SORBONNE UNIVERSITÉ

NETMET 2020

Projet de Géolocalisation des adresses IP

Auteurs :

Fabien MANSON
Alexandre MAZARS

Encadrant :

Matthieu GOUEL

I) Le choix de la solution

La solution originale

Dans un premier temps nous avons décidé de mettre en place un algorithme utilisant le réseau Planet-Lab pour réaliser des pings vers les adresses IP. Notre script était découpé en 2 parties principales, la récupération des valeurs de ping et le calcul des coordonnées géographiques approximatives. La première partie consistait en un script python qui se connectait à 5 nœuds du réseau Planet-Lab situé en différents endroits de la planète. Le script se connectait en ssh sur chacun des nœuds puis lançais une vingtaine de pings vers l'adresse IP cible, ensuite le script passait à un autre nœud et avant de retourner au premier nœud pour refaire des pings, il attendait environ une minute et ce pour avoir une valeur de RTT la plus cohérente possible. La valeur de RTT retenue est la moyenne du maximum des RTT. Ce temps est ensuite écrit dans un fichier texte avec le nom du nœud correspondant. Le script boucle sur la liste des IP donnée pour le projet et enregistre tous les résultats des requêtes dans le même fichier texte. La seconde partie du script traite les données enregistrées dans le fichier en calculant des cercles ayant pour centre chaque nœud utilisé pour les mesures et pour rayon une distance en mètres calculée à partir du RTT mesuré. On obtient 5 cercles. Si ces cercles s'intersectent en une seule zone (appelée le centroïd) alors on est dans le meilleur cas si il y a plusieurs zones d'intersection, on prend l'intersection qui réunit le plus de cercles. Une fois que l'on a les coordonnées des cercles et le rayon on peut en déduire les équations des cercles et en calculant les points d'intersection de chaque cercle, on obtient une liste de point qui délimite le centroïd. On sauvegarde cette liste et on pour calculer le centre approximatif du centroïd on calculer simplement la moyenne des coordonnées (ces coordonnées sont d'ailleurs exprimées en coordonnées géographiques). Apres le calcul, on obtient un point qui est désigné comme le best-guest c'est-à-dire la meilleure estimation de la localisation de l'adresse IP. On écrit ce résultat dans un fichier texte au format accepté par l'api de vérification :

{''192.168.0.1'' : [x_coord, y_coord]}

Ce fichier est créé dans le but de vérifier automatiquement les résultats obtenus.

Le problème que l'on a rencontré pendant la mise en place de cette solution c'est que notre accès à Planet-Lab a été supprimé, nous avons perdu les permissions de nous connecter en ssh aux nœuds… Nous avons testé de regénérer la clé ssh pour se connecter mais sans succès, la connexion est refusée même si la clé est validée sur l'interface web du site internet de Planet-Lab. Nous avons dû changer totalement la solution adoptée car réaliser des pings depuis notre machine s'avère très peu exact, de plus il faudrait au moins deux machines suffisamment éloignée pour obtenir des mesures exploitables.

La solution alternative de secours

Nous avons donc changé de méthode est opté pour une analyse approfondie des noms de domaine que l'on obtient en faisant une requête DNS inverse sur l'IP. Notre programme d'analyse est découpé en trois grandes parties principales. La première partie récupère le nom de domaine associé à chaque IP en faisant une requête DNS inverse et stock le résultat dans un fichier texte *(cf. Figure 1)* contenant sur chaque ligne l'IP suivie du nom de domaine associé ou « HostUnknown » si la requête ne donne rien. Cette partie charge ensuite en mémoire plusieurs dictionnaires nécessaires pour la suite du programme. Le premier dictionnaire est un dictionnaire associant les codes ISO de tous les pays du monde avec leur nom complet *(cf. Figure 2)*. Ce dictionnaire est chargé à partir d'un fichier texte *(cf. Figure 3)* contenant pour chaque pays une liste d'informations détaillée comme les régions et sous régions ainsi que divers code d'identification que nous n'utilisons pas. Le second dictionnaire est un dictionnaire associant pour chaque code de pays la liste de toutes les villes principales. Ce dictionnaire étant trop gros pour être affiché lisiblement dans ce rapport, il est disponible sur la page GitHub du projet sous le nom de CountryCities.txt. Ce dictionnaire est construit avec des fonctions de scraping et un parser HTML. Une fonction va chercher le code html d'un page web contenant la liste des villes sous forme de table html et traite ce code à l'aide d'un parser pour en extraire la liste des villes. Il y a une page par pays donc cela représente 250 pages à analyser. Comme cela représente beaucoup de donnée en une seule connexion, le serveur hébergeant les données envoie régulièrement des reset de connexion pour des raisons de

```
79.219.47.226;p4fdb2fe2.dip0.t-ipconnect.de
91.45.168.37;p5b2da825.dip0.t-ipconnect.de
62.159.96.154;HostUnknown
93.202.251.43;p5dcafb2b.dip0.t-ipconnect.de
80.152.240.115;p5098f073.dip0.t-ipconnect.de
217.249.58.181;pd9f93ab5.dip0.t-ipconnect.de
90.223.193.3;uk-slo-as5607.anchors.atlas.ripe.net
87.173.87.138;p57ad578a.dip0.t-ipconnect.de
91.21.221.146;p5b15dd92.dip0.t-ipconnect.de
202.52.0.25;atlas-anchor.nren.net.np
84.155.78.222;p549b4ede.dip0.t-ipconnect.de
84.139.224.125;p548be07d.dip0.t-ipconnect.de
217.91.97.156;pd95b619c.dip0.t-ipconnect.de
79.226.82.253;p4fe252fd.dip0.t-ipconnect.de
87.238.48.243;no-osl-as39029.anchors.atlas.ripe.net
84.131.205.253;p5483cdfd.dip0.t-ipconnect.de
79.231.173.223;p4fe7addf.dip0.t-ipconnect.de
217.251.128.172;pd9fb80ac.dip0.t-ipconnect.de
```

Figure 1 Fichier des IP et noms de domaine associés

```
{
    'AX' : Åland Islands
    'AL' : Albania
    'DZ' : Algeria
    'AS' : American Samoa
    'AD' : Andorra
    'AO' : Angola
    'AI' : Anguilla
    'AQ' : Antarctica
    'AG' : Antigua and Barbuda
    'AR' : Argentina
```

Figure 2 Dictionnaire des codes de pays

```
name,alpha-2,alpha-3,country-code,iso_3166-2,region,sub-region,intermediate-region,region-code,sub-region-code,intermediate-region-code
Afghanistan,AF,AFG,004,ISO 3166-2:AF,Asia,Southern Asia,"",142,034,""
Åland Islands,AX,ALA,248,ISO 3166-2:AX,Europe,Northern Europe,"",150,154,""
Albania,AL,ALB,008,ISO 3166-2:AL,Europe,Southern Europe,"",150,039,""
Algeria,DZ,DZA,012,ISO 3166-2:DZ,Africa,Northern Africa,"",002,015,""
American Samoa,AS,ASM,016,ISO 3166-2:AS,Oceania,Polynesia,"",009,061,""
Andorra,AD,AND,020,ISO 3166-2:AD,Europe,Southern Europe,"",150,039,""
```

Figure 3 Fichier d'information des pays

```
AF AX AL DZ AS AD AO AI AQ AG AR AM AW AU AT AZ BS BH BD BB BY BE BZ BJ BM BT BO BQ BA BW BV BR IO BN BG
ER EE SZ ET FK FO FJ FI FR GF PF TF GA GM GE DE GH GI GR GL GD GP GU GT GG GN GW GY HT HM VA HN HK HU IS
ML MT MH MQ MR MU YT MX FM MD MC MN ME MS MA MZ MM NA NR NP NL NC NZ NI NE NG NU NF MK MP NO OM PK PW PS
SB SO ZA GS SS ES LK SD SR SJ SE CH SY TW TJ TZ TH TL TG TK TO TT TN TR TM TC TV UG UA AE GB US UM UY UZ
```

Figure 4 Fichier de log permettant de retenir l'état d'avancement lors de la fermeture ou du reset de la connexion

sécurité. C'est pourquoi notre fonction maintient un fichier de log *(cf. Figure 4)* contenant la liste des pays déjà traités. Les données obtenues sont elles aussi écrites dans un fichier texte avec une ligne par pays contenant le code ISO à deux lettres du pays suivi des villes le tout séparé par des points-virgules. Cela permet de relancer la fonction de téléchargement sans recommencer du début. En tout il a fallu lancer 5 fois la fonction et attendre environ 10 minutes pour obtenir la base de données complète. Cette base est nommée « CountryCities.txt ». Une autre fonction permet de charger ce dictionnaire en mémoire à partir du fichier texte.

Une fois que les dictionnaires sont chargés en mémoire, la seconde partie du programme entre en jeu. Cette partie est dédiée à l'analyse des noms de domaines récupérés. Nous avons mis en place plusieurs fonctions, la première analyse chaque mot d'une longueur de deux lettres du nom de domaine et cherche un code de pays à 2 lettres, la seconde cherche quant à elle un nom de pays en toute lettres dans le nom de domaine. Ces deux fonctions retournent le même type de résultat à savoir un dictionnaire avec le code du pays qui est trouvé suivi d'un score de certitude donné en pourcentage. Ce score correspond au nombre de lettre identique entre le mot extrait du nom de domaine et le code ou le nom complet du pays. Par exemple pour le nom de domaine suivant :

p549b4ede.dip0.t-ipconnect.de

La première fonction va analyser uniquement le mot « de » et trouver qu'il s'agit du code pays de l'Allemagne (DE). Le score associé à ce résultat est de 100%. La seconde fonction ne trouvera aucun nom de pays dans le nom de domaine et retournera le dictionnaire non modifié. Voici un second exemple avec le nom de domaine suivant :

ripe-atlas-anchor.franceix.net

La première fonction ne va trouver aucun code pays car il n'y a pas de mot de 2 lettres dans ce nom de domaine. Elle retourne un dictionnaire vide. La seconde fonction va quant à elle trouver le mot « france » qui correspond à un nom de pays dans le dictionnaire des noms de pays. Cependant le mot « France » n'est qu'une sous chaine du mot complet qui est « franceix » le score calculé ne sera donc pas de 100% mais de (6/8)*100 = 75% de certitude. La fonction ajoutera donc 'FR' : 75.0 au dictionnaire des résultats. Cette fonction dispose d'une condition pour ajouter les résultats au dictionnaire. Nous avons décidé arbitrairement de fixer le seuil minimum de certitude à 75%. Tout score supérieur ou égal 75% sera ajouté aux résultats, les autres ne seront pas pris en compte car cela veut dire que seulement une partie insuffisante du mot a été détectée comme étant un nom de pays. Le dictionnaire des résultats est gardé en mémoire le temps que la troisième partie du programme agisse.

Cette troisième et dernière partie est en charge de la résolution des noms en coordonnées géographiques ainsi que de la vérification et de l'enregistrement des résultats dans un fichier texte. Dans un premier temps il faut traduire le nom du pays en coordonnées géographiques. Pour ce faire nous utilisons le module Nominatim présent dans

geopy.geocoders. Une fonction dédiée, prenant une adresse sous forme de string en paramètre, s'occupe de faire une ou plusieurs requêtes pour obtenir les coordonnées géographiques. Tant que la requête échoue, la fonction relance cette dernière dans la limite de 20 tentatives après quoi la localisation est déclarée comme introuvable. La fonction retourne les coordonnées sous forme d'un tuple (latitude, longitude) ou 'None' en cas d'erreur. Une fois qu'une localisation est obtenue, une autre fonction se charge de vérifier le résultat via une simple requête POST vers le port 8000 de l'api Planet-Lab contenant un micro dictionnaire avec l'adresse IP suivie des coordonnées géographiques. La réponse est ensuite traitée pour n'extraire que le texte correspondant au score ainsi que sa valeur. Cette chaine de caractère ainsi obtenue est finalement retournée par la fonction. Une fois que le score est

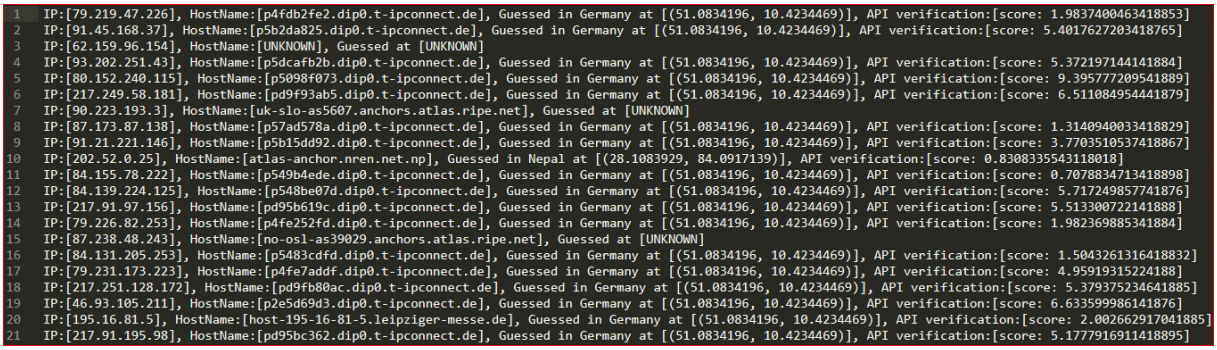

Figure 5 Fichier résultat obtenu à la fin du traitement de la liste des IP

obtenu, la fonction principale s'occupe d'afficher correctement le résultat du traitement de chaque adresse et peut, dépendamment d'un paramètre spécifié, écrire cet affichage dans un fichier texte nommé « GeoIPyyyy-mm-dd.txt » *(cf. Figure 5)*. Cette fonction maintient aussi des statistiques basiques, comme le nombre total d'adresse traité ainsi que le nombre d'adresse dont le nom de domaine est inconnu ainsi que le nombre de localisation qui ne sont pas suffisamment précises *(cf. Figure 6)*.

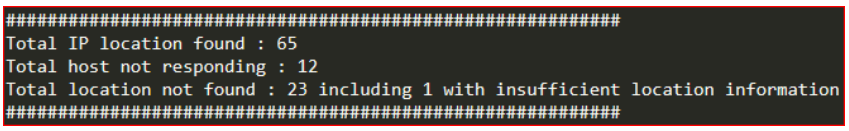

Figure 6 Statistiques de traitement des IP

II) Analyse des solutions

La solution originale : les avantages

Il est évident que la précision de cette solution est bien plus grande que celle que notre solution de secours car avec cette solution on peut géo localiser à plusieurs échelles, généralement au niveau du pays mais bien plus souvent au niveau de la région (comme ce que nous avons fait au Lab 9 de cette UE). La prise de mesure n'est pas si compliqué qu'elle n'y parait il faut simplement automatiser le lancement de commande ping sur les différents nœuds de mesure.

La solution originale : les inconvénients

Un des inconvénients de cette technique c'est qu'elle nécessite beaucoup de données de mesures faite depuis plusieurs endroits différents dans le monde (au moins trois). Plus il y a de mesures et plus l'estimation est fiable. Le second inconvénient est lié au calcul de distance résultant des mesures. Les calculs sont simples si on projette les cercles de délai dans un plan en deux dimensions mais si on passe à une projection sphérique (Oui la terre est ronde, dehors les platistes !) c'est une autre histoire, les calculs mathématiques sont bien plus complexes. Un derniers inconvénient lié à la prise de mesure c'est la charge réseau, en faisant un grand nombre de mesure on peut cause des engorgements sur le réseau…

La solution de secours : les avantages

Le principal avantage est le faible coût en requête réseau. Oui, lors de la récupération des listes de villes et de pays le réseau est très sollicité mais une fois que les bases de données sont sauvegardées dans des fichiers, il n'y plus qu'une requête DNS inverse par adresse IP. Ce qui est rien comparé aux dizaines voire centaines de ping request envoyé vers chaque IP dans la solution originale. De plus la récupération du nom de domaine est très rapide, il faut quelques secondes pour une centaine d'adresse. Une fois que la base de noms de domaine est créée, on peut l'analyser en local. Cette solution est aussi bien plus simple en termes de calcul car, il n'y en a aucun en tout cas concernant les distances géographiques. Elle est aussi assez rapide, la partie qui prend le plus de temps c'est la résolution des adresses physiques en coordonnées GPS via geopy.

La solution de secours : les inconvénients

Le principal inconvénient est le manque de précision des résultats. En effet en ne se basant que sur l'analyse des noms de domaine cette technique est dépendant de la bonne fois des enregistreurs de nom de domaine. Il suffit qu'il n'y ait aucun indice comme un nom de pays ou de ville dans le nom de domaine pour que l'on ne puisse tout simplement pas localiser l'adresse IP. La première solution est intéressante dans ce cas car elle ne rencontre pas ce problème. Un second inconvénient lié à l'analyse du nom de domaine peut être la complexité de l'algorithme. Dans notre solution on ne fait qu'une analyse basique pour trouver un code de pays ou un nom de pays mais pour avoir de meilleurs résultats il faudrait mettre en place un algorithme intelligent capable de trouver la localisation en analysant le nom de domaine à différentes échelles d'abord comme un seul groupe puis en zoomant sur chaque terme. Par exemple il n'est pas rare de tomber sur un nom de domaine contenant le nom d'une société par exemple « belnet ». Cet algorithme devrait deviner qu'il s'agit d'une IP probablement située en Belgique. Le cout en temps de calcul pourrait alors être long s'il faut analyser en détail chaque nom de domaine. Un dernier inconvénient concernant la traduction des noms de pays et adresse GPS c'est le fait que l'API geopy retourne toujours le centre du pays demandé, ce qui dans la majorité des cas ne pose pas trop de problème bien que cela manque de précision. En revanche cela pose un gros problème pour les pays ayant une grande surface terrestre comme la Russie les États-Unis ou l'Inde. Le score obtenu est y est bien trop élevé.

III) Conclusion

Nous somme conscient que notre solution atteint vite ses limites cependant, elle reste efficace et rapide si le nom de domaine est bien formé, mais inutile dans le cas contraire. Notre solution ne dispose pas non plus beaucoup d'axes d'amélioration mise à part changer l'algorithme d'analyse des noms de domaine pour augmenter sa fiabilité. Peut-être qu'en combinant les deux méthodes on peut sans doute arriver à de très bons résultats. On analyse chaque nom de domaine avec la méthode DNS en premier lieux pour restreindre la zone géographique et ensuite on fait des mesures de délai et on calcul les distances associés pour avoir un emplacement plus précis.

Le code complet étant très long, nous ne l'avons pas mis en annexe. Il reste disponible en libre accès sur la page GitHub du projet : *https://github.com/FabienMnsn/GeoIP/*

Le fichier principal se nomme « GeoIP.py »
La fonction principale est à la ligne 61 et se nomme « readHostName(…) »