle de congestion, je suppose (donc non contractuel).

4) EDST

Le routage sans interblocages (DFR) existe déjà mais le passage à l'échelle est compliqué (milliers de serveurs). On veut montrer qu'avec EDST, on peut faire du DFR en conservant de bonnes performances. (soit proche de ECMP, qui a un risque d'interblocage mais est très utilise)

On exploite deux faits :

1. Le routage par plus court chemin est sans interblocage sur les arbres.
2. Les arbres à arêtes disjointes sont garantis d’être isolés les uns des autres

On va transformer la topologie du réseau en foret de spanning tree à arrêtes disjointes (EDST) qu'on utilise pour le routage. On obtient ainsi un certain nombre d'arbre pour un réseau donné et plus ce nombre est grand mieux c'est. Ethernet permet par les VLANS d'utiliser les mêmes liens pour plusieurs routes et de les

distinguer, de les mettre ne pause séparément etc

On peut ainsi multiplier le nombre d'arbres et en profiter pour trouver de nouvelles routes pour plus de diversité.

En terme de performances, EDST atteint 85% des performances d'ECMP dans le pire des cas

Exposé 9/ ICTCP

Objectif : diminuer les pb de congestion de TCP- pour cela on anticipe l'encombrement en utilisant des fenêtres de réception dynamiques.

Pour ce faire un implementé un Algorithme ICTCP. L'auteur a décidé de le faire sur une Network Interface Card(NIC) pour que cela soit naturellement adapté aux machines virtuelles, utilisable sur tous les os. De plus ainsi c'est la NIC qui va calculer le checksum = on gagne du CPU.

Fonctionnement Algorithme : 1 analyseur d'en-tête, 1 table de flux, et un Algorithme dont le rôle est d'adapter les tailles de fenêtre.

Problème : machine virtuelle = on connaît pas la vraie bande passante (elle est surévalué dans les data center car tous les machines ne sont pas utilisés en simultané) > on déploie ICTCP aussi sur serveur hôte des machines hôtes.

Problème 2 : granularité de RTT-> on modifie le TCP timestamp pour la diminuer.

Expérience : bah ça marche mieux que TCP quoi mais que d'un switch vers d'autres ports du même switch. 4 expériences ou c'est mieux que TCP à chaque fois : Équitable, Avec d'autres flux d'arrière plan, en faisant varier le volume de donner ou le nombre de serveur

Exposé 10/ [redacted] : automatisation de la configuration réseau

La configuration réseau à beaucoup évolué avec le temps. De nombreux protocoles se sont succédés et tendent vers une approche plus intelligente et orientée logicielle. La CLI est peu robuste car écrite manuellement par un administrateur et ne constitue qu'une configuration locale. SNMP est une évolution et propose une architecture orienté API qui configure le réseau à distance. Un NMS (newtork managamenet system) va configurer les périphériques réseaux via SNMP. Les informations de configuration réseaux sont stockées dans les bases de données des équipements (MIB : management information base). SNMP est cependant peu adapté à la configuration réseau mais bien plus au monitoring et à la gestion des élément réseaux.

YANG est un modèle de langage de description réseau. Netconf encode en XML un modèle YANG et utilise des messages RPC pour configurer des équipements réseaux. Des requêtes client/serveur RPC

permettent de modifier, récuperer, effacer … des configurations réseaux. RestConf s'appuie sur une architecture Rest et reprend le modèle NetConf. Seuls changements, XML devient JSON et RPC devient HTTP. (on utilise des requêtes HTTP pour récupérer les configurations). RestConf s'inscrit donc dans l'avénement des applis webs.

SDN/NFV consitue une nouvelle fasson (dédicace) de penser une architecture réseau. Les réseaux en plus d'être programmables, ne sont pas épargnés par le tout logiciel. Les outils devops (Git, Ansible, Pytest) ainsi que les solutions constructeurs (Cisco NSO ou VMWARE NSX) vont permettre d'automatiser les configurations réseaux, en s'appuyant sur des protocoles en vogue comme NetConf ou RestConf (avec le modèle YANG).

Exposé 11/ [redacted] : Trafic Engineering dans B4

1) Challenges

Il y a quatre principaux challenges que l'on veut atteindre :

- la livraison de bande passante massive sur plusieurs milliers de liens individuels qui peuvent avoir de débits différents.
- La tolérance aux pannes. Dans le réseau WAN traditionnel, les liens sont sur-dimensionnées de 2 à 3 fois la bande passante réellement utilisée pour gagner en terme de fiabilité. Mais en faisant ce surdimensionnement, ça veut dire que on a besoin plus de débits qu'on utilise actuellement et c'est un problème parce que le WAN entre les datacenters nécessite déjà une bande passante massive.
- La demande de flux élastique, c'est-à-dire un flux qui peut adapter le délai et le débit sur internet. On voudrait que chaque application bénéfice de la bande passante autant qu'elle peut obtenir, mais peut toujours tolérer des échecs périodiques (par, par exemple, réduire temporairement son débit). Et on veut maximiser la bande passante moyenne en respectant le plus possible le partage équitable entre différents applications.
- On veut avoir le control total sur les serveurs extrémités et le réseau, c'est-à-dire on veut contrôler à la fois les applications et les réseaux connectés au réseau B4. Pourquoi on veut pouvoir contrôler tous ? parce que on veut pouvoir appliquer des priorités relatives aux applications et contrôler les bursts à l'extrémité du réseau, plutôt que de faire le surprovisionnement ou d'effectuer les fonctionnalités très complexes dans B4.

Du coup, pour pouvoir satisfaire tous les challenges, Google a pensé à l'architecture SDN avec OpenFlow qui permet de séparer le plan donnée et le plan control et à faire le trafic engineering centralisé qui a pour l'objectif d'atteindre l'utilisation des liens à 100% en essayant de balancer la capacité des liens selon les priorités / les demandes.

2) Architecture B4

- Séparer le protocol et matériel

- Ajouter un SDN Controleur au milieu (entre protocol et silicon) pour envoyer les paquets aller et retour, calculer les flux et les met dans les switchs.
- Ajouter SDN Controleur secondaire pour la tolérance aux pannes

Allocation de la bande passante

une bandwidth function est une fonction qui alloue une bande passante à une application donnée selon ce qu'on appelle le "Fair-Share" (une priorité d'une application en prenant en compte le partage équitable avec les autres applications).

2 types de fonctions :

- Bandwidth function per application (chaque application est associée une bandwidth function)
- Bandwidth function per flow group (l'addition linéaire de chaque bandwidth function par application dans ce groupe de flux)

Résultat obtenus

on arrive à avoir une utilisation globale proche de 100% sur une période de 24 heures. tous les liens sont utilisés à presque 100% tout le temps.

De manière générale, le load-balancing dans le réseau B4 est très bien fait. Presque pas de perte de paquet de haute priorité.

- Un avantage clé du Trafic Engineering centralisé est la possibilité de mélanger les classes de priorité sur tous les noeuds extrémités.
- Il assure que les liens fortement utilisés transportent les flux de faible priorité.
- Les schedulers de QoS peuvent garantir que les flux de haute priorité est protégé contre les pertes.

3) Conclusion

La décision de construire B4 a été motivée par l'observation que l'on en pouvait pas atteindre le niveau d'échelle(c-à-d la bande passante massive), la tolérance aux pannes, la rentabilité (c-à-d maximiser l'utilisation des liens) et le contrôle requis pour le réseau Datacenter Google en utilisant des architectures WAN traditionnelles.

Google a deux réseaux WAN :

- Le user-facing WAN : c'est le réseau qui relie les utilisateurs et qui échange les trafics avec les autres domains d'internet
- Le B4 : qui relie les douzaines datacenters de Google et qui est en archi SDN

Le B4 était en production pendant trois ans, et a servi plus de trafic que le user-facing WAN de Google, et a un taux de croissance plus élevé.

L’architecture SDN, OpenFlow, et le Traffic Engineering centralisé permet l’utilisation à 100% de la capacité des liens

Le fait de pouvoir contrôler les noeuds extrémités pour mesurer la demande et pour jouer sur la priorité global (c-à-d le fair share) permet non seulement l'augmentation de l'utilisation du WAN, mais aussi l'amélioration de la tolérance aux pannes.

Le trafic engineering centralisé permet :

- d’augmenter le contrôle à l’extrémité du réseau pour ajuster les demandes pendant une panne.
- d’utiliser le multiple-path forwarding pour augmenter la capacité disponible du réseau en fonction de la priorité de l'application.
- de réallouer dynamiquement la bande passante ou déplacer la demande des applications pendant la période de panne.

Exposé 12/ [redacted] : Ethernet ring protection switching

ERPS spécifie des mécanismes de commutation de protection et un protocole pour les anneaux de réseau Ethernet (ETH). Les anneaux Ethernet peuvent fournir une connectivité multipoint sur une large zone plus économique grâce à leur nombre réduit de liaisons. Les mécanismes et le protocole définis dans la présente recommandation assurent une protection extrêmement fiable et stable et ne forment jamais de boucles, ce qui aurait une incidence fatale sur le fonctionnement du réseau et la disponibilité du service.

Chaque nœud en anneau Ethernet est connecté aux nœuds en anneau Ethernet adjacents participant au même anneau Ethernet, en utilisant deux liaisons indépendantes. Une liaison en anneau est délimitée par deux nœuds annulaires Ethernet adjacents, et un port pour une liaison en anneau est appelé un port en anneau. Le nombre minimum de nœuds d'anneau Ethernet dans un anneau Ethernet est de trois.

Les principes fondamentaux de cette architecture de commutation de protection d'anneau sont :

- Le principe de l'évitement des boucles.
- L'utilisation des mécanismes d'apprentissage, de transfert et de filtrage de base de données (FDB) définis dans la fonction de transfert de flux Ethernet (ETH_FF).

L'évitement de boucle dans un anneau Ethernet est obtenu en garantissant que, à tout moment, le trafic peut circuler sur toutes les liaisons en anneau sauf une. Cette liaison est appelée Ring Protection Link (RPL) et, dans des conditions normales, cette liaison en anneau est bloquée, c'est-à-dire qu'elle n'est pas utilisée pour le trafic de service. Un nœud en anneau Ethernet désigné, le nœud propriétaire RPL, est responsable du blocage du trafic à une extrémité du RPL.

En cas de défaillance de l'anneau Ethernet, le nœud propriétaire RPL est responsable du déblocage de son extrémité du RPL (à moins que le RPL n'ait échoué), permettant l'utilisation du RPL pour le trafic. L'autre

nœud en anneau Ethernet adjacent au RPL, le nœud voisin RPL, peut également participer au blocage ou au déblocage de son extrémité du RPL.

En cas de défaillance de l'anneau Ethernet, la commutation de protection du trafic est déclenchée. Ceci est réalisé sous le contrôle des fonctions ETH_FF sur tous les Ring Nodes Ethernet. Un protocole APS est utilisé pour coordonner les actions de protection sur l'anneau.

Exposé 13/ PFC

Priority Flow Control a été mis en place pour rendre la couche 2 plus fiable.

Entre deux noeuds qui s'échangent des paquets, lorsque l'un des 2 prédit une saturation du buffer d'une CoS en particulier (8 au total) il envoie une trame PAUSE.

Il envoie cette trame PAUSE lorsque son buffer atteint un seuil. Pour définir ce seuil, il y a 4 paramètres à prendre en compte : la MTU (Maximum Transmission Unit) du receveur, la vitesse du lien, le temps de réponse de l'émetteur et la MTU de l'émetteur.

Si on éloigne trop les noeuds (qqs km), le PFC ne garantit plus un échange sans perte.