RFC 7920 : Problem Statement for the Interface to the Routing System

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 21 juillet 2016

Date de publication du RFC : Juin 2016

https://www.bortzmeyer.org/7920.html

Autrefois, la configuration des routeurs était relativement statique. On indiquait la politique de routage (le coût de tel ou tel lien, par exemple), les préfixes IP, les pairs BGP, parfois des routes statiques, et le routeur parlait avec ses copains routeurs, construisant ses tables qu'il allait ensuite utiliser pour la transmission des paquets. La configuration n'était pas modifiée tous les jours et quand c'était nécessaire, on se connectait à tous les routeurs qu'il fallait changer et on modifiait la config. Dans les organisations plus modernes, on édite la config, on la "*commite*" dans un VCS et on la "*pushe*" vers les routeurs avec Ansible ou un truc équivalent. Aujourd'hui, même cela ne suffit pas, on voudrait être plus agile. On voudrait modifier la configuration des routeurs à peu près en temps réel, pour répondre à des considérations de business (créer un nouveau service pour un client, profiter de variations de prix chez les transitaires...) ou à des problèmes de sécurité (déployer des filtres subtils). Cette exigence nécessite une interface vers le routeur, utilisable par des programmes. C'est le projet **I2RS**, "*Interface To the Routing System*". Ce premier RFC du groupe de travail <https://tools.ietf.org/wg/i2rs> décrit précisement le problème qu'on essaie de résoudre. (Notez que le "*buzzword*" SDN n'apparait pas une seule fois dans ce RFC...)

C'est que les réseaux modernes sont grands et complexes : il n'est plus possible de faire les choses à la main en se connectant sur chaque routeur isolément. Il faut donc automatiser, et il faut donc qu'un programme puisse se connecter aux routeurs et changer leurs configurations. Cela se fait bien sûr depuis longtemps (cf. Rancid et l'exposé de Joe Abley à NANOG <https://www.nanog.org/meetings/nanog26/presentations/stephen.pdf> et l'annexe A de notre RFC qui liste les solutions existantes), mais, en l'absence d'interface normalisée, c'était assez pénible à programmer et maintenir. Il s'agit donc de standardiser ce qui existe déjà, ce qui ne devrait pas être une tâche insurmontable.

La terminologie utilisée par I2RS est décrite dans un autre RFC, le RFC 7921 [1]. Pour le résumer (section 2 du RFC) : le routeur contient un **agent I2RS**, qui sait parler aux différents composants du routeur

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7921.txt

(sa base de données indiquant les différents liens connectés, son système de gestion de la configuration, sa base de routes - RIB, etc). Le logiciel qui pilote les changements est le **client I2RS**. Il y aura sans doute un seul client pour beaucoup d'agents, installé dans le système de gestion du réseau. Entre le client et l'agent, le protocole I2RS, qui sera normalisé dans les futurs RFC du groupe de travail I2RS <https://tools.ietf.org/wg/i2rs>. A priori, le client sera juste un intermédiaire pour des **applications** qui piloteront le routage (le but du découplage client/application étant d'éviter que chaque application doive parler I2RS : elles pourront interagir avec le client au-dessus d'un protocole plus classique comme REST).

Pour que le client puisse parler intelligemment à l'agent, il devra avoir en tête un **modèle de données** décrivant le routeur, ce qu'il sait faire, ce qu'on peut lui demander.

La section 3 de notre RFC présente la nécessité de ce modèle de données. Un tel modèle avait déjà été développé pour le projet ForCES (RFC 3746), en se focalisant sur la transmission, alors que I2RS est surtout intéressé par le routage (le plan de contrôle, accès à la RIB, etc).

Comme le note la section 4, un logiciel qui voudrait vraiment donner des instructions au routeur devrait apprendre la topologie du réseau, quels sont les liens physiques ou virtuels auxquels le routeur est connecté, leur état, etc. C'est évidemment le routeur qui connait le mieux cette information et il est donc nécessaire de lui demander.

L'application aura souvent besoin de connaitre en outre le trafic réellement échangé. IPFIX (RFC 5470) est certainement utilisable pour cela, si l'application peut le configurer dynamiquement.

Enfin, la section 5 rassemble les « points divers ». Elle rappelle qu'I2RS n'impose pas forcément le développement d'un nouveau protocole ; si un protocole, ou un ensemble de protocoles existant(s) suffit, c'est parfait (l'annexe A du RFC propose des pistes en ce sens). Mais la solution doit penser à :

- Permettre des opérations asynchrones : pas question d'attendre la fin d'une opération avant de commencer la suivante.
- Bonne granularité des opérations ; si on veut changer la configuration pour **un** préfixe IP, il ne faut pas verrouiller tout le routeur.
- Possibilité que plusieurs clients parlent au routeur en même temps, avec le minimum de coordination entre eux.
- Débit maximum élevé ; 10 000 opérations par seconde ne devraient pas être un problème.
- Faible latence ; les opérations simples devraient pouvoir se faire en bien moins qu'une seconde (contrairement à, par exemple, un changement de configuration du routeur avec l'interface CLI des routeurs actuels, où le "*commit*" peut être très long).
- Possibilité de filtrer l'information au départ du routeur, pour que l'application n'ait pas à tout récupérer avant le traitement.
- Et, bien sûr, sécurité ; un système mettant en œuvre I2RS va pouvoir modifier la configuration du routeur, ce qui est potentiellement très dangereux. Il faut donc pouvoir authentifier et autoriser les accès.

Voici pour le cahier des charges. L'annexe A propose quelques pistes qui vont en ce sens. À l'heure actuelle, l'interface la plus générale des routeurs est la CLI. Elle permet d'apprendre l'état du routeur et de changer sa configuration. Voici un exemple sur un Juniper :

```
bortzmeyer@MX-1> show interfaces

...

Physical interface: ge-1/1/9, Enabled, Physical link is Up
  Interface index: 177, SNMP ifIndex: 531
  Description: To infinity and beyond
```

```
  Link-level type: Ethernet, MTU: 1514, MRU: 1522, Speed: 1000mbps, BPDU Error: None,
  MAC-REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online
  Pad to minimum frame size: Disabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x0
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 5c:5e:ab:0e:4b:f1, Hardware address: 5c:5e:ab:0e:4b:f1
  Last flapped   : 2016-02-12 11:31:56 CET (22w5d 23:10 ago)
  Input rate     : 672 bps (1 pps)
  Output rate    : 672 bps (1 pps)
  Active alarms  : None
  Active defects : None
  Interface transmit statistics: Disabled

  Physical interface: xe-0/0/2, Enabled, Physical link is Up
  Interface index: 156, SNMP ifIndex: 510
  Link-level type: Ethernet, MTU: 1514, MRU: 1522, LAN-PHY mode, Speed: 10Gbps, BPDU Error: None,
  MAC-REWRITE Error: None, Loopback: None, Source filtering: Disabled, Flow control: Enabled
  Pad to minimum frame size: Disabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x0
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 5c:5e:ab:0e:4b:72, Hardware address: 5c:5e:ab:0e:4b:72
  Last flapped   : 2016-07-07 14:28:31 CEST (1w6d 21:11 ago)
  Input rate     : 0 bps (0 pps)
  Output rate    : 0 bps (0 pps)
  Active alarms  : None
  Active defects : None
  PCS statistics                      Seconds
    Bit errors                             0
    Errored blocks                         1
  Interface transmit statistics: Disabled

...
```

Ce "*shell*" n'est pas normalisé : chaque marque de routeur a le sien. Comme l'information retournée n'est pas structurée, si on veut la traiter depuis un programme, il faut faire du "*scraping*", ce qui est pénible et peu gratifiant (d'autant plus que le « format » peut changer sans prévenir lors de la sortie d'une nouvelle version du système d'exploitation du routeur).

Les routeurs ont aussi des interfaces reposant sur un protocole et des données structurées. La plus connue est SNMP. SNMP est très utilisé pour récupérer de l'information (état des interfaces, quantité de trafic qui passe) mais, bien qu'il permette en théorie la configuration des équipements réseau, en pratique, cette possibilité a été très peu utilisée. Les raisons de cette non-utilisation sont nombreuses (complexité, absence de sémantiques avancées comme le regroupement de plusieurs changements dans une « transaction » unique, sur laquelle on peut revenir globalement, etc). SNMP ne semble donc pas envisageable pour I2RS.

Enfin, cette annexe A cite Netconf comme étant sans doute la solution disponible la plus proche, même si elle n'est pas parfaite et aurait besoin d'être complétée (voir les travaux en cours dans le RFC 7921).