Chapitre II

Algorithme de détection des anomalies

II.1 Introduction

Dans leur travail [14], R. Fontugne et al. ont exploité la distribution répandue des sondes Atlas dans le monde afin d'étudier un des problèmes relatifs aux performances des réseaux informatiques.

Il est difficile d'avoir une idée globale sur la topologie de l'Internet. Toutefois, les opérateurs des réseaux informatiques disposent d'un aperçu de l'état des entités qui forment leurs réseaux, les relations entre ces entités ainsi que les éventuels problèmes. Avec la distribution abondante des sondes Atlas dans le monde en termes de type d'adressage : sondes Atlas supportant seulement l'adressage IPv4, d'autres qui supportent en plus l'adressage IPv6, en terme de la diversité géographique, la diversité en termes d'ASs hébergeant les sondes Atlas, etc, il était possible d'aborder les délais dans les réseaux informatiques à travers de nouvelles approches, reposées sur des fondements statistiques. Parmi les points forts de l'analyse menée par R. Fontugne et al., c'était la possibilité de valider les méthodes proposées avec des événements marquants sur Internet.

Le travail de R. Fontugne et al. reprend trois méthodes basées sur les données collectées par les sondes Atlas, chaque méthode reflète l'approche utilisée pour étudier les performances des réseaux informatiques. Ces méthodes sont les suivantes :

1. la détection des changements des délais que subissent les liens intermédiaires dans les traceroutes ;

2. la conception d'un modèle de forwording pour un routeur donné. Ce modèle prédit l'acheminement du trafic afin d'identifier les routeurs et les liens en

panne dans le cas d'un problème de perte de paquets ;

3. la création d'un score par Système Autonome afin d'évaluer l'état de ce dernier.

Dans la suite de ce travail, nous allons reprendre seulement la première méthode. Il s'agit d'étudier le délai d'un lien topologique, c'est le délai entre deux routeurs adjacents sur Internet.

II.2 L'étude des délais des liens

II.2.1 Les données utilisées dans l'analyse des délais

La méthode conçue pour la détection des changements des délais se base sur des fondements statistiques. Ces derniers sont capables de montrer leurs performances si la taille des échantillons [1] considérés est grande. Afin de surveiller un grand nombre de liens sur Internet, il faut avoir un grand nombre de sondes Atlas avec une certaine diversité et qui sont capables de collecter une quantité importante de données relatives aux performances des réseaux informatiques, c'est ce qu'assure le projet RIPE Atlas. Le travail de référence implique principalement les mesures de traceroutes, ainsi deux catégories de mesures sont utilisées :

- *builtin* : ce sont les traceroutes effectués par toutes les sondes Atlas vers les instances des 13 serveurs DNS racines. Les traceroutes sont effectués chaque 30 minutes. En pratique, certains serveurs racines DNS déploient l'anycast. Au moment de la réalisation du travail de référence, c'étaient des traceroutes vers 500 instances des serveurs DNS racines ;

  DNS Anycast est une solution utilisée pour accélérer le fonctionnement des serveurs DNS. Les serveurs DNS adoptant cette approche fournissent des temps de réponse plus courts, et ce partout dans le monde. Les requêtes en provenance de l'utilisateur sont redirigées vers un nœud adéquat suivant un algorithme prédéfini.

- *anchoring* : ce sont les traceroutes effectués par environ 400 sondes Atlas à destination de 189 serveurs [2] et ce chaque 15 minutes.

1. L'échantillon de la métrique qui caractérise un lien : RTT différentiel.
2. Sondes Atlas ayant des fonctionnalités avancées.

En ce qui concerne les traceroutes analysés, le tableau II.1 reprend plus de détails.

	Nombre de traceroutes	**Nombre de sondes**
IPv4	2.8 billion	11,538
IPv6	1.2 billion	4,30

TABLE II.1 – Récapitulatif des traceroutes utilisés dans le travail de référence

L'étude des délais ne concerne pas les adresses privées, ainsi, le suivi des délais ne concerne pas les réseaux privés. De plus, ce suivi se base sur les requêtes de type traceroute, et traceroute reprend une partie de la topologie de l'Internet. En effet, les liens considérés sont ceux topologiques et ne sont pas les liens physiques.

II.3 La description de la détection des délais anormaux des liens

II.3.1 RTT différentiel

Il est indispensable de présenter la définition du RTT (Round Trip Time) différentiel d'un lien avant de procéder à la description de l'algorithme de la détection des anomalies.

Le **temps RTT** est obtenu en calculant la différence entre le timestamp associé à l'envoi du paquet sondé et le timestamp associé à la réception de la réponse ICMP[a]. C'est une métrique pour évaluer les performances d'un réseau en matière de temps de réponse. Les mesures du RTT sont fournies avec l'utilitaire traceroute et ping. En ce qui concerne traceroute, ce dernier fournit les sauts impliqués dans le chemin de forwarding, c'est le chemin parcouru par le trafic entre la source et la destination. Le temps RTT inclut le temps pour atteindre un saut dans le sens du forwarding, à ce temps, il s'ajoute le temps de propagation des réponses. De plus, il y a le temps de traitement des requêtes au niveau des routeurs.

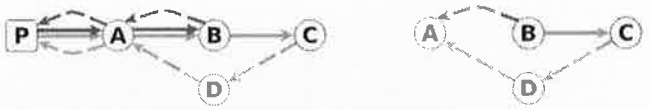
a. Internet Control Message Protocol est un protocole utilisé pour véhiculer des messages de contrôle sur Internet.

La figure II.1 (a) illustre le RTT entre la sonde P et les deux routeurs B et C. Le RTT différentiel entre deux routeurs B et C adjacents, noté Δ_{PBC}, est la

différence entre le RTT entre la sonde P et B (bleu) d'une part, et le RTT entre la sonde P et C (rouge) dans la figure II.1 (b).

$$\begin{aligned}\Delta_{PBC} &= RTT_{PC} - RTT_{PB} \\ &= \delta_{BC} + \delta_{CD} + \delta_{DA} - \delta_{BA} \\ &= \delta_{BC} + \varepsilon_{PBC}\end{aligned}$$

où δ_{BC} est le délai du lien BC et ε_{PBC} est la différence entre les deux chemins de retour (B vers P et C vers P).



(a) Round-trip to router B (blue) and C (red).

(b) Difference of the two round-trips (Δ_{PBC}).

FIGURE II.1

II.3.2 Le principe de la détection des changements des délais

Le suivi du délai d'un lien est déduit du suivi de l'évolution de son RTT différentiel. Reprenons la formule du RTT différentiel du lien BC : $\delta_{BC} + \varepsilon_{PBC}$. Supposons qu'on dispose d'un nombre n de sondes Atlas P_i, $i \in [1, n]$, telles que toutes les sondes ont un chemin de retour différent depuis B et depuis C. En effet, les RTTs différentiel pour chacune des sondes Atlas Δ_{P_iBC} partagent la même composante δ_{BC}, toutefois, ces RTTs ont des valeurs des ε_{P_iBC} indépendantes. L'indépendance de ces valeurs implique que la distribution Δ_{P_iBC} est estimé d'être stable au cours du temps si δ_{BC} est constant. Cependant, un changement significatif de la valeur du δ_{BC} influence les valeurs des RTTs différentiel, dans ce cas, la distribution des RTTs différentiel changes si δ_{BC} change. Enfin, les changements des délais sont déduits des changements des RTTs différentiel qu'on peut les quantifier.

La détection des anomalies en délais repose sur un théorème très important en statistiques, c'est le théorème central limite (TCL). Ce théorème annonce que si on a une suite de variables aléatoires indépendantes ayant la même espérance et la même variance, la moyenne de ces variables aléatoires est une variable aléatoire qui suit une loi normale. De manière générale, le théorème central limite explique la distribution des moyennes des échantillons. Ce théorème peut être appliqué

aux différents lois. Par exemple la loi normale [3], binomiale, etc.

II.3.3 Les résultats de l'analyse des délais des liens

Nous distinguons deux sortes de résultat à l'issue de l'analyse des changements des délais. Premièrement, ce sont les changements identifiés tout au long de la durée de l'analyse, chaque changement est caractérisé par un ensemble de détails. Pour le deuxième résultat, ce sont des graphiques reprenant les changements ainsi que les changements jugés anormaux, ce qu'on va appeler par la suite par *anomalies*, précisément ces graphiques présentent l'évolution du RTT différentiel d'un lien au cours du temps.

II.3.4 Présentation de l'algorithme de la détection des changements anormaux : Résultat I

Description de l'algorithme de détection

Les étapes principales de la détection des changements des délais sont résumées dans l'algorithme 1. Nous allons détailler chaque étape : ses objectifs, ses entrées et ses sorties. En ce qui concerne l'entrée du programme principal, ce sont les paramètres présentés dans la section II.3.4.

Algorithm 1 Les étapes de la procédure detectRttChangesMongo()

```
1 : procedure DETECTRTTCHANGESMONGO(expId)
2 :     for all currDate ∈ dates do
3 :         computeRtt()
4 :         mergeRttResults()
5 :         outlierDetection()
6 :     end for
7 : end procedure
```

Description des paramètres de l'analyse des délais

La détection des changements des délais nécessite l'ajustement d'un nombre de paramètres. La valeur de chaque paramètre est relative au fondement utilisé théorique ou bien empirique, qui a été justifié par les auteurs du travail de référence. Ci-dessous les paramètres à ajuster avant de lancer une analyse. Chaque paramètre sera défini dans son contexte.

3. Un exemple illustratif dans A.

start : c'est la date de début de l'analyse. Ce sont les traceroutes capturés par les sondes Atlas à partir de cette date qui seront analysés.

end : c'est la date marquant la fin de l'analyse. Comme le paramètre *start*, c'est la date des derniers traceroutes capturés par les sondes Atlas à considérer dans la présente analyse.

timeWindow : ce paramètre est exprimé en seconde. La durée de l'analyse, qui est le temps écoulé entre *start* et *end*, est divisée sur des périodes de même taille : *timeWindow*. Pour chacune de ces périodes, on caractérise les liens identifiés sur les traceroutes capturés en cette période. La figure II.2 reprend le contexte des trois paramètres *start*, *end* et *timeWindow* avec les étapes principales.

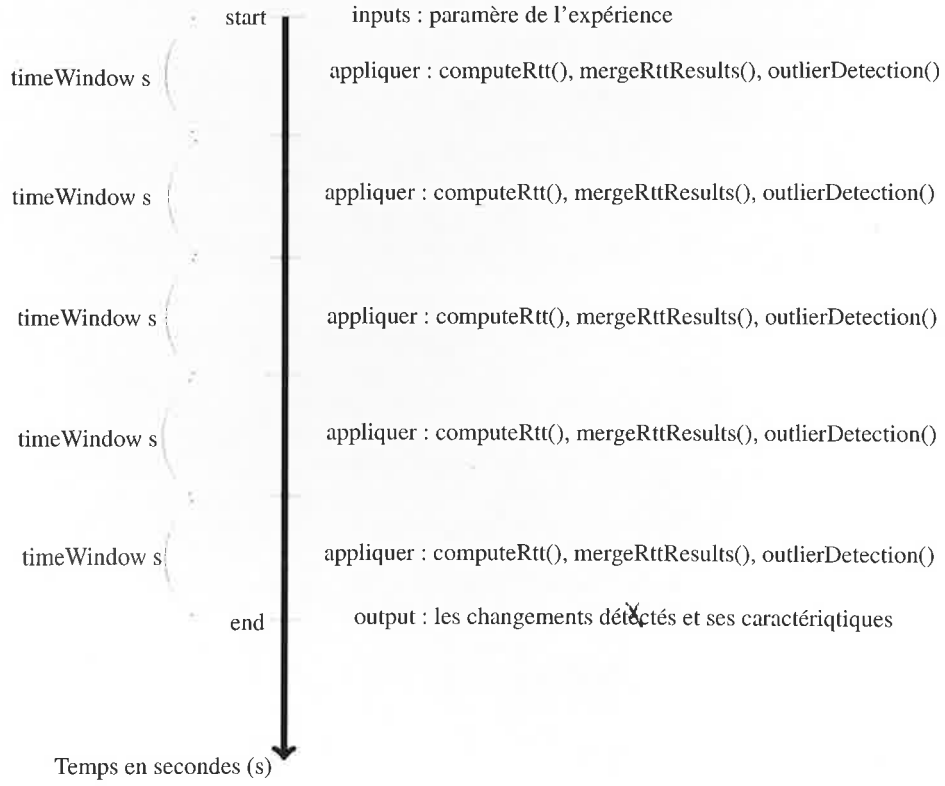


FIGURE II.2 – Illustration du paramètre timeWindow

alpha : c'est le paramètre du lissage exponentiel.

« *Les méthodes de lissage exponentiel sont un ensemble de techniques empiriques de prévision qui accordent plus ou moins d'importance aux valeurs du passé d'une série temporelle.* [4] »

Pour calculer la prochaine valeur de la médiane des RTTs différentiel de référence $\overline{m}_t$ du timeWindow courant t , soit :

$m_t = \Delta^{(m)}$ la médiane des RTTs différentiel observée pour un lien durant le timeWindow t.

$\overline{m}_{t-1} = \overline{\Delta}^{(m)}$ est la médiane des RTTs différentiel de référence durant le timeWindow $t-1$, la prochaine valeur de la médiane de référence $\overline{m}_t$ est obtenue par :

$$\overline{m}_t = \alpha\, m_t + (1\text{-}\,\alpha)\, \overline{m}_{t-1}$$

Le paramètre alpha α, tel que $\alpha \in (0, 1)$, est le seul paramètre à définir dans le calcul de $\overline{m}_t$. Ce paramètre contrôle l'importance des mesures précédentes par rapport aux mesures récentes.

«*Plus α est proche de 1 plus les observations récentes influent sur la prévision, à l'inverse un α proche de 0 conduit à une prévision très stable prenant en compte un passé lointain.* [5]» Dans la présente étude, le paramètre α est préféré d'être petit.

minASN : les paramètres *minASN*, *minASNEntropy* ainsi que *minSeen* dont les deux derniers seront détaillés dans la suite sont associés à l'aspect diversité des sondes Atlas pour assurer une exactitude des résultats obtenus.

L'analyse des RTTs différentiel est appliquée seulement sous certaines conditions. Ainsi, la détection des anomalies dans les délais d'un lien est valide si les éléments suivants sont vrais. (1) Le lien est surveillé par plusieurs sondes et que le chemin de retours vers ces sondes soit différent à chaque fois. (2) Les paquets ayant passés par le lien XY, doivent aussi passer par le lien XY en leur retour, mais dans le sens opposé.

Les valeurs des RTTs ambiguës sont filtrées en éliminant les liens surveillés seulement par les sondes appartenant au même Système Autonome, car généralement le chemin de retour est similaire pour ces sondes suite à leur présence au sein du même Système Autonome (généralement même politique de routage). Seuls les liens surveillés par au moins 3 Systèmes Autonomes qui sont conservés, la valeur de 3 pour le paramètre *minASN* est choisie de manière empirique. Sachant qu'une valeur plus petite que 3 peut affecter l'exactitude des résultats.

4. Source : https://perso.math.univ-toulouse.fr/lagnouz/files/2013/12/Chap6.pdf, consultée le 30/09/2018.

5. Source : https://www.math.u-psud.fr/~goude/Materials/time_series/cours3_lissage_expo.pdf, consultée le 30/09/2018.

minASNEntropy : Il important qu'un lien soit identifié par au moins *minASN* Systèmes Autonomes, toutefois, le nombre de sondes Atlas ayant identifié ce lien doit être équilibré entre ces Systèmes Autonomes. En ce qui concerne l'équilibre du nombre de sondes ayant surveillé un lien par AS, il est mesuré par une entropie normalisée. Soit $A = \{a_i \mid i \in [1, n]\}$, a_i est le nombre de sondes pour chaque AS parmi les n ASs surveillant un lien donné. L'entropie $H(A)$ est défini avec :

$$H(A) = -\frac{1}{\ln} \sum_{i=1}^{n} P(a_i) \ln P(a_i)$$

Le nombre de sondes par Système Autonome sera équilibré jusqu'à l'atteinte de l'entropie minimale donnée par *minASNEntropy*. L'idée est d'éliminer des sondes Atlas, de manière aléatoire, de l'AS qui a un grand nombre de sondes Atlas jusqu'à avoir l'équilibre mesuré par l'entropie. Dans la présente analyse, les liens ayant une entropie > 0.5 sont conservés, cependant, si l'entropie d'un lien est < 0.5, chercher une sonde, de manière aléatoire, qui se trouve dans l'AS i tel que $a_i = max(A)$, ensuite recalculer l'entropie avec la sonde. L'opération de l'élimination est répétée jusqu'à avoir une entropie > 0.5.

L'entropie

> ***L'entropie*** *est une grandeur d'état extensive qui caractérise l'état de désordre du système.*[a] De faible valeurs d'entropie,$H(A) \backsimeq 0$, indiquent que la majorité des sondes sont concentrées dans un seul AS, et les grandes valeurs d'entropie, $H(A) \backsimeq 1$, indiquent que les sondes sont réparties équitablement sur les ASs.

a. Source : http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/chimie/01/03-Reaction_chimique/co/module_03-Reaction_chimique_26.html, consultée le 30/09/2018.

minSeen : comme l'analyse est faite sur plusieurs périodes : timeWindow, le paramètre *minSeen* indique le nombre de fois où un lien a été identifié. Par exemple, un lien peut être identifié dans 3 timeWindow, ou bien être identifié en une seule fois durant toute la période de l'analyse.

af : ce paramètre indique l'étendue de l'analyse des délais en matière de type d'adressage. Telle qu'une analyse peut concentrer sur les liens en IPv4 ou bien en IPv6. Pour précision, il est pris en compte le type d'adressage IP lors du stockage

des traceroutes dans MongoDB [6]. Ce que permet de choisir les traceroutes suivant ce paramètre [7].

comment : un commentaire est utile pour donner plus d'informations sur une analyse en particulier. Les commentaires sont à titre informatifs et n'affectent pas l'analyse.

prefixes : les liens analysés sont finalement les liens entre deux routeurs adjacents dans la topologie. Avec l'expression régulière fournie à travers le paramètre *prefixes*, il est possible de limiter l'analyse sur les liens où les routeurs appartiennent aux blocs d'adresses définis par *prefixes*.

experimentDate : le paramètre *experimentDate* indique la date de lancement de l'expérience. Les auteurs enregistrent les expériences dans une collection dans la base de données MongoDB. Ce que permet de faciliter la comparaison entre les différentes expériences.

confInterval :

Afin de calculer l'incertitude associée à un ensemble de résultats, il faut répéter les mesures. Chaque mesure sur un échantillon peut donner des résultats différents. Ainsi, en se basant sur la déviation sur les résultats, il est possible de calculer l'incertitude de la "moyenne" calculée de ces résultats. Cette incertitude permet de donner une indication sur les données. Par exemple, est-ce que la moyenne calculée N représente la valeur réelle avec une incertitude de plus ou moins m ?

6. C'est une base de données NoSQL utilisée dans le stockage des données dans le travail de référence.

7. Dans MongoDB, la collection traceroute_2017_05_15 reprend les traceroutes de la date 15/05/2017 en IPv4, et la collection traceroute6_2017_05_15 pour la même date mais en IPv6.

En statistiques, le *binomial proportion confidence interval* est l'intervalle de confiance pour la probabilité de succès calculée à partir des séries d'expériences de succès-échec. C'est un intervalle qui estime la probabilité de succès p si seulement le nombre d'expériences n et celles réussites n_s sont connus.
Il existe plusieurs formules pour calculer l'intervalle de confiance binomial. Toutefois, elles se basent toutes sur une distribution binomiale. Une distribution binomial s'applique si une expérience est répétée un nombre fixe de fois, chaque tentative a deux possibilités : succès ou échec. La probabilité est la même à chaque tentative et les tentatives sont statistiquement indépendantes. La distribution binomiale est une distribution de probabilité discrète, il est difficile de calculer pour un grand nombre de tentatives, il existe une variété d'approximations pour le calcul de l'intervalle de confiance.

Les intervalles de confiance sont formulés par un calcul binomial avec distribution - free. Ce calcul est approché par le score de Wilson. C'est une méthode pour calculer l'intervalle de confiance. Le score de Wilson fournit deux valeurs dans l'intervalle [0, 1].

Intervalle de confiance d'une moyenne

L'intervalle de confiance (IC) est défini comme représentant les valeurs probables que peut prendre une moyenne, si l'on accepte une marge d'erreur définie à l'avance (e.g 5%). Il existe plusieurs méthodes pour calculer l'intervalle de confiance d'une proportion. Parmi les critères impliqués sur le choix de la méthode de calcul, il y a N, le nombre total d'essais (nombre des expériences).

Dans le travail de référence, les auteurs ont utilisé la méthode du score de wilson pour calculer l'intervalle de confiance de la médiane [8]. Le score de wilson a montré ses performances même dans le cas où le nombre total d'essais est petit. Par exemple, il se peut qu'un lien soit caractérisé par seulement 3 RTTs différentiel durant un timeWindow. [9]

Chaque valeur de médiane a son intervalle de confiance. On compare le chevauchement entre l'intervalle de confiance de la médiane de référence avec l'intervalle de confiance de la valeur de médiane calculée en cours, d'un lien donné. Afin d'évaluer si la différence entre ces deux intervalles est significative statistiquement. En particulier une différence de $1\ ms$ est non significative. On distingue trois cas comme illustré par la figure II.3 et la formule II.1.

8. Le choix d'utilisation de la médiane à la place de la moyenne a été justifié dans le travail de référence.

9. Source : http://npsycog.over-blog.com/article-3274585.html, consultée le 12/10/2018.

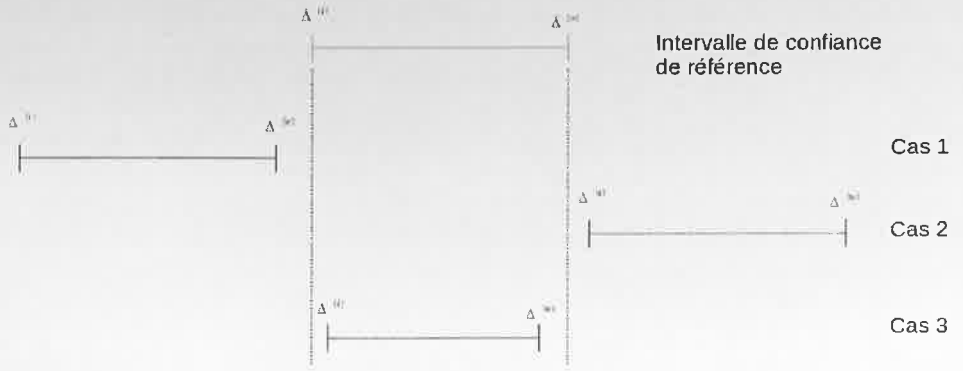


FIGURE II.3 – La comparaison entre les intervalles de confiance de la médiane de référence et de la médiane calculée.

Le cas 1 et 2 dans la figure II.3 illustrent un changement anormal dans le délai du lien en question. Toutefois, le cas 3 où l'intervalle de confiance courant est inclus dans l'intervalle de référence, est le cas normal, autrement dit, le délai de ce lien est normal. La différence entre les deux intervalles intervalles de confiance est quantifiée par la déviation d définie par la formule II.1.

$$d = \begin{cases} \frac{\Delta^{(l)} - \overline{\Delta}^{(u)}}{\overline{\Delta}^{(u)} - \overline{\Delta}^{(m)}}, & \text{if } \overline{\Delta}^{(u)} < \Delta^{(l)}. \\ \frac{\overline{\Delta}^{(l)} - \Delta^{(u)}}{\overline{\Delta}^{(u)} - \overline{\Delta}^{(m)}}, & \text{if } \overline{\Delta}^{(m)} < \Delta^{(l)}. \\ 0, & \text{otherwise.} \end{cases} \tag{II.1}$$

L'intervalle de confiance est calculé, dans la présente implémentation, avec la fonction *sm.stats.proportion_confint* [10].

```
statsmodels.stats.proportion.proportion_confint(count, nobs, alpha=0.05, method='normal')
```

proportion_confint est une implémentation, en python, pour le calcul du score de Wilson. Elle prend les paramètres suivants :

- count : c'est le nombre de fois où une expérience réussit.
- nobs : c'est le nombre total d'expériences.
- alpha : c'est le paramètre de risque. Généralement il prend les valeurs suivantes 5%, 1% ou 0, 1%. C'est un des paramètres de l'expérience.
- method : on distingue plusieurs méthodes, celle utilisée est *wilson*.

10. Source : https://www.statsmodels.org/dev/generated/statsmodels.stats.proportion.proportion_confint.html, consultée le 07/10/2018.

En retour, la fonction *proportion_confint* fournie *ci_low* et *ci_upp*. Ces deux grandeurs sont utilisés pour construire l'intervalle de confiance.

Exemple des paramètres de l'analyse des délais

```
expParam = {
        "timeWindow": 60*60, # in seconds
        "start": datetime(2018, 1, 1, 0, 0, tzinfo=timezone("UTC")),
        "end": datetime(2018, 1, 1, 23, 0, tzinfo=timezone("UTC")),
        "alpha": 0.01,
        "confInterval": 0.05,
        "minASN": 3,
        "minASNEntropy": 0.5,
        "minSeen": 3,
        "experimentDate": datetime.now(),
        "af": "",
        "comment": "some comment",
        "prefixes": None
}
```

Les étapes de l'analyse des délais

Les sections suivantes présentent chaque étape brièvement des étapes présentées dans l'algorithme 1, c'est pour l'illustration, car des détails peuvent manquer comme les paramètres des fonctions.

computeRtt :
Objectif : identification de tout lien dans les traceroutes analysés.
Entrées : af, start, end, prefixes.
Sorties : la caractérisation des liens en calculant leur RTT différentiel.
Pseudo-code :

Algorithm 2 caractérisation des liens

```
1 : function COMPUTERTT(af, start, end, prefixes)
2 :     diffRtt ← {}
3 :     nbRow ← 0
4 :     traceroutes ← findTraceroutes(af, start, end, prefixes)
5 :     for all trace ∈ traceroutes do
6 :         readOneTraceroute( trace )
7 :     end for
        return diffRtt, nbRow
8 : end function
```

Description :

1. Rechercher les traceroutes, dans la base de données, capturés entre la date *start* et *end*, parmi les traceroutes ayant *af* comme adressage. Si *prefixes* est fixé, seuls les traceroutes ayant impliqué des routeurs appartenant au bloc d'adresses défini par *prefixes* qui seront conservés.

2. Analyser chaque traceroute, *trace*, à travers les étapes suivantes :

   (a) élimination du traceroute échoué en vérifiant la présence des attributs "error" ou "err" dans les résultats de ce traceroute ;

   (b) si le traceroute n'a pas été éliminé durant l'étape précédente, on passe à l'évaluation de chaque saut. Un saut est éliminé si l'adresse IP est absente pour ce saut, s'il s'agit d'une adresse privée, si l'information sur le RTT est absente et enfin si le RTT a une valeur négative ;

   (c) caractérisation des liens en notons pour chaque couple d'adresses IP la distribution des RTTs, les sondes Atlas et les identifiants des mesures.

Soit un exemple illustratif des opérations décrites ci-dessus, la figure II.4 reprend les détails. Tel que :

- Lien : lien à suivre, défini par deux adresses IP.
- RTT différentiel : le RTT différentiel calculé du lien en question.
- Probe : l'ensemble de sondes ayant surveillé le lien.
- msmId : l'ensemble de mesures ayant surveillé le lien. Comme un identifiant de mesure définit la requête à lancer par toutes les sondes Atlas, ainsi, un lien peut être surveillé dans le cadre d'une mesure et avec une ou plusieurs sondes Atlas.

Pour le lien $('160.242.100.88','196.216.48.144')$, le calcul du RTT différentiel est décrit ci-dessous :

$$
\begin{aligned}
RTT(160.242.100.88) &= 4.263; 6.082; 11.834 \\
mediane(4.263; 6.082; 11.834) &= 6.082 \\
RTT(160.242.100.88) &= 3.678; 15.568; 3.655 \\
mediane(3.678; 15.568; 3.655) &= 3.678 \\
RTTDiff('160.242.100.88','196.216.48.144') &= 6.082 - 3.678 = 2,404
\end{aligned}
$$

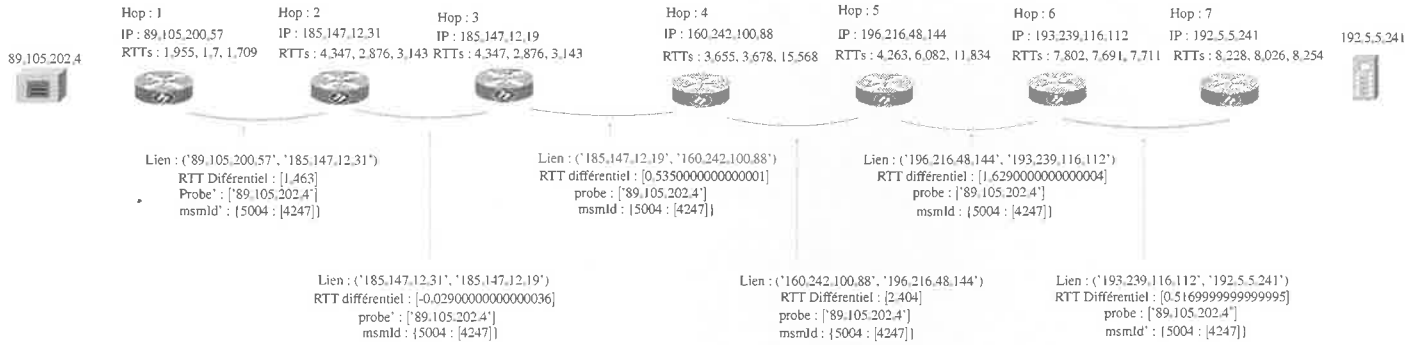


FIGURE II.4 – Caractérisation des liens dans un traceroute

mergeRttResults

Durant l'analyse des traceroutes identifiés dans un timeWindow, il se peut qu'un lien soit identifié dans plusieurs traceroutes. La méthode *mergeRttResults* permet de fusionner les détails d'un lien donné, ce sont des détails obtenus avec plus d'un traceroute. Prenons le lien ('160.242.100.88','196.216.48.144'), ce dernier a été identifié dans trois traceroutes comme suit :

(IP1,IP2)	RTT	Probes	MSM
('160.242.100.88', '196.216.48.144')	[63.4740000000000004]	[u'196.216.164.50']	5004 : set([14465])
('196.49.6.10', '192.5.5.241')	[3.4729999999999848]	['196.216.164.50']	5004 : set([14465])
('196.216.164.1', '196.12.10.246')	[-1.379]	['196.216.164.50']	5004 : set([14465])
('196.12.10.246', '160.242.100.88')	[0.21799999999999997]	['196.216.164.50']	5004 : set([14465])

TABLE II.2 – Exemple de traceroute : 1

(IP1,IP2)	RTT	Probes	MSM
('185.147.12.31', '185.147.12.19')	[-0.029000000000000036]	[u'89.105.202.4']	5004 : set([4247])
('185.147.12.19', '160.242.100.88')	[0.53500000000000001]	['89.105.202.4']	5004 : set([4247])
('89.105.200.57', '185.147.12.31')	[1.463]	['89.105.202.4']	5004 : set([4247])
('196.216.48.144', '193.239.116.112')	[1.6290000000000004]	['89.105.202.4']	5004 : set([4247])
('193.239.116.112', '192.5.5.241')	[0.51699999999999995]	['89.105.202.4']	5004 : set([4247])
('160.242.100.88', '196.216.48.144')	[2.404]	['89.105.202.4']	5004 : set([4247])

TABLE II.3 – Exemple de traceroute : 2

(IP1,IP2)	RTT	Probes	MSM
('199.189.117.17', '199.189.116.229')	[-0.024000000000000091]	['134.197.113.7']	5004 : set([6201])
('160.242.100.88', '207.197.17.101')	[19.261999999999997]	['134.197.113.7']	5004 : set([6201])
('199.189.116.229', '63.158.61.169')	[2.34]	[u'134.197.113.7']	5004 : set([6201])
('205.171.1.18', '192.5.5.241')	[-0.184000000000000105]	['134.197.113.7']	5004 : set([6201])
('63.158.61.169', '205.171.234.102')	[-2.0219999999999985]	['134.197.113.7']	5004 : set([6201])
('134.197.113.14', '196.216.48.144')	[-0.53699999999999999]	['134.197.113.7']	5004 : set([6201])
('207.197.17.101', '199.189.117.17')	[2.1290000000000013]	['134.197.113.7']	5004 : set([6201])
('205.171.234.102', '205.171.1.18')	[-0.170999999999999937]	['134.197.113.7']	5004 : set([6201])
('196.216.48.144', '160.242.100.88')	[0.065]	[u'134.197.113.7']	5004 : set([6201])

TABLE II.4 – Exemple de traceroute : 3

Afin d'illustrer l'opération de fusion, on reprend seulement les détails du lien qui a subi la fusion en se référant aux résultats intermédiaires présentés dans les tableaux II.2, II.3 et II.4. En effet, on obtient :

(IP1,IP2)	RTT	Probes	MSM
('160.242.100.88', '196.216.48.144') :	[0.065, 2.404, 63.4740000000000004]	['134.197.113.7', '89.105.202.4', '196.216.164.50']	5004 : set([6201, 4247, 14465])

Ce qu'on peut apprendre à travers cette fusion, le délai d'un lien ne dépend pas de l'ordre des routeurs.

outlierDetection

Dans le présent contexte, la détection des *outliers* dénote la détection des changements anormaux. Cette opération implique plusieurs notions et principes. Les étapes de la détection des outliers sont illustrées dans l'algorithme 3.

Objectif : trouver les changements anormaux des délais après la caractérisation des liens.

Entrées : diffRTT, le résultat de l'étape *mergeRttResults*, la liste des éléments par lien comme :

```
1
2   (u'160.242.100.88', u'106.216.48.144'): {'rtt': [0.065, 2.404, 63.474000000000004],
3    'probe': [u'134.197.113.7', u'89.105.202.4', u'196.216.164.50'],
4    'msmId': defaultdict(<type 'set'>, {5004: set([6201, 4247, 14465])})}
5   }
```

Sorties : la liste des alarmes.

Pseudo-code :

Algorithm 3 La détection des changements anormaux des liens

```
1 : function OUTLIERDETECTION(diffRTT, smoothMean)
2 :     smoothMean ← {}                          ▷ La médiane de référence d'un lien
3 :     alarms ← []                              ▷ La liste des anomalies détectées
4 :     alpha ← float(param["alpha"])
5 :     minAsn ← param["minASN"]
6 :     minASNEntropy ← param["minASNEntropy"]
7 :     confInterval ← param["confInterval"]
8 :     minSeen ← param["minSeen"]
9 :     for all ipPair ∈ sampleDistributions do
10 :        dist ← la distribution des RTTs différentiel précédemment calculée.
11 :        probes ← les sondes ayant surveillé le lien ipPair
12 :        asn ← probe2asn(probes)
13 :        asnEntropy ← computeAsnEntropy()
14 :        while asnEntropy < minASNEntropy AND len(asn) > minAsn do
15 :            trimDistribution()
16 :        end while
17 :        findAlarms(smoothMean)
18 :    end for
        return diffRtt, nbRow
19 : end function
```

Description :

– probe2asn(probes) : cette étape permet de faire l'association entre l'adresse IP de la sonde Atlas et son ASN. Cette association permet d'avoir une idée

sur le nombre de sondes Atlas ayant surveillé le lien par Système Autonome. Ce que explique la section II.3.4.

- trimDistribution() : l'objectif de cette étape est d'équilibrer le nombre de sondes Atlas par AS ayant surveillé un lien en vue d'éviter les résultats biaisées.
- computeAsnEntropy() : calcul de l'entropie d'une distribution pour des valeurs de probabilité données. *stats.entropy* est l'implémentation, en python, du calcul de l'entropie. $stats.entropy(asn.values())/np.log(len(asn))$.
- findAlarms() : dans cette étape, on discute la procédure d'identification des anomalies, autrement dit, les alarmes. Pour une raison de clarté et de lisibilité, cette étape est détaillée avec l'algorithme 4.

Algorithm 4 caractérisation des liens

```
1 : nbProbes ← len(probes)
2 : n ← len(dist)
3 : med ← np.median(dist)
4 : wilsonCi ← sm.stats.proportion_confint(len(dist)/2, len(dist), confInterval,
    "wilson")
5 : currLow ← dist[int(wilsonCi[0])]
6 : currHi ← dist[int(wilsonCi[1])]
7 : reported ← False
8 : if ipPair in smoothMean then ▷ mise à jour de la référence d'un lien identifié
9 :     if ref["nbSeen"] >= minSeen then
10 :        if ref["high"] < currLow or ref["low"] > currHi then
11 :            if med < ref["mean"] then
12 :                update diff, diffMed, deviation, devBound
13 :            else
14 :                update diff, diffMed, deviation, devBound
15 :            end if
16 :            alarm ← describeAlarm()
17 :            reported ← True
18 :        end if
19 :    end if
20 :    updateReference()
21 : else                                   ▷ nouveau lien donc nouvelle référence
22 :    if minSeen > 1 then
23 :        smoothMean[ipPair] ← updateReference()
24 :    else
25 :        smoothMean[ipPair] ← updateReference()
26 :    end if
27 : end if
```

Les deux valeurs de l'intervalle de confiance calculé avec le score de Wilson sont illustrées dans les deux lignes 6 et 5 de l'algorithme 4.

Le principe de la détection des anomalies est repris dans la section II.3.5, c'est le même principe de la détection, toutefois les étapes sont adaptées pour générer les résultats.

II.3.5 Présentation de l'algorithme de la détection des changements anormaux : Résultat II

Introduction

Cette analyse permet le suivi des RTTs différentiel d'un lien donné, entre la date *start* et la date *end*. Ce suivi se base sur les dernières valeurs du RTT différentiel. Rappelons que chaque lien est caractérisé par son RTT différentiel qui représente la médiane de tous les RTTs différentiel enregistrés pendant une durée d. Cette médiane a un intervalle de confiance. Afin de pouvoir répondre à la question suivante : à 95 %, la médiane de référence calculée, *smoothAvg*, ayant comme intervalle de confiance *smoothHi* et *smoothLow* représente l'estimation réelle du RTT différentiel du lien en question.

L'idée de base de l'outil de détection est d'analyser les données, qui sont des traceroutes, d'une période D en la décomposant en de petites périodes :

$d_1, d_2, ..., d_i, ..., d_j, ..., d_n$, tel que *timeWindow* = d_{j+1} - d_j, *timeWindow* est exprimé en secondes, d_1 représente *start* et d_n représente *end*.

Caractéristiques d'un lien

A l'issue d'une préparation préalable des traceroutes, chaque lien est caractérisé par les dates pendant lesquelles il était identifié ainsi que les RTTs différentiel relatives à ces dates. C'est ce que illustre *rttDiff*.

rttDiff = $([d_1, d_2, ..., d_i, d_j, ..., d_n], [r_1, r_2, ..., r_n])$, $i, j \in [0, n]$, n est entier.

où d_i est la date marquant le début d'un timeWindow et r_i est le RTT différentiel enregistré durant d_i, n est le nombre de fois où e a été identifié. $\exists i, j$ tel que $d_i = d_j$, c'est le cas où un lien a été identifié plusieurs fois durant un timeWindow, et comme on présente l'évolution par timeWindow, les RTTs différentiel seront combinés par la suite en calculant la médiane par ce timeWindow.

```
rttDiff[e] = (dates , rttDiffs)
dates : [datetime.datetime(2018, 1, 2, 1, 5), datetime.datetime(2018, 1, 2, 1,5), datetime.datetime(2018,
    1, 2, 2, 5)]
rttDiffs : [1.463, 4.394, 358.394]
```

On caractérise un lien e durant la période d_j avec les éléments suivants :

- *dest[]* : c'est la distribution des RTTs différentiel du lien e durant d_i. Une distribution de moins de 3 RTTs différentiel pour d_i est à ne pas considérer.
- *smoothAvg[]* : c'est l'estimation de la médiane de référence.
- *smoothHi[]* : c'est l'estimation de la médiane minimale de référence. C'est la borne supérieure de l'intervalle de confiance de référence.
- *smoothLow[]* : c'est l'estimation de la médiane maximale de référence. C'est la borne inférieure de l'intervalle de confiance de référence
- *alarmsDates[]* : les dates pendant lesquelles une anomalie a été détectée.
- *alarmsValues[]* : les médianes des RTTs différentiel considérées comme anormales.

- *median[]* : les médianes des RTTs différentiel calculées caractérisant le lien tout au long de la période de l'analyse, une médiane par d_j.
- *mean[]* : la moyenne des RTTs différentiel. L'utilisation de la moyenne a pour but la comparaison des performances de cette dernière avec celles de la médiane. Alors que l'outil de détection repose principalement sur la médiane.
- *ciLow[]* : la borne inférieure de l'intervalle de confiance de la médiane calculée.
- *ciHigh[]* : la borne supérieure de l'intervalle de confiance de la médiane calculée.
- *dates[]* : les dates à présenter dans le graphique. Certaines dates ne sont pas présentables si le nombre des RTTs différentiel est petit durant ces dates.

La procédure de création de l'évolution des RTTs différentiel d'un lien

Objectif : Analyser l'évolution du RTT différentiel d'un lien donné, noté e.

Entrées : les détails du lien, c'est une instance du rttDiff.

Sorties : l'évolution des RTTs différentiel du lien avec la présentation des anomalies si elles étaient identifiées.

Etapes : d'abord on distingue deux étapes principales, la première a pour objectif la préparation de données. Ainsi, pour chaque lien trouvé, on le caractérise avec leurs RTTs différentiels ainsi que la date pendant laquelle un RTT différentiel a été enregistré. Deuxièmement, en partant des résultats de la première étape, on génère l'évolution du RTT d'un lien. La deuxième étape est décrite avec l'algorithme 5 dont on peut noter les étapes de cette dernière :

1. Préparation des données du timeWindow, entre 2 et 14.
   - calcul du score de wilson
   - calcul de la médiane et de son intervalle de confiance.
2. Mise à jour des données du lien en calculant la médiane de référence et l'intervalle de confiance de la médiane de référence :

- Tant que le nombre des médianes est inférieure strictement à 24, on met à jour les distributions smoothAvg, smoothHi et smoothLow.
- Si le nombre de médianes est égale à 24, on construit une sorte de médiane de référence (25 ème), c'est la médiane des médianes précédemment notées pour ce lien, ensuite les 24 premières médianes seront mises à jour avec cette médiane de référence.
- A partir d'un nombre de médianes plus grand que 24, la nouvelle médiane est prédite à travers la formule du lissage exponentiel.

3. Détection des anomalies en comparant les deux intervalles de confiance : le courant avec celui de référence.
4. Génération de l'évolution : après avoir calculé pour chaque timeWindow de la période l'analyse :
   - la médiane des RTTs différentiel de référence ;
   - l'intervalle de confiance de la médiane de référence ;
   - la médiane des RTTs différentiel estimée ;
   - l'intervalle de confiance de la médiane estimée.

   La comparaison des deux intervalles de confiance permet d'affirmer si la médiane estimée est une anomalie.

Précisions relatives au pseudo-code :

- a.append(b) : ajouter b à la fin de la série de valeurs a.
- median(a) : calculer la médiane de la série de valeurs a.
- a.sort() : ordonner les valeurs de la série de valeurs a, par défaut, par ordre croissant.
- size(a) : donner la taille de la série de valeurs a.
- proportion_confint() : calcul de la borne inférieure et supérieur du score de Wilson.
- *dateranges* : la succession des dates marquant la durée de l'analyse.

Pseudo Code :

Algorithm 5

```
1 : for all d in dateranges do
2 :     indices ⇐ rttDiff[1]==d        ▷ Trouver les indices des RTTs différentiel identifiés durant d
3 :     dist ⇐ rttDiffs[indices]    ▷ Récupérer les RTTs différentiel aux indices
4 :     if size(dist) < 3 then
5 :         passer à la date suivante dans dateranges
6 :     end if
7 :     dates.append(d)
8 :     median.append(median(dist))                ▷ mettre à jour la médiane
9 :     mean.append(mean(dist))
10 :    dist.sort()
11 :    wilsonCiLow, wilsonCiHi Leftarrow proportion_confint()  ▷ Calcul de score de Wilson
12 :    wilsonCi ⇐ np.array(wilsonCi)*len(dist)
13 :    currLow.append(median[-1] - dist[int(wilsonCi[0])])    ▷ mettre à jour la médiane minimale
14 :    ciHigh.append( dist[int(wilsonCi[1])] - median[-1] )    ▷ mettre à jour la médiane maximale
15 :    if len(smoothAvg) < 24 then
16 :        smoothAvg.append(median[-1])
17 :        smoothHi.append(dist[int(wilsonCi[1])])
18 :        smoothLow.append(dist[int(wilsonCi[0])])
19 :    else if len(smoothAvg) == 24 then
20 :        smoothAvg.append(np.median(smoothAvg))
21 :        smoothHi.append(np.median(smoothHi))
22 :        smoothLow.append(np.median(smoothLow))
23 :        for i in [0,24] do
24 :            smoothAvg[i] = smoothAvg[-1]
25 :            smoothHi[i] = smoothHi[-1]
26 :            smoothLow[i] = smoothLow[-1]
27 :        end for
28 :    else        ▷ Utilisation du lissage exponentiel pour prédire smoothAvg, smoothHi et smoothLow
29 :        smoothAvg.append(0.99*smoothAvg[-1]+0.01*median[-1])
30 :        smoothHi.append(0.99*smoothHi[-1]+0.01*dist[int(wilsonCi[1])])
31 :        smoothLow.append(0.99*smoothLow[-1]+0.01*dist[int(wilsonCi[0])])
            ▷ La détection des anomalies se déclenche dés avoir un échantillon de taille 25
32 :        if (median[-1]-ciLow[-1] > smoothHi[-1] OR median[-1]+ciHigh[-1] < smoothLow[-1]) AND np.abs(median[-1]-smoothAvg[-1])>1 then
33 :            alarmsDates.append(d)
34 :            alarmsValues.append(median[-1])
35 :        end if
36 :    end if
37 :
38 : end for
```

II.3.6 Quelques chiffres sur la médiane des RTTs différentiel

L'adaptation du théorème centrale limite pour l'utilisation de la médiane au lien de la moyenne peut engendrer des contraintes en matière de performances si la distribution pour laquelle on souhaite calculer la médiane est grande.

La figure II.5 présente les trente premières distributions des médianes. L'axe des abscisses représente les liens et l'axe des ordonnées représente la taille de la distribution des médianes des RTT différentiel d'un lien, ce sont les RTTs différentiel caractérisant le lien pendant lors un timeWindow parmi les timeWindows entre *start* et *end*. Pour une raison de clarté, le reste des distributions est présenté dans le document disponible sur GitHub [11].

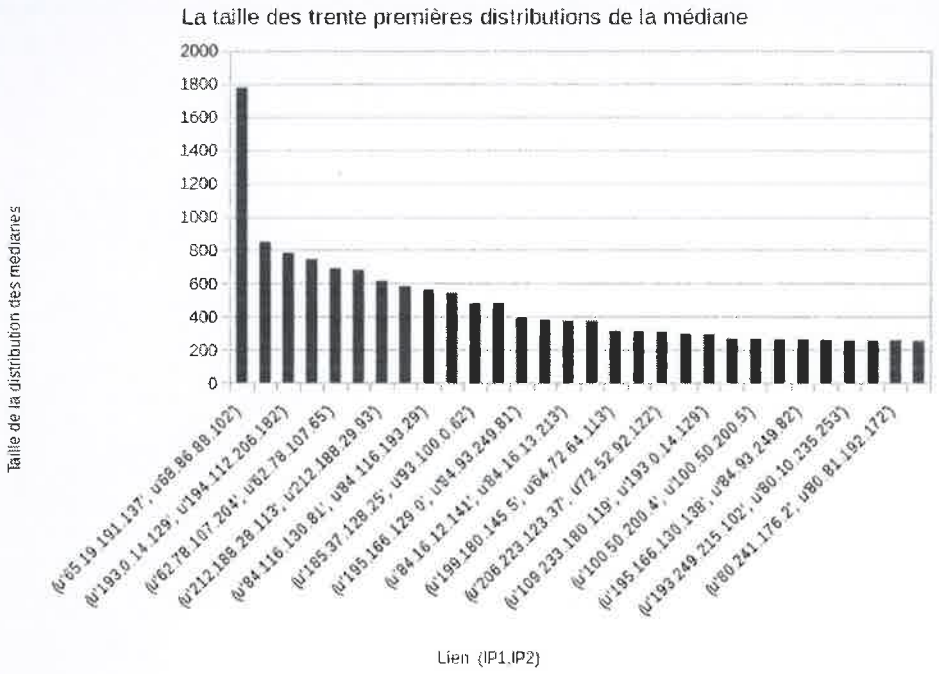


FIGURE II.5 – La taille de la distribution des médianes des RTT différentiel

II.3.7 Notes sur les traceroutes

L'analyse menée sur les traceroutes collectées par les sondes Atlas n'utilise pas totalement les traceroutes sélectionnés suivant quand ils ont été capturés.

11. `RipeAtlasTraceroutesAnalysis/tableaux/taille_distributions_medianes.pdf`, consulté le 14/10/2018.

Traceroute réussi ou échoué totalement. Dans le travail de référence, les traceroutes sont stockés tels qu'ils sont, sans prétraitement à l'avance. Or un traceroute peut être éliminé à l'avance si ce dernier a échoué pour atteindre la destination prédéfinie.

Traceroute réussi ou échoué partiellement. Dans certains cas, la sonde Atlas ne parvient pas à atteindre un routeur intermédiaire durant son chemin vers la destination finale. Rappelons que pour un saut, l'implémentation du traceroute utilisée par les sondes Atlas envoie 3 signaux par saut, ainsi, on distingue deux cas :

- la sonde ne reçoit aucune information sur les trois signaux ;
- la sonde reçoit les informations de 1 ou 2 signaux.

Les adresses IP privées dans traceroute. Dans la présente analyse, les adresses IP privées n'ont aucune valeur ajoutée. Les liens à surveiller sont ceux visibles sur Internet, alors que les adresses IP privées reflètent la configuration des réseaux non publique.

L'utilité des attributs d'un enregistrement traceroute. Une requête traceroute est caractérisée par plusieurs attributs, plus de 40, décrits dans la figure II.6. Les nœuds en gris sont de type liste, un élément de la liste est un objet formé les successeurs du nœud en question. Les nœuds en vert sont les attributs utilisés par l'outil de détection des changements anormaux dans les délais d'un lien.

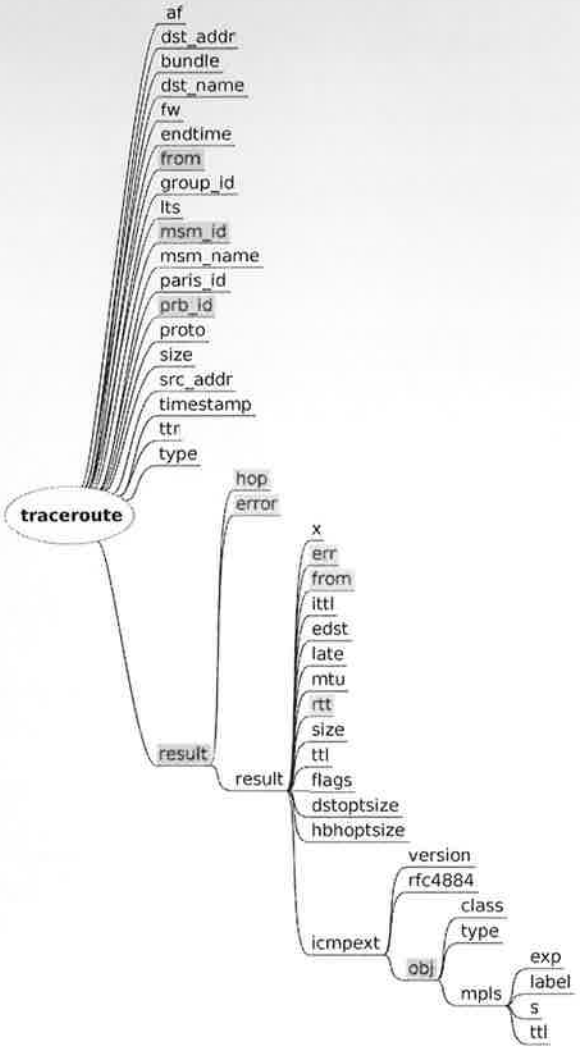


FIGURE II.6 – Les attributs possibles dans le résultat d'une requête traceroute

II.4 En pratique ...

Le travail de référence

Le processus de l'analyse se fait en récupérant d'abord les enregistrements des traceroutes depuis la base de données MongoDB. Ensuite, le traitement de chaque traceroute se poursuit dans la machine locale. Les résultats de type I, qui sont les changements des délais de tous les liens identifiés, sont stockés localement dans la machine locale. De plus, les détails de l'expérience sont aussi stockées dans MongoDB. Pour conclure toutes les opérations se déroulent dans la machine locale comme c'est illustré dans la figure II.7.

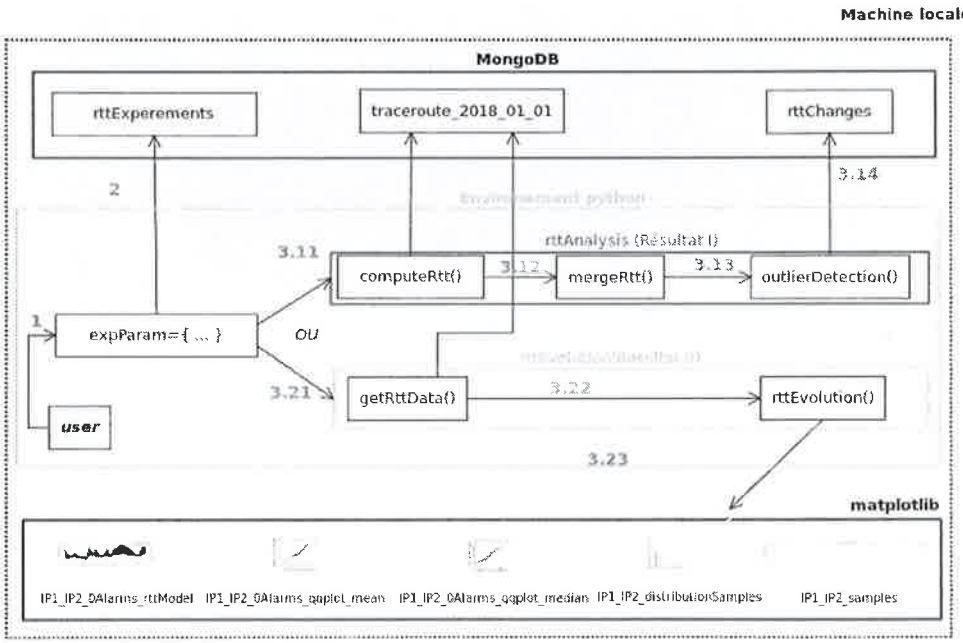


FIGURE II.7 – Le processus d'analyse des traceroutes dans le travail de référence (MongoD)

Intégration des AWS Athena + S3 dans le travail de référence

La première adaptation du travail de référence en vue d'intégrer les services web d'Amazon est présentée dans la figure II.8. La différence par rapport à l'implémentation proposée dans le travail de référence est au niveau du stockage des données. En effet, au lieu de récupérer les traceroutes depuis les collections présentes dans MongoDB (traceroute_2018_01_01), les traceroutes sont récupérés depuis le service de stockage Amazon S3. Ces données ont été adaptées pour être utilisées par l'outil de la détection.

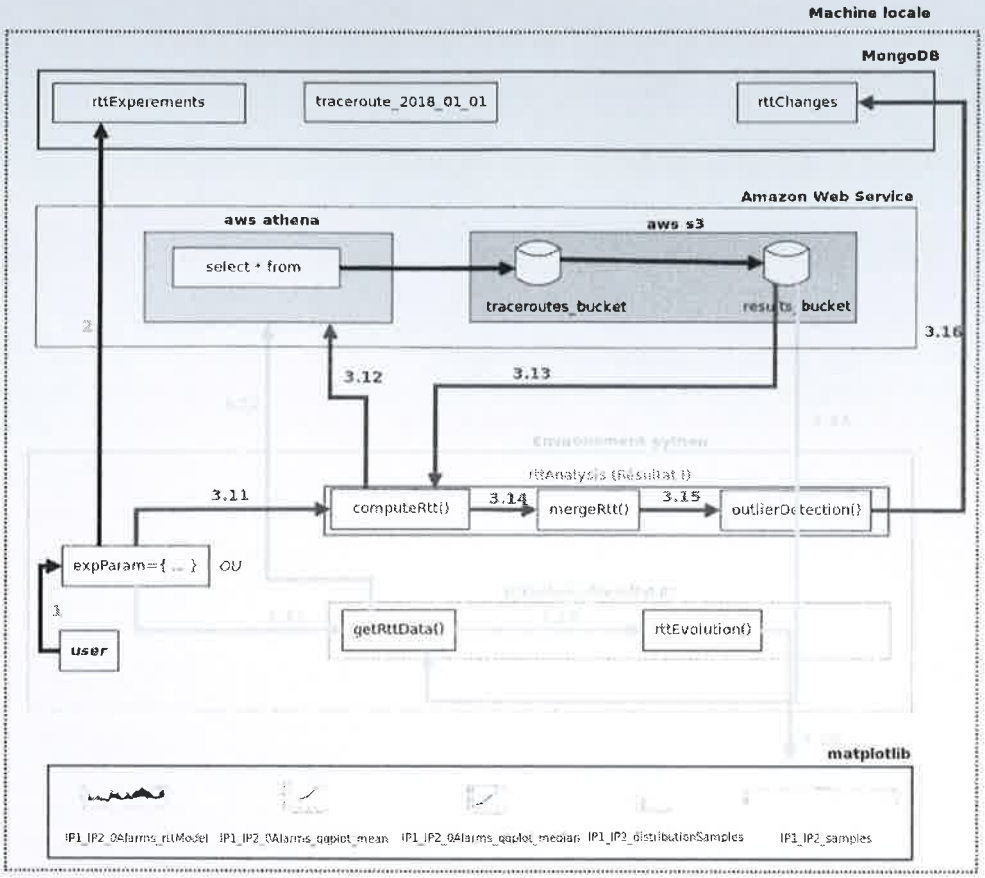


FIGURE II.8 – Intégration des AWS Athena + S3 dans le travail de référence

Aller au delà du stockage sur Amazon S3

Les données manipulées par l'outil de la détection sont volumineuses, en plus du besoin du stockage, il faut aussi adopter les outils adéquats pour le traitement de ces données, ce qui n'était pas pris en considération dans l'adaptation décrite dans la section II.4.

Et s'il existe une implémentation prenant en considération la manipulation des données massives en plus du stockage de ces dernières ? On rappelle qu'une détection se déroule en récupérant d'abords les traceroutes de la période souhaitée, ensuite, chaque traceroute est analysé pour identifier les liens avec leur RTT différentiel (computeRTT()).Ces résultats sont fusionnés avec mergeRTT(). Enfin, la détection des anomalies se déclenche (outlierDetection()). Afin d'évaluer cette possibilité, on présente dans la figure II.9 l'organigramme de la première étape dans la détection les changements des délais : computeRtt().

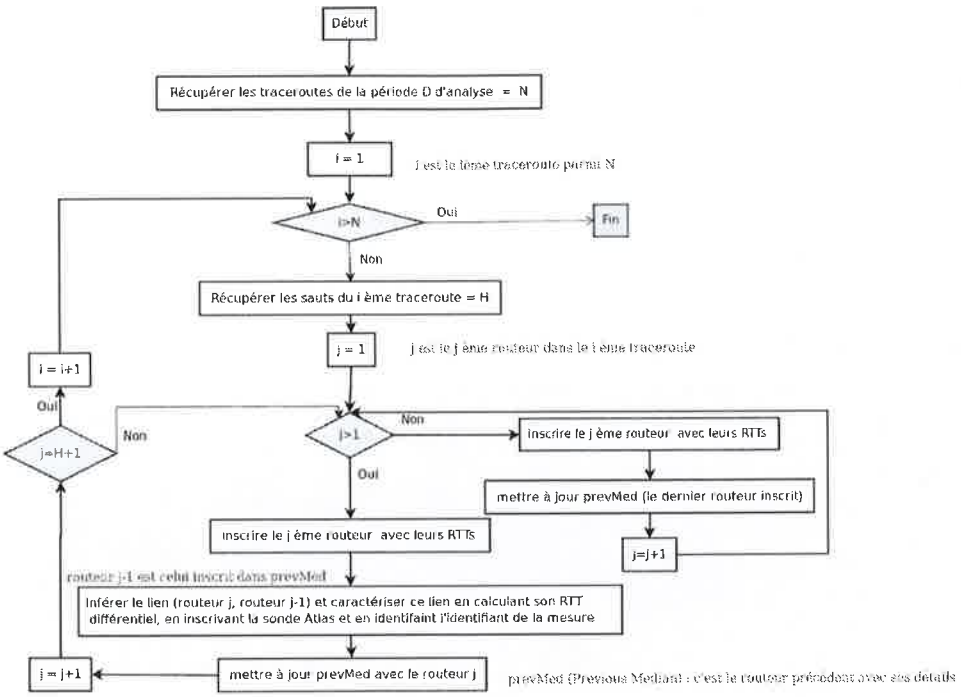


FIGURE II.9 – L'organigramme de l'étape computeRTT()

Soient les deux cas suivants :

cas 1 Supposant qu'il n'existe pas une requête SQL valide sur AWS Athena pour traiter tous les traceroutes en une seule fois, dans ce cas, on doit créer une requête par traceroute. De plus, on doit créer un fichier résultat qui stocke les résultats à passer à l'étape mergeRTT(). En pratique, une heure comme période d'analyse a repris environ 18400 traceroutes. En effet, pour une heure, il faut lancer 18400 requêtes sur AWS Athena et 18400 opérations de lecture/écriture sur un fichier résultat si on souhaite le stocker dans AWS S3. Alors pour une semaine d'analyse ?

cas 2 Jusqu'à maintenant, avec la requête suivante [12] :

12. A faire évoluer si les services web d'Amazon qui seront utilisés dans la suite du mémoire.

```
with dataset AS
    (SELECT de,
          msm_id,
          prb_id,
          mydata.hop,
          hopDetails."from",
          hopDetails.rtt
     FROM
         (SELECT "from" AS de,
          msm_id,
          prb_id ,
         mydata
         FROM traceroutes
         CROSS JOIN unnest(result) AS t(mydata)) AS tab
         CROSS JOIN unnest(mydata.result) AS tt(hopDetails))
     SELECT cast(approx_percentile(rtt,0.5) AS double) AS med ,
          "from",
          cast(hop AS integer) AS hop
FROM dataset
GROUP BY  "from", hop
```

FIGURE II.10 – Requête AWS Athena intermédiaire concernant le cas 2

Nous avons comme résultat :

Résultats

	med	from	hop
1			5
2	24.05699920654297	62.158.61.169	7
3	0.5440000295639038	196.12.10.246	2
4	182.28799438476562	196.49.6.10	6
5	3.1429998874664307	185.147.12.19	3
6	1.9229999780654907	196.216.164.1	1
7	21.86400032043457	205.171.1.18	9
8	3.1719999313354492	185.147.12.31	2
9	8.253999710083008	192.5.5.241	7
10	6.081999778747559	196.216.48.144	5
11	0.6980000138282776	160.242.100.88	3
12	7.710999965667725	193.239.116.112	6
13	64.23600006103516	196.216.48.144	4
14	3.677999973297119	160.242.100.88	4
15	22.03499984741211	205.171.234.102	8

FIGURE II.11 – Résultats de la requête AWS Athena intermédiaire concernant le cas 2

Un lien est formé par l'adresse IP ayant avec *hop* = i+1 et l'adresse IP ayant *hop* = i. Le RTT différentiel est calculé en faisant la différence entre *med* avec hop

= i+1 et med avec hop = i, avec i entier et doit concerner le même traceroute. Les résultats présentés dans II.11 ne différencient pas les sauts du même traceroute.

Supposons qu'il existe une seule requête SQL valide sur AWS Athena et capable d'assurer toutes les opérations décrites dans l'organigramme II.9 (y inclus l'inférence des liens), dans ce cas, il faut stocker les résultats intermédiaires pour qu'ils soient l'entrée de l'étape mergeRTT(). Et supposons aussi qu'il existe une requête SQL sur Athena capable de lire les résultats en question et d'appliquer la fusion entre les liens. Evaluons maintenant l'étape *outlierDetection()* [13].

On peut présenter l'étape *outlierDetection()* brièvement via l'organigramme II.12 :

13. Le principe de la détection est le même pour outlierDetection (résultat I) et rttEvolution (Résultats II)

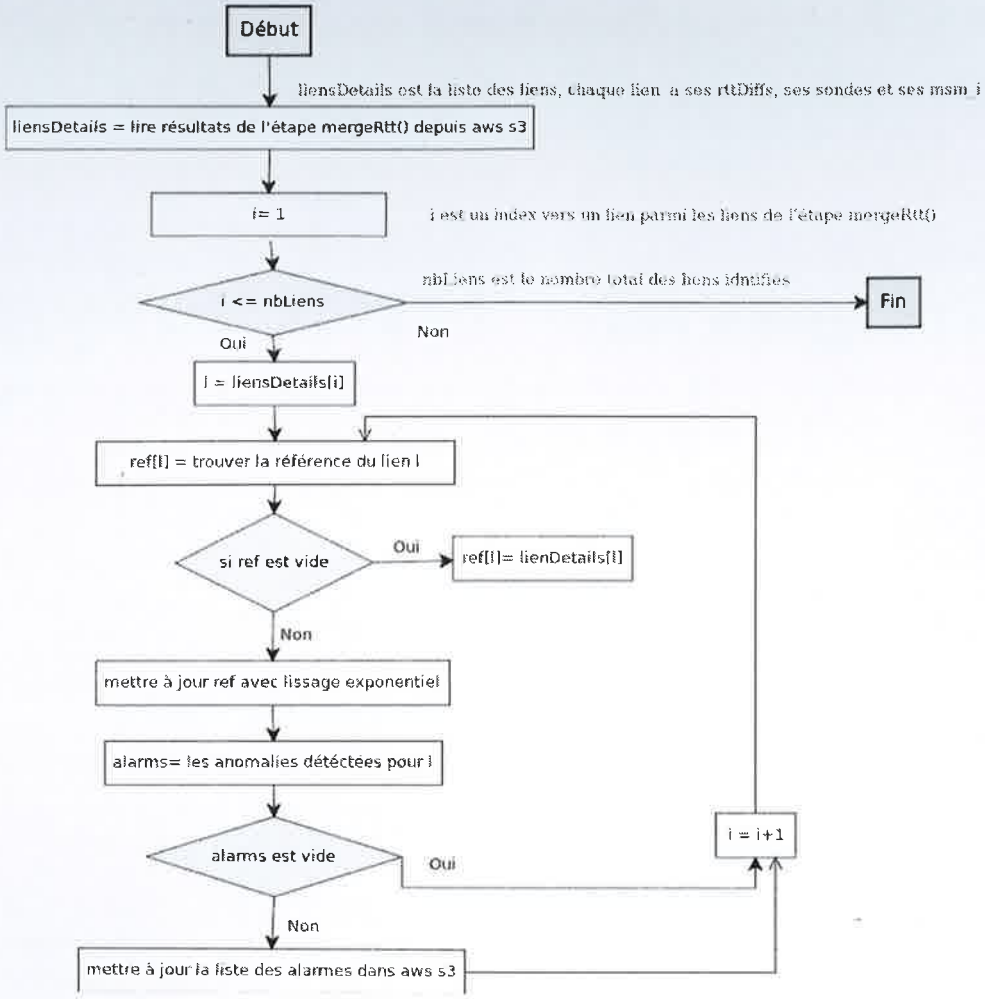


FIGURE II.12 – L'organigramme de l'étape outlierDetection()

Ce qu'on peut noter à travers cet organigramme, c'est que l'évaluation de chaque lien l nécessite l'accès en lecture aux résultats de l'étape mergeRtt() d'une part et d'accéder à la référence de chaque lien en lecture et écriture d'autre part. Si on suppose que les résultats intermédiaires sont stockés dans des fichiers disponibles sur AWS S3, est-il efficace de continuer l'analyse des traceroutes avec les deux service AWS Athena et S3 ?

En pratique si on considère l'analyse ayant les caractéristiques suivantes :

Période	entre 2015-10-20 00 :00 :00 et 2015-10-20 01 :00 :00
Id de la mesure	5004
Nb traceroutes	17816
Adressage IP	4
Nb de liens	26295

TABLE II.5 – Récapitulatif d'une expérience de détection des anomalies

Les données analysées sont disponibles sur GitHub [14]. Le tableau ?? reprend 10 (10/26295) liens issus de l'analyse décrite dans II.5. Ce sont les liens les plus rencontrés durant l'heure de l'analyse, autrement dit, ceux ayant la distribution la plus grande des RTTs différentiel.

Lien	Nb RTT différentiel	Nb de sondes
('192.5.5.241', '80.249.208.111')	3737	3737
('192.5.5.241', '80.81.194.57')	1084	1084
('192.5.5.241', '216.200.0.10')	827	827
('216.200.0.10', '64.125.31.46')	802	802
('192.5.5.241', '80.249.208.140')	629	629
('195.219.194.46', '80.249.208.111')	443	443
('64.125.25.53', '64.125.31.46')	422	422
('192.5.5.241', '193.232.244.140')	420	420
('192.5.5.241', '193.239.116.112')	396	396
('64.125.20.250', '64.125.25.53')	387	387
('192.5.5.241', '195.35.65.250')	371	371
('192.5.5.241', '193.232.246.140')	369	369
('192.5.5.241', '62.115.42.86')	360	360
('192.5.5.241', '206.223.119.2')	353	353
('192.5.5.241', '5.57.80.224')	353	353

Conclusion Le service web d'Amazon Athena permet le mode lecture seulement des données, ainsi les tables créées ne peuvent pas être mises à jour, ce sont des tables utiles pour lire les données présentes sur AWS S3. L'opération de la détection des anomalies des délais dans les liens passe par plusieurs étapes intermédiaires, de ce fait, il faut passer plusieurs opérations de lecture/écriture des résultats intermédiaires. En matière d'efficacité, on peut trouver mieux que les deux services d'Amazon Athena et S3 pour ce projet en particulier. En ce qui concerne

14. Source : https://github.com/hayatbellafkih/RipeAtlasTraceroutesAnalysis/blob/master/data/2015-10-20%2000:00:00_msmId5004.json.gz., consultée le 19/10/2018.

la médiane, on a pas considéré l'efficacité de la fonction *approx_percentile(col, 0.5)* [15] en terme de précision, vu que le choix de ces deux services d'AWS n'est pas efficace pour l'opération de la détection.

15. Plus de détails sur `https://prestodb.io/docs/current/functions/aggregate.html`, consultée le 19/10/2018.