Projet 2 Java avancé M1 Miage Apprentissage

Toutes les routes mènent à Mido.

1 Dates importantes

- Rendu du projet : le mardi 19/12/ 2023, 23h.
- Soutenances : le vendredi 22/12/ 2023, à partir de 08h30.

2 Description

Depuis un nœud d'Internet, la commande traceroute permet de connaitre la liste des nœuds formant un des chemins du réseau internet vers un autre nœud. Par exemple :

```
traceroute to www.google.fr (173.194.66.94), 30 hops max, 60
 byte packets
 1  10.1.3.254    (10.1.3.254)   0.782 ms   0.784 ms   0.777 ms
 2  lamsade-gw.lamsade.dauphine.fr      (193.48.71.254)   1.661 ms   1.683 ms   1.659 ms
 3  192.168.2.1     (192.168.2.1)   1.658 ms   1.656 ms   1.932 ms
 4  interco-9.01-auteuil.rap.prd.fr         (195.221.127.153)   2.389 ms   2.841 ms   2.385 ms
 5  pe-odeon.rap.prd.fr      (195.221.125.25)   2.839 ms   2.836 ms   2.832 ms
 6  vl260-te4-4-paris1-rtr-021.noc.renater.fr          (193.51.186.102)   3.534 ms   2.754 ms   2.703 ms
 7  te0-0-0-3-paris1-rtr-001.noc.renater.fr          (193.51.177.24)   6.088 ms * *
 8  te0-6-0-2-paris2-rtr-001.noc.renater.fr          (193.51.177.89)   4.067 ms   4.046 ms   195.760 ms
 9  * * *
10  google-te1-6-paris2-rtr-021.noc.renater.fr            (193.51.182.197)   194.530 ms   193.455 ms   193.392 ms
11  72.14.238.234     (72.14.238.234)   196.899 ms   202.685 ms   204.217 ms
12  *  209.85.245.70   (209.85.245.70)   4.357 ms   2.992 ms
13  216.239.51.198     (216.239.51.198)   8.168  ms   209.85.242.229   (209.85.242.229)   18.170  ms   209.85.248.202   (209.85.248.202)   8.000 ms
14  72.14.238.215     (72.14.238.215)   7.659  ms   66.249.95.238   (66.249.95.238)   8.824  ms   216.239.49.45   (216.239.49.45)   8.721 ms
15  * * *
16  we-in-f94.1e100.net      (173.194.66.94)   8.000 ms   7.446 ms   7.780 ms
```

Chaque ligne correspond à un routeur. On y trouve un numéro de ligne, une adresse IP (4 octets !), parfois un nom (pas sur les lignes 1, 3...), des temps de réponse (ignorés dans ce projet). Il y a parfois plusieurs adresses pour une ligne (par ex. ligne 13) – comme traceroute fait 3 essais pour chaque étape, il y a parfois 3 résultats. À vous de voir et de justifier comment vous prenez en compte ceci. Les étoiles correspondent à des absences de réponse. Un même traceroute peut donner des résultats différents – le graphe d'internet n'est pas figé. Les chemins donnés ne sont pas forcément les plus courts non plus.

3 Ce qui est demandé

Le but est de constituer, visualiser et localiser en temps réel le graphe non-orienté d'internet en effectuant des traceroute automatiquement. Chaque sommet du graphe sera une adresse IP. **Il y a une arête entre deux sommets A et B s'il y a un traceroute qui indique A et B à la suite. Dans le cas d'étoiles entre A et B, on peut considérer que A et B sont reliés tout de même (mais peut-être afficher cette arête avec une autre couleur sur le globe...)** Pour chaque sommet (IP), on voudra pouvoir connaitre ses voisins (affichage des IPs), sa distance *minimale* (cela peut être différent pour des traceroute différents) à l'origine (la machine sur laquelle sont lancés les traceroute). On voudra en plus visualiser ce graphe sur un globe terrestre en géolocalisant chaque sommet du graphe (les adresses IP) grâce à des services externes – comme ces derniers ne sont pas forcément très pertinents, l'utilisateur veut pouvoir choisir le service utilisé.

Le projet doit s'exécuter sous Linux et Windows et optionnellement sur les autres plateformes où Java et les services externes fonctionnent (*a priori* MacOS), sans avoir à modifier le programme.

On veut pouvoir :

- exécuter plusieurs traceroute en parallèle pour assembler plus rapidement les informations,
- choisir à la volée combien de traceroute s'effectuent simultanément,
- choisir combien de traceroute sont effectués par minute (de 1 à infini) (optionnel),
- ajouter à la volée des nouvelles cibles pour traceroute (par IP, par nom, par tranche d'adresses ou depuis un fichier contenant des adresses à explorer),
- ajouter des cibles aléatoires (des adresses IP aléatoires (mais pas les adresses réservées – comment les traitez-vous ?)),
- changer de service de géolocalisation,
- visualiser des statistiques sur le graphe construit (nombre de sommets vus, de traceroute lancés, classement par distance, etc),
- indiquer les voisins d'un sommet du graphe,
- visualiser au fur et à mesure, en temps réel, le graphe (y compris pendant qu'un traceroute s'exécute) sur le globe,
- déterminer le plus court chemin (en terme de sommets) dans votre graphe entre un point et l'origine (l'ordinateur où s'exécute le programme), et entre deux points (optionnel)
- exporter et importer (sans écraser l'existant) le graphe construit sous forme de 2 fichiers CSV (séparé par des virgules) : un premier avec la liste des sommets, leur distance à l'origine et leurs coordonnées GPS (1 par ligne), un second avec l'ensemble des couples de sommets reliés (1 couple par ligne) (optionnel),
- importer les résultats de traceroute effectués depuis une autre machine (on ignore alors les distances à l'origine) (optionnel).

Le projet utilisera la commande traceroute (ou équivalent) (voir la classe ProcessBuilder). L'utilisateur pourra choisir le nombre maximum de sauts (hop), le temps maximum d'attente par saut. Le traceroute est plus rapide si on désactive la résolution de nom DNS. Les résultats peuvent être plus ou moins intéressants si traceroute est utilisé en mode UDP, TCP ou ICMP.

3.1 Services externes

- Géolocalisation :
  - GeoIP. Télécharger la base de données et l'API Java.
  - HostIP. Faire un GET HTTP avec URLConnection pour avoir un retour en XML et utiliser SAX pour le parser. **VOUS DEVEZ OBLIGATOIREMENT UTILISER LA METHODE EN XML**.
  - Des services supplémentaires optionnellement.
- Visualisation :
  - Nasa WorldWind. Voir ici, et notamment les classes RenderableLayer, AnnotationLayer, GlobeAnnotation dans la javadoc.

Aucune autre bibliothèque externe (au JDK) est autorisée.

Il est conseillé de faire des tests unitaires (pertinents) tout au long du projet avec JUnit. "Tests should be written before the code. Test-driven development is a lot more fun than writing tests after the code seems to be working. Give it a try !"

La propreté du code, sa lisibilité et l'architecture choisie comptera largement dans la note finale.

4 Conditions de rendu

Votre projet est à rendre avant la date précisée plus haut à l'adresse mail suivante khitem.blidaoui@dauphine.psl.eu. Un seul envoi par binôme ! Une fois votre fichier envoyé, vous devez avoir un écran de confirmation avec votre fichier et la date de l'envoi. **Il y aura un point en moins par heure de retard, une heure entamée est due.** Le format de rendu est une archive au format **ZIP** contenant :

- Un répertoire *src* avec les sources de votre implémentation java.
- Un répertoire *classes* vide dans l'archive, devant contenir les classes du projet une fois compilé.
- Un répertoire *docs* contenant :
  - Une documentation pour l'utilisateur *user.pdf* décrivant à un utilisateur comment se servir de votre projet.
  - Une documentation pour le développeur *dev.pdf* , devant justifier les choix effectués, présenter l'architecture choisie, indiquer quelles ont été les difficultés rencontrées au cours du projet ainsi que la répartition du travail entre les membres du binôme. Ce rapport doit faire le point sur les fonctionnalités apportées, celles qui n'ont pas été faite (et expliquer pourquoi). Il ne doit pas paraphraser le code, mais doit rendre explicite ce que ne montre pas le code. Il doit montrer que le code produit a fait l'objet d'un travail réfléchi et minutieux (comment un bug a été résolu, comment la redondance dans le code a été évitée, comment telle difficulté technique a été contournée, quels ont été les choix, les pistes examinées ou abandonnées...).

Ce rapport est le témoin de vos qualités scientifiques mais aussi littéraires (style, grammaire, orthographe, présentation).

  - Un répertoire vide *doc* pour la javadoc.
- Un répertoire *lib* contenant le minimum de bibliothèques externes nécessaires à votre projet pour s'exécuter.
- Un fichier ant *build.xml* permettant de compiler, créer le jar, générer la javadoc, etc.
- Votre jar exécutable pouvant être lancés au moyen de la commande `java -jar`.

Votre projet doit pouvoir s'exécuter sans utiliser eclipse ! (Vous avez le droit d'utiliser java.library.path).

L'archive aura pour nom Nom1Nom2.zip, où Nom1 et Nom2 sont les noms des membres du binôme par ordre alphabétique. L'extraction de l'archive devra créer un dossier Nom1Nom2 contenant les élements précisés ci-dessus.

Il va sans dire que les différents points suivants doivent être pris en compte :

- Uniformité de la langue utilisée dans le code (anglais conseillé).
- Uniformité des conventions de nommage et de code (convention Java obligatoire).
- Projet compilant sans erreur et fonctionnant sur les machines de l'université.
- Gestion propre des différentes exceptions.
- La documentation ne doit pas être un copié-collé du code source du projet.
- Les sources doivent être commentées, dans une unique langue, de manière pertinente (pas de commentaire "fait un test" avant un if.).
- Le nom des variables, classes et méthodes doivent être choisis judicieusement.
- Le code devra être propre, les exceptions correctement gérées, les classes correctement organisées en packages. La visibilité des méthodes et champs doit être pertinente (privée ou non...).
- Le projet doit évidement être propre à chaque binôme. Un détecteur automatique de plagiat sera utilisé. Si du texte ou une portion de code a été empruntée (sur internet, chez un autre binome), il faudra l'indiquer dans le rapport. Tout manque de sincérité sera lourdement sanctionné.

La documentation (rapports, commentaires...) compte pour un quart de la note finale. On préférera un projet qui fonctionne bien avec peu de fonctionnalités qu'un projet bancal avec plus de fonctionnalités.

L'utilisation d'un gestionnaire de version type CVS, SVN ou GIT est conseillée (attention toutefois aux sites où le code est public). Riouxsvn semble être gratuit et privé mais je ne le connais pas spécialement.

4.1 Soutenance

Une soutenance d'un quart d'heure aura lieu binôme par binôme à la date précisée plus haut. Elle doit être préparée et menée par le binôme (i.e. fonctionnant parfaitement du premier coup, avec des jeux de tests pertinents etc). Des jeux de tests mettant en valeur la correction de votre projet sont grandement recommandée (à fournir au moment du rendu).

4.2 Binômes

-1 Nathan Zafizara-Benetrix Guillaume Pereira
- 2 Julie HUYEN Joëlle HUYEN
- 3 Elodie Dai Léonore Sarfati
- 4 Mustapha Doubabi Neveu Pierre
- 5 Meriame SLIMANE Leila Taghouti

- 6 Derouiche Wassim Sebastien Giret-imhaus
- 7 Hector CORBLET Cyril RAMEAUX
- 8 Jeanne-Emma Lefèvre Paul Malet
- 9 Arvinde SENGUTTUVAN Ilyess BATAL
- 10 Qinming JIANG Yang YANG
- 11 Abdelhalim Zerara Sid-Ali Amrani
- 12 MARTIN-DIPP Daryl KARAOUANE Guillaume
- 13 Ayoub Moqrich Mustafa Caglayan
- 14 BOURSIN Alice HADDAD Chirine
- 15 CHEN Luc INCORVAIA Tony
- 16 Nelson PROIA Binh Minh NGUYEN
- 17 ANDRIN Mathieu BOKA Ricardo TORDJMAN Noam

5 Questions/Réponses

- **Les sommets sont seulement les IP machines de destinations ou toutes les machines et IP rencontrées lors du traceroute ?** Tout ce qui a été rencontré.
- **Les deux services de géolocalisation sont à utiliser au minimum ?** Oui, mais l'utilisateur choisit l'un ou l'autre à un instant t pour ne pas avoir de conflit.
- **Lorsqu'il y a plusieurs adresses IP sur une même ligne, y a -t il une unique manière correcte de le traiter ?** à vous de voir.
- **Pour le graphe, faut-il considérer un graphe orienté ?** Un graphe non-orienté.
- **Et par conséquent les voisins d'un sommet sont-ils ses "successeurs" ou bien tout les sommets relié à lui par une arrête ?** Tous les ommets.
- **Pour le plus court chemin,les arrêtes sont-elles pondérées par la distance séparant les deux routeurs ou faut-il prendre en compte uniquement le nombre de sommets séparant la source et la cible ? (s'agit-il de calculer la distance en terme de sommets parcourus ou la distance en terme de km ?)** En terme de sommets, restons simple...
- **Pour " l'ajout à la volée des nouvelles cibles pour traceroute", s'agit-il de changer de cible lorsque des traceroutes s'exécutent ?** Non, un traceroute commencé se termine. Il s'agit de pouvoir ajouter des cibles pour traceroute lorsque le programme a déjà commencé (et donc pas uniquement avant le début).
- **Ces cibles doivent-elles faire partie des sommets existants ? (sommets présents dans le graphe déjà construit)** Non, n'importe quelle cible.
- **Qu'entendez-vous par "ajouter des cibles aléatoires" ?Aléatoire parmi quelles adresses IP, les sommets du graphe existants ?** Non, aléatoire sur l'ensemble des adresses IP possibles moins les adresses interdites.
- **Nous aimerions savoir si lorsqu'on lance un traceroute et que nous avons plusieurs adresse ip mais qui se trouve à la même localisation(coordonnées GPS), voulez-vous plusieurs sommets sur la map ou bien un seul par localisation ?** 1 IP = 1 sommet
- **Faut il que nous lancions un seul traceroute pour une seule adresse ou bien plusieurs pour une seule et même adresse et que nous tracions ensuite sur la map le plus court chemin trouvé ?** 1 seul suffit

— **our des traceroutes lancées sur des adresses différentes, doit on tout afficher sur la map ? (ex : 1000 adresses donc énormément d ?arêtes sur la map) ?** Le but est de visualiser le graphe d'Internet donc oui. On peut aussi envisager la possibilité pour l'utilisateur de visualiser uniquement les chemins entre une source et une destination pour plus de clarté.

— **vous recommandez l'utilisation d'une interface graphique ?** comme vous voulez.

— **Vous dîtes : "il y a une arrête si les deux sommets sont voisins, une d'une autre couleur si il y a des étoiles entre les deux sommets". Cela part du fait que nous récupérons toutes les latitudes et longitudes, si nous avons trois point A B C, et que nous avons A qui envoie à B, qui envoie à C, mais que notre service de localisation ne trouve pas les coordonnées de B, que devons-nous faire ? Devons-nous afficher une arrête entre A et C et considérer B comme une étoile ? A et C sont-ils voisins du coup ? A et C sont ils à une distance plus grande que si B n'existait pas ?** Chaque sommet du graphe est une adresse IP. Dans tous les cas, si vous avez trouvé une adresse IP, vous devez la stocker dans le graphe, indépendamment du fait d'avoir trouvé des coordonnées GPS pour l'adresse. Donc A et C ne sont pas voisins dans le graphe stocké. Par contre, pour l'affichage sur le globe géolocalisé, vous pouvez les relier par une arête d'une couleur différente comme pour l'étoile.

— **Ou alors devons-nous stocker lors de la récupération des IPs la distance entre les points et la racine, et pondérer chaque arrête avec une distance qui serait le nombre de point non récupérés entre les deux sommets de l'arrête ?** Il n'est pas assuré par traceroute que X séries d'étoiles soient équivalents à X routeurs inconnus. Cela semble donc pas indispensable de pondérer pour les étoiles.

— **De plus, en exportant et important les fichiers comme écrit dans l'énoncé, nous perdons la liste des voisins si nous ne travaillons pas uniquement avec les points dont nous connaissons latitude et longitude, puisque nous gardons uniquement les sommets reliés (dont nous connaissons la latitude et la longitude) et la distance à la racine. Et dans le cas ou nous travaillons uniquement avec les points dont nous connaissons les coordonnées, lors de l'export puis import, nous perdrions la distance entre deux points si nous pondérions ses actes. Du coup pouvons-nous adapter les fichiers CSV à nos besoins ?** A vous de voir. Faites attention aussi au fait qu'on doit pouvoir gérer deux services de localisation.