RFC 7149 : Software-Defined Networking: A Perspective From Within A Service Provider

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 6 mars 2014

Date de publication du RFC : Mars 2014

https://www.bortzmeyer.org/7149.html

Ah, le SDN ("*Software-Defined Networking*")... Un "*buzz word*" fort du moment, il est, chez les techniciens, l'équivalent de "*cloud*" chez les commerciaux... Le terme est tellement vague et mal défini qu'il peut désigner tout et n'importe quoi. Dans ce RFC, les deux auteurs explorent le concept, tentent une définition, et nous exposent leur analyse (plutôt critique) du SDN.

SDN désigne en général un ensemble de techniques (dont la plus connue est OpenFlow) permettant de gérer centralement un réseau, lorsque ce réseau est sous une autorité unique (contrairement à ce qu'on lit parfois, le SDN n'a donc pas vocation à être déployé sur l'Internet). L'idée est de ne plus confier aux seuls protocoles de routage la capacité de configurer les éléments du réseau, mais de définir une politique, de la configurer dans une machine centrale, et de l'appliquer ensuite aux éléments du réseau. On voit que, dans la définition la plus générale du SDN, si on remplace les employés qui se loguent sur les routeurs et les commutateurs pour les configurer par un logiciel du type de Ansible ou Puppet, on fait du SDN sans le savoir.

Ah, mais justement, quelle est la bonne définition de « SDN » ? Notre RFC 7149 [1] commence par noter qu'il n'existe pas de définition rigoureuse unique (comme avec tous les "*buzz words*"). L'approche des auteurs est de partir des services réseaux (d'où le titre du RFC) puis de tenter une introduction à SDN. La section 1 du RFC se penche sur les problèmes du fournisseur de services, à partir de l'exemple du VPN. Fournir un service de VPN à ses clients nécessite l'activation de plein de capacités différentes : adressage et routage, bien sûr, mais aussi création des tunnels, qualité de service, filtrage pour la sécurité, et bien sûr tout ce qui est nécessaire à la supervision. La méthode traditionnelle, qui consiste à configurer manuellement (via la ligne de commandes ou via une interface graphique) un grand nombre de machines différentes (car ce n'est pas une seule machine qui va avoir toutes les capacités citées plus haut) est lente,

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7149.txt

coûteuse et l'erreur arrive vite. Il est évident qu'il faut automatiser, si on ne veut pas laisser le client attendre son VPN pendant des jours... pour qu'il découvre finalement qu'une des fonctions attendues ne marche pas suite à une légère erreur humaine. Donc, si vous voulez survivre dans le "*business*" du VPN, vous allez utiliser un programme pour configurer tout cela.

Bon, mais, d'abord, c'est quoi le SDN ? La section 2 du RFC se penche sur le problème de la définition. C'est évidemment difficile car le sigle a été usé et abusé par les marketeux. Par exemple, on définit parfois le SDN comme la séparation, dans les routeurs entre le système de contrôle (en général fait en logiciel sur un processeur généraliste) et le système de transmission (souvent fait par des circuits électroniques spécialisés donc considéré à tort comme n'étant pas du logiciel). Cette séparation offre plus de souplesse d'un côté et plus de performances de l'autre. Mais notre RFC pointe tout de suite que cette séparation est faite depuis de nombreuses années : si c'est là toute la nouveauté du SDN, cela ne vaut pas la peine. En fait, quand on dit « SDN », on ajoute souvent un point important à cette séparation : que le système de contrôle, au lieu d'être dans chaque routeur, vaguement coordonnés par un protocole de routage comme OSPF, soit centralisé, et standardisé pour que le contrôleur et le contrôlé puissent être achetés à des fournisseurs différents. Dans cette définition du SDN, les routeurs et autres équipements réseaux sont « bêtes », juste capables de transmettre les paquets, et un contrôleur « intelligent » fait tous les calculs avant de transmettre des ordres simples aux routeurs. (On parle de PDP - "*Policy Decision Point*" - pour le contrôleur et de PEP - "*Policy Enforcement Point*" - pour les équipements réseau.)

L'argument en général donné par les pro-SDN est celui de la flexibilité. Lorsque de nouvelles exigences apparaissent, par exemple pour offrir un nouveau service aux clients, on change juste le logiciel du contrôleur et tout le reste du réseau suit gentiment. À noter que cela va au delà du routage : il faut aussi pouvoir transmettre aux clients des ordres portant sur la réservation de capacité réseau, par exemple. (Les solutions commerciales qui se disent « SDN » se vantent toutes de leur flexibilité, sans toujours préciser qu'est-ce qui est flexible et qu'est-ce qui est fixé.)

À noter que ces calculs dans le contrôleur supposent que celui-ci dispose d'un bon modèle du réseau. Puisque les équipements réseau vont lui obéir aveuglément, il ne faut pas qu'il transmette des ordres qui mènent à, par exemple, une congestion du réseau. Pour cela, il doit donc connaître les détails du réseau, de façon à être capable de prédire les conséquences de ces actions (ou de façon à pouvoir dire à l'administrateur réseaux « désolé, Dave, ce que tu me demandes est impossible »). Cela peut nécessiter le développement de meilleurs émulateurs de réseau, pour pouvoir tester les conséquences d'un ordre avant de l'envoyer.

Armés de ces précautions, les auteurs du RFC définissent donc le SDN comme « l'ensemble des techniques utilisées pour :

— faciliter l[Caractère Unicode non montré [2]]ingénierie,
— améliorer le temps de production,
— simplifier l[Caractère Unicode non montré]exploitation de services réseau d[Caractère Unicode non montré]une manière déterministe, dynamique et capable d[Caractère Unicode non montré]être déployée à grande échelle.

». On note qu'ils n'insistent pas sur le côté « logiciel » (le S de SDN), les mécanismes réseau étant par essence à base de logiciels.

Ce beau programme nécessite des techniques pour :

— Découvrir la topologie du réseau (on ne peut pas commander un réseau qu'on ne connait pas), les capacités des équipements actifs, et comment les configurer,
— Donner des ordres aux équipements réseau,

2. Car trop difficile à faire afficher par LaTeX

— Avoir un retour sur l'exécution des ordres donnés, et sur leurs résultats.

OK, c'est très joli mais est-ce réaliste ? La section 3 se le demande. Un écosystème comme l'Internet est l'un des objets les plus complexes jamais conçus par l'humanité (et il existe des tas de réseaux qui, sans être aussi gros que l'Internet, sont d'une taille et d'une complexité redoutables). Les exigences des utilisateurs sont innombrables et souvent difficiles à concilier (résilience, y compris face aux attaques délibérées, performances, flexibilité, rapidité de déploiement des changements...) Le RFC suggère donc une certaine modestie. D'abord, se souvenir du passé. On l'a vu, les promesses du SDN étaient souvent plus ancrées dans le verbiage commercial que dans la réalité. Quand elles faisaient référence à quelque chose de concret, c'était parfois quelque chose qui existait depuis longtemps mais qui avait été rebaptisé « SDN » (qui se souvient des vieux slogans commerciaux comme « "*Active Networks*" » ou « "*Programmable Networks*" » ?) C'est ainsi que les NMS ou le PCE (RFC 4655) ont parfois été enrôlés contre leur gré sous la bannière « SDN ». Pour la séparation du système de contrôle et du système de transmission, on peut aussi citer le routage via le DNS (RFC 1383) ou évidemment ForCES (RFC 5810).

Autre suggestion de modestie, être pragmatique. Le SDN doit être utilisé s'il aide, pas juste pour le plaisir de faire des exposés à NANOG. Par exemple, la centralisation du contrôle ne doit pas conduire à diminuer la résilience du réseau en en faisant un SPOF. Si le contrôleur est en panne ou injoignable, il faut que les équipements réseau puissent au moins continuer à assurer leurs tâches de base (bien sûr, les contrôleurs sont redondés mais cela ne suffit pas dans tous les cas). Autre exemple de pragmatisme, il faut mesurer le réseau et s'assurer qu'on a réellement obtenu le résultat voulu.

D'autant plus que rien n'est gratuit dans ce monde cruel. La souplesse et la réactivité que fournissent les possibilités de configuration des équipements ont un coût, par exemple en rendant plus difficile, si on utilise des interfaces standard, d'optimiser les performances, puisqu'on ne pourra pas forcément toucher aux fonctions les plus spécifiques du matériel.

Lorsqu'on dit « SDN », on pense souvent immédiatement à OpenFlow, la plus connue des techniques vendues sous ce nom. Mais le SDN ne se réduit pas à OpenFlow et bien d'autres protocoles peuvent être utilisés pour de la configuration à distance (de PCEP - RFC 5440 - à NETCONF - RFC 6241 en passant par COPS-PR - RFC 3084 - ou RPSL - RFC 2622). OpenFlow n'est donc qu'un composant possible d'une boîte à outils plus large.

Enfin, lorsqu'on fait une analyse technique d'un mécanisme, il est important de voir quels sont les « non-objectifs », les choses qu'on n'essaiera même pas d'atteindre (dans le discours commercial, il n'y a pas de non-objectifs, la solution proposée est miraculeuse, guérit le cancer systématiquement et fait toujours le café) :

— La flexibilité rencontrera toujours des limites, ne serait-ce que les capacités du matériel (aucune quantité de SDN ne permettra à un routeur d'aller plus vite que ce que lui permettent ses interfaces réseaux et ses ASIC),
— La modularité et la programmabilité se heurteront aux limites du logiciel, comme la nécessité de tests (le "*software*" est nettement moins "*soft*" que ne s'imaginent les décideurs),
— Le contrôle absolu depuis un point central est très tentant pour certains mais il se heurte aux problèmes de passage à l'échelle et à la nécessité que les équipements aient assez d'autonomie pour gérer les déconnexions temporaires.

La section 4 du RFC discute ensuite plus en détail certains problèmes qu'on rencontre sur la route du SDN. Premier problème cité, toute automatisation d'un processus est vulnérable aux bogues dans le logiciel : c'est par exemple le syndrome du « robot fou » qui, ne se rendant pas compte que ses actions mènent à des catastrophes, les continue aveuglément (deux exemples, un avec Ansible <http://jpmens.net/2013/02/06/don-t-try-this-at-the-office-etc-sudoers/> et un avec Flowspec <http://seenthis.net/messages/118644>). Autre difficulté qui ne disparaitra pas facilement, l'amorçage du SDN : la première fois qu'on le met en place, les équipements et le contrôleur ne se connaissent pas et doivent apprendre leurs capacités. (Et utiliser le réseau pour configurer le réseau soulève plein de problèmes intéressants...)

Gérer un réseau, ce n'est pas seulement le configurer et faire en sorte qu'il démarre. Il faut aussi assurer le débogage, la maintenance, les statistiques, la détection (et la correction) de problèmes, tout ce qui est désigné sous le sigle OAM ("*Operations And Management*", cf. RFC 6291). Le SDN doit prendre cela en compte.

On l'a vu, un des principes du SDN est la concentration de l'« intelligence » en un point central, le PDP ("*Policy Decision Point*", là où on configure les services attendus). Cela pose des problèmes pour la résilience du réseau ou tout simplement pour son passage à l'échelle. Un mécanisme permettant d'avoir plusieurs PDP synchronisés semble donc nécessaire. En parlant de passage à l'échelle, le RFC signale aussi que certaines promesses du SDN vont être difficiles à tenir. Par exemple, avoir des machines virtuelles ayant des politiques de QoS différentes, et migrant librement d'un "*data center*" à l'autre est faisable en laboratoire mais ne va pas passer à l'échelle facilement.

Enfin, il ne faut pas oublier les risques, souvent négligés lors de la présentation d'une belle technologie qui brille. Le contrôleur va devenir le point faible du réseau, par exemple. Et la sécurité du SDN n'a pas encore été tellement mise à l'épreuve (section 6 du RFC).

Merci à Mohamed Boucadair et Christian Jacquenet pour leur relecture.