traceroute depuis les sondes Atlas

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 17 juillet 2013

https://www.bortzmeyer.org/traceroute-atlas.html

Lorsqu'on débogue un problème réseau sur l'Internet, on utilise souvent traceroute. Il vous indique jusqu'où vous allez et par quel chemin. Le problème est que cela ne dit pas ce que voit le pair en face, d'autant plus le trafic n'est pas forcément symétrique. Il est souvent utile de pouvoir commander un traceroute **depuis** le pair vers **votre** machine. Les sondes Atlas <https://www.bortzmeyer.org/atlas-udm.html> permettent cela.

C'est quand même nettement plus pratique que de demander aux gens d'en face de le faire (d'autant plus qu'ils peuvent n'avoir pas le temps, ou simplement pas la compétence pour le faire corrrectement.) On peut déclencher ce traceroute depuis une sonde donnée, depuis un ensemble de sondes (identifiées par leur pays, par leur numéro de système autonome, etc). Cela peut se faire via l'interface Web des UDM ("*User-Defined Measurements*" <https://atlas.ripe.net/doc/udm>) ou via un programme (j'utilise traceroute.py <https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/traceroute.py>). Une fois la mesure faite, on peut, comme avec les autres mesures Atlas, télécharger le résultat en format JSON et l'analyser. Voici un extrait des résultats de la mesure #1013442 montrant les deux premiers sauts :


```
 "prb_id": 2014,
 "result": [
   {
     "result": [
       {
         "rtt": 1.6990000000000001,
         "ttl": 255,
         "from": "185.12.232.2",
         "size": 56
       },
      ...
     ],
     "hop": 1
   },
   {
     "result": [
       {
         "rtt": 1.601,
         "ttl": 255,
```

```
      "from": "185.12.232.3",
      "size": 56
    },
   ...
  ],
  "hop": 2
},
```

Bon, le JSON, ce n'est pas forcément très agréable à lire (mais c'est documenté <https://atlas.ripe.net/doc/data_struct#v4460_traceroute>). Alors, Jan Hugo Prins a fait un excellent script <https://github.com/RIPE-Atlas-Community/ripe-atlas-community-contrib/blob/master/json2traceroute.py> qui traduit cela en format plus lisible :

```
% python json2traceroute.py 1013442.json
...
From:  130.79.86.251    2259    FR-U-STRASBOURG OSIRIS - UNIVERSITE DE STRASBOURG
Source address:  130.79.86.251
Probe ID:  2279
1    130.79.86.253    2259    FR-U-STRASBOURG OSIRIS - UNIVERSITE DE STRASBOURG    [2.922, 2.542, 2.686]
2    193.51.183.130    2200    FR-RENATER Reseau National de telecommunications pour la Technologie    [2.66
3    193.51.189.85    2200    FR-RENATER Reseau National de telecommunications pour la Technologie    [10.08
4    193.51.189.161    2200    FR-RENATER Reseau National de telecommunications pour la Technologie    [11.0
5    193.51.180.14    2200    FR-RENATER Reseau National de telecommunications pour la Technologie    [76.38
6    194.68.129.118    None    None    [10.383, 10.125, 9.975]
7    217.70.176.214    29169    GANDI-AS Gandi SAS    [11.094, 10.871, 10.186]
8    217.70.176.250    29169    GANDI-AS Gandi SAS    [32.749, 9.386, 10.046]
9    217.70.190.232    29169    GANDI-AS Gandi SAS    [11.058, 10.215, 9.371]
...
```

On voit, pour chaque saut, l'adresse IP du routeur, le numéro d'AS, le nom de l'opérateur et le RTT pour les trois tests faits par sauts. (Au saut 6, le SFINX n'a pas été reconnu, probablement parce que ce préfixe n'est pas annoncé dans la DFZ.) Ce script nécessite d'installer d'abord la bibliothèque cymruwhois <http://pythonhosted.org/cymruwhois/>. Elle permet de convertir les adresses IP en numéros de système autonome et en noms d'opérateurs. Pour éviter de charger le serveur whois qui fait ces conversions, elle utilise un cache local. Le délai d'attente de cette bibliothèque est bien trop court et vous aurez donc souvent des erreurs :

```
socket.timeout: timed out
```

Dans ce cas, relancez la commande (le cache vous évitera de tout recommencer).

La mesure #1013442, citée plus haut, a été faite vers l'une des machines qui porte ce blog. J'ai utilisé l'option « traceroute avec ICMP » car le pare-feu filtre l'UDP. La commande exacte était :

```
% python traceroute.py -v -r 100 --protocol ICMP 217.70.190.232
```

Il existe plusieurs autres paramètres réglables pour lancer les mesures traceroute, n'hésitez pas à consulter la documentation <https://atlas.ripe.net/doc/measurement-creation-api/>, section « "*Traceroute-Specific Properties*" ».

Une des options les plus pratiques, à mon avis, est celle qui permet d'utiliser les mêmes sondes que dans une mesure précédente. Mettons qu'on mesure les performances DNS avec les Atlas. Puis on veut comprendre pourquoi certaines sondes ont des problèmes. On fait un traceroute avec l'option « mêmes sondes que dans la mesure N » et on peut alors analyser par où elles passent.

Un autre mécanisme pour lancer des traceroutes à distance est le RING <https://ring.nlnog.net/>. Et, bien sûr, il y a traceroute.org mais on a peu d'outils de sélection, on ne peut pas automatiser, beaucoup des services listés sont en panne, etc.