François Espinet Gaspard Ferey

Promotion X2011

NetProbes

INF441-NET

Diagnostics réseau distribués

École Polytechnique ParisTech

Table des matières

1 NetProbes : un programme de diagnostic réseau distribué

Le but de ce programme est de permettre d'effectuer des tests sur un réseau donné.

Après avoir établi un réseau de sondes au-dessus du réseau test via http, on pourra lancer les tests de notre choix sur ce réseau et collecter les résultats pour analyse.

L'architecture de l'application nous permet d'implémenter des tests "à la volée". En effet, il suffit de placer un module python implémentant les bonnes fonctions dans le dossier probe/tests afin que le nouveau test puisse être utilisé. Notre but a donc plus été de proposer une méthode de synchronisation autour des tests plutôt que le code des tests eux-mêmes. Cependant, nous avons implémenté trois tests basiques : unicast, multicast et broadcast.

2 Guide de l'utilisateur

2.1 Prérequis

Python3 l'interpréteur python est requis. Nous avons développé notre application pour la version 3.3. Pour plus d'information sur l'installation de Python3, consulter cette page : http ://www.python.org/downloa

Connexion HTTP une connexion HTTP entre les sondes est requise afin de permettre la synchronisation autour des tests et les opération de construction du réseau de sonde.

2.2 Installation

1. Décompresser l'archive dans un dossier au choix.
2. Se placer en ligne de commande dans un des sous-dossiers : commander ou probe.
3. Lancer l'exécutable python : main.py

Remarque : dans les exemples sous Windows suivants, Python3 a éventuellement été ajouté au path via une commande du type :

```
C:\...\NetProbes>path %path%;c:\python33
```

2.3 Manuel d'utilisation

2.3.1 Options au lancement

Les options pour les différents programme sont donnés par l'option `-h` à l'invocation du main. Ceci est valable pour les deux exécutables que nous fournissons.
Exemple
Sous Windows :

```
C:\...\NetProbes>cd app\probe
C:\...\NetProbes\app\probe>python main.py -h
```

2.3.2 Sondes : package probe

La première étape de l'utilisation de notre outil consister à faire tourner plusieurs sonde sur différentes machines connectées à un même réseau. L'utilisation des sondes est transparente et autonome. Les commandes d'ajout, de suppression et de tests étant effectuées à partir de l'interface utilisateur.

Une sonde se lance par une simple exécution du programme python `main.py`.

La sonde est par défaut verbeuse, à moins de spécifier l'option —no-debug lors de son lancement.

Il est également possible de lui spécifier un identifiant avec l'option -id.

Pour quitter la sonde, la seule solution implémentée est d'utiliser Ctrl+C sous Linux ou de fermer la console sous Windows.

Exemple sous Windows : `C:\...\NetProbes\app\probe>python main.py -id sondeA --no-debug`

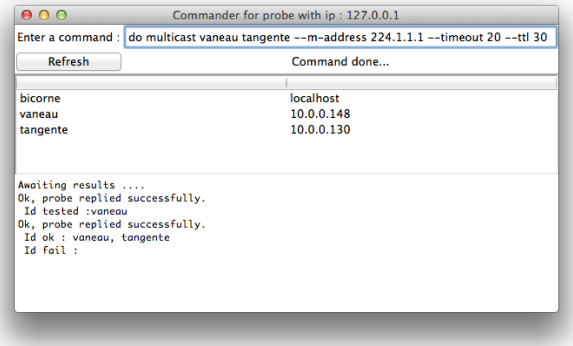
2.3.3 Interface utilisateur : package commander

L'interface permet de commander une sonde de son choix. On peut choisir l'adresse IP de cette sonde au démarrage avec l'option -ip.

Exemple sous Windows :

```
C:\...\NetProbes\app\commander>python main.py -i gui -ip 127.0.0.1
```

Un fois la connexion établie, on obtient l'affichage suivant :



ligne d'entrée permet de donner des commandes à la sonde

ligne de statut permet d'afficher le status de la commande exécutée

bouton rafraichir permet de rafraichir les sondes connues en les demandant au serveur commandé

affichage des sondes tableau id | adresse IP des sondes connues

panneau résultats affiche les résultats des tests lancés

Les commandes supportés sont les suivantes :

add ip ajoute la sonde d'adresse IP 'ip' à notre réseau de sondes. On peut ensuite effectuer des tests dessus.

remove id supprime la sonde id du réseau. Supprimer la sonde commandée permet de la retirer du réseau.

do testname testoptions effectue le test 'testname' sur la sonde commandée avec les options testoptions. Les options sont de la forme option=valeur.

Affichage des résultats

Les résultats sont automatiquement affichés lorsqu'ils sont disponibles sur la sonde commandée dans le panneau prévu à cet effet.

2.3.4 Exemples

Ajouter des sondes

```
add 10.0.0.148
add 10.0.0.130
```

Effectuer des tests

```
do unicast tangente -port 5611 -protocol tcp
do unicast vaneau -timeout 2 -protocol udp
do broadcast -port 7267 -timeout 3.5
do multicast vaneau tangente -port 12456 -timeout 3.4 -ttl 20 -ma 224.4.6.6
do empty
```

3 Architecture et fonctionnement

3.1 Architecture

On représente ci-dessous l'organisation globale des différents threads qui composent les sondes du réseau. Les conventions de représentations sont les suivantes :

Une sonde est composé dans sa version la plus simple de 3 threads et de 3 structures de stockage de données.

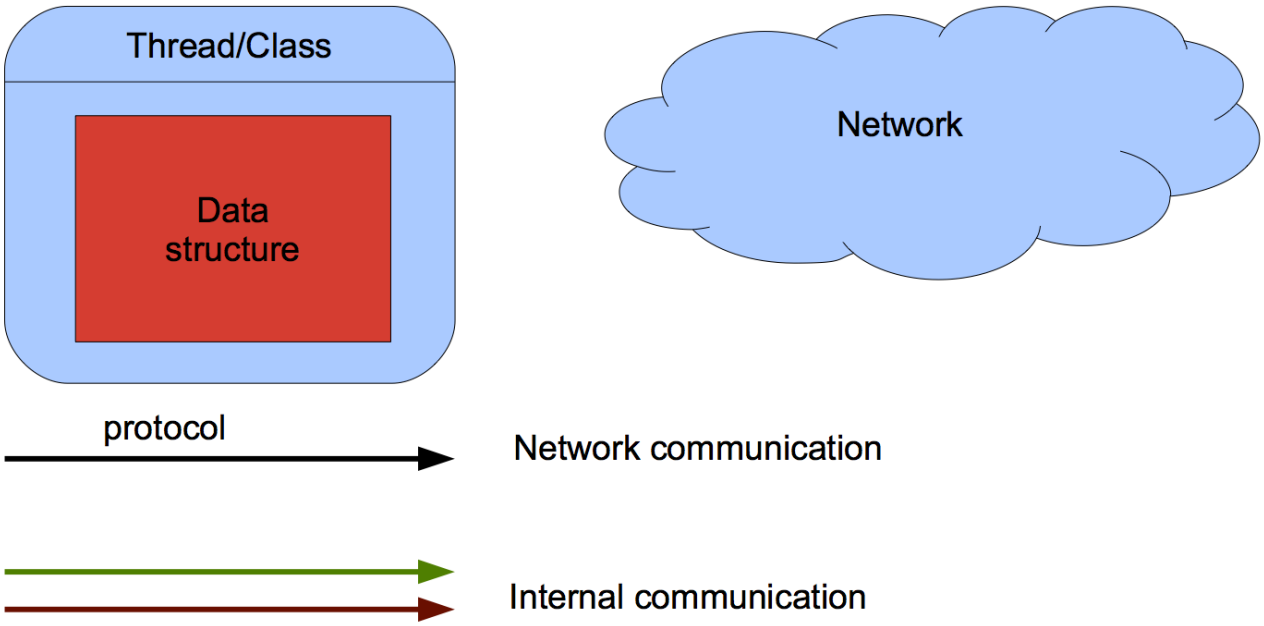


FIGURE 1 – Légende

- L'objet `ProbeStorage` contient les sondes du réseau (Id, IP), ainsi que les objets représentant les connexions HTTP vers ces sondes.
- Le thread `Client` gère une file de messages. Il utilise les connexions du `probestorage` pour les envoyer au fur et à mesure.
- Le thread `Server` écoute les requêtes HTTP sur le port de communication interne des sondes et transforme les messages reçus en actions qu'il stocke dans une file d'attente.
- Le thread `ActionMan` dépile les actions dans l'ordre de priorité et les exécute.

Le commander est un thread à part qui peut même être lancé à partir d'une machine sur laquelle ne tourne aucune sonde. Ce programme prend en entrée une machine (adresse IP) sur laquelle fonctionne une sonde avec qui communiquer. Il utilise cette sonde pour connaitre le réseau et déclencher des tests.

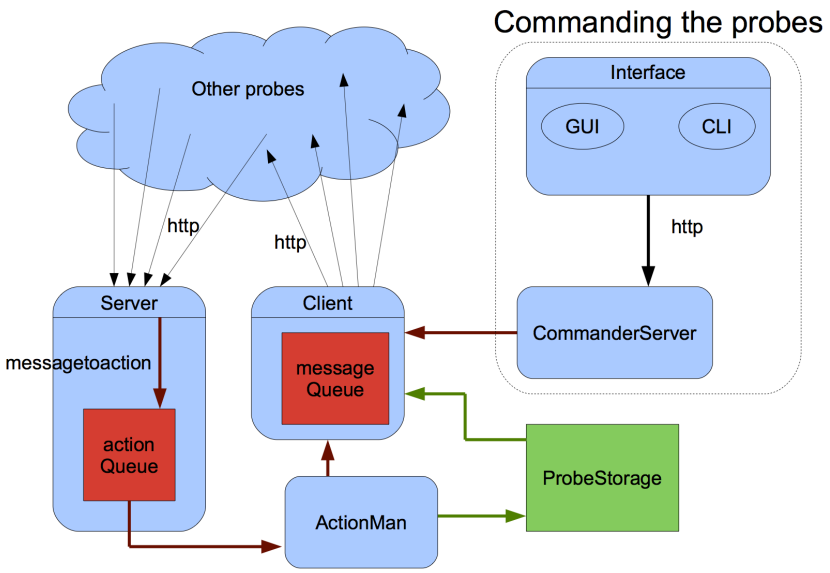


FIGURE 2 – Fonctionnement d'une sonde

Les threads mis en jeu sont :

Le thread `Interface` graphique (GUI) ou en ligne de commande (CLI) , elle est à la fois le serveur et le client qui permet de communiquer avec une sonde commandée.

Le thread `CommanderServer` tourne sur la sonde et écoute les requêtes HTTP sur le port de communication "commander" des sondes. Il réagit immédiatement aux messages reçu du commander.

Lors de l'exécution d'un test, un thread particulier, est lancé par les sondes. Le thread `ActionMan` est bloqué jusqu'à l'arrêt du test.

- `TestManager` pour la sonde qui exécute le test.
- `TestResponder` pour les sondes qui réagissent au test.

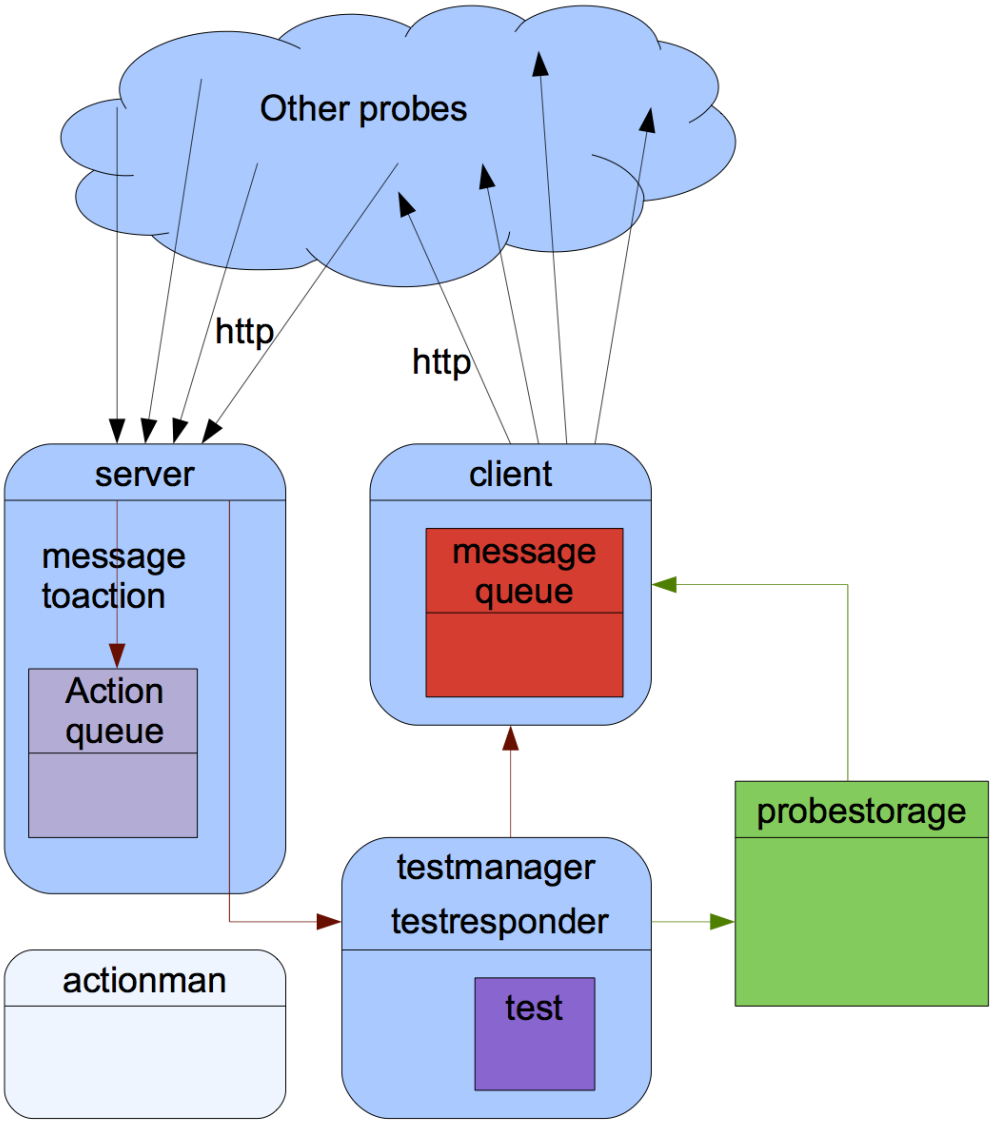


FIGURE 3 – Fonctionnement alternatif lors d'un test

3.2 Fonctionnement

3.2.1 Ajout d'une sonde au réseau

Détail des étapes lors de l'ajout d'une sonde au réseau :

- L'utilisateur lance une commande "add" : `add 10.0.0.130`
- L'interface crée un requête HTTP "POST" contenant l'IP de la machine sur laquelle tourne la sonde à ajouter.
- A la réception de cette requête, le `CommanderServer` sur lequel est branché l'interface envoie une requête "GET" à l'adresse IP désirée et attend la réponse de la sonde.
- La nouvelle sonde (si elle existe bien) renvoie son ID à la réception de la requête "GET"
- Le `CommanderServer` demande alors au `client` d'envoyer à toute les ondes (y compris celle sur laquelle il tourne) le message "Add" avec le flag "Hello" à True et les paramètres de la nouvelle sonde.
- Le client dépile les messages "Add" et les envoie au fur et à mesure.
- Les serveurs des sondes du réseau, à la réception de ce message, empilent une action "Add" avec flag "hello".

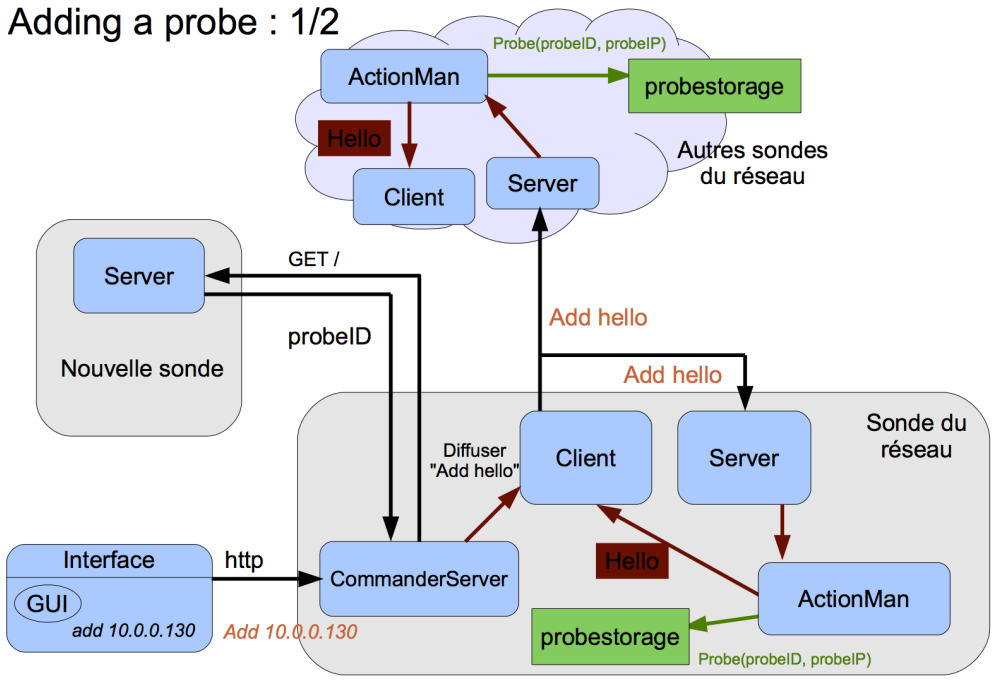


FIGURE 4 – Ajout d'une sonde - Etape 1

- Les `ActionMan` de chaque sonde dépilent cette action et l'exécutent : Ils ajoutent la nouvelle sonde dans le `ProbeStorage` et chargent le `client` d'envoyer à cette nouvelle sonde un message "Hello".
- Le client s'exécute.
- Le serveur de la nouvelle sonde empile une action "Add" sans flag "hello" pour chaque message "Hello" qu'elle reçoit.
- A la lecture de ces "Add" de toutes les sondes du réseaux, l'`ActionMan` de la nouvelle sonde se constitue un répertoire complet des sondes du réseau.

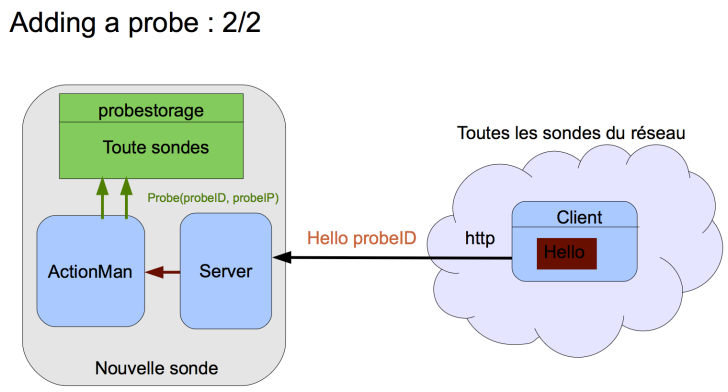


FIGURE 5 – Ajout d'une sonde - Etape 2

3.2.2 Exécution d'un test

Détail des étapes lors de l'exécution d'un test :

- L'utilisateur lance une commande "do" : `do unicast tangente`
- Le `CommanderServer` empile une action correspondant dans la pile du `Server`.
- L'`ActionMan`, à la lecture de cette action lance le test et bloque jusqu'à la fin de celui-ci. La main est laissée aux fonctions du `TestManager`.
- Le `TestManager` lit les options du test et envoie aux sondes concernées un message "Prepare" contenant les options du test.

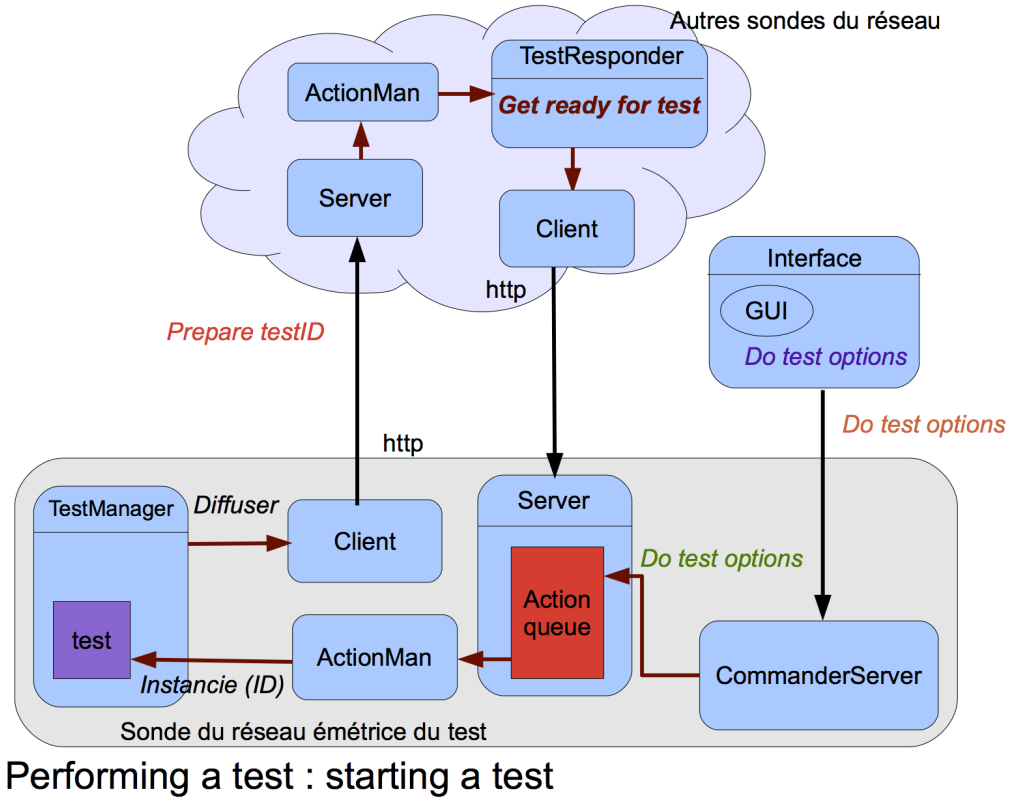


FIGURE 6 – Exécution d'un test - Etape 1

- Les serveurs de ces sondes transforme ce message en action.
- L'`ActionManager` de ces sondes bloque et cède la main aux fonctions du `TestResponder`.
- Ce dernier effectue les préparatifs pour le test puis envoie, via le `Client` un message "Ready" à la sonde qui a initié le test.
- Le `TestManager` de cette sonde attend de recevoir les messages "Ready" de toute les sondes cibles. En cas d'erreur ou de non réception d'un des "Ready", elle renvoie à tous un message "Abort".
- Le test est effectué par la sonde initiatrice.
- Le `TestManager` envoie un message "Over" à toute les sondes participant au test.
- Ces dernières répondent un message "Result" contenant les résultats qu'elles ont obtenus.
- A la réception de tout les résultats, le `TestManager` construit un rapport qu'il retourne au `CommanderServer`, lequel était bloqué jusqu'à la réception de ce rapport.
- Le `CommanderServer` répond à l'interface qui met à jour son affichage.

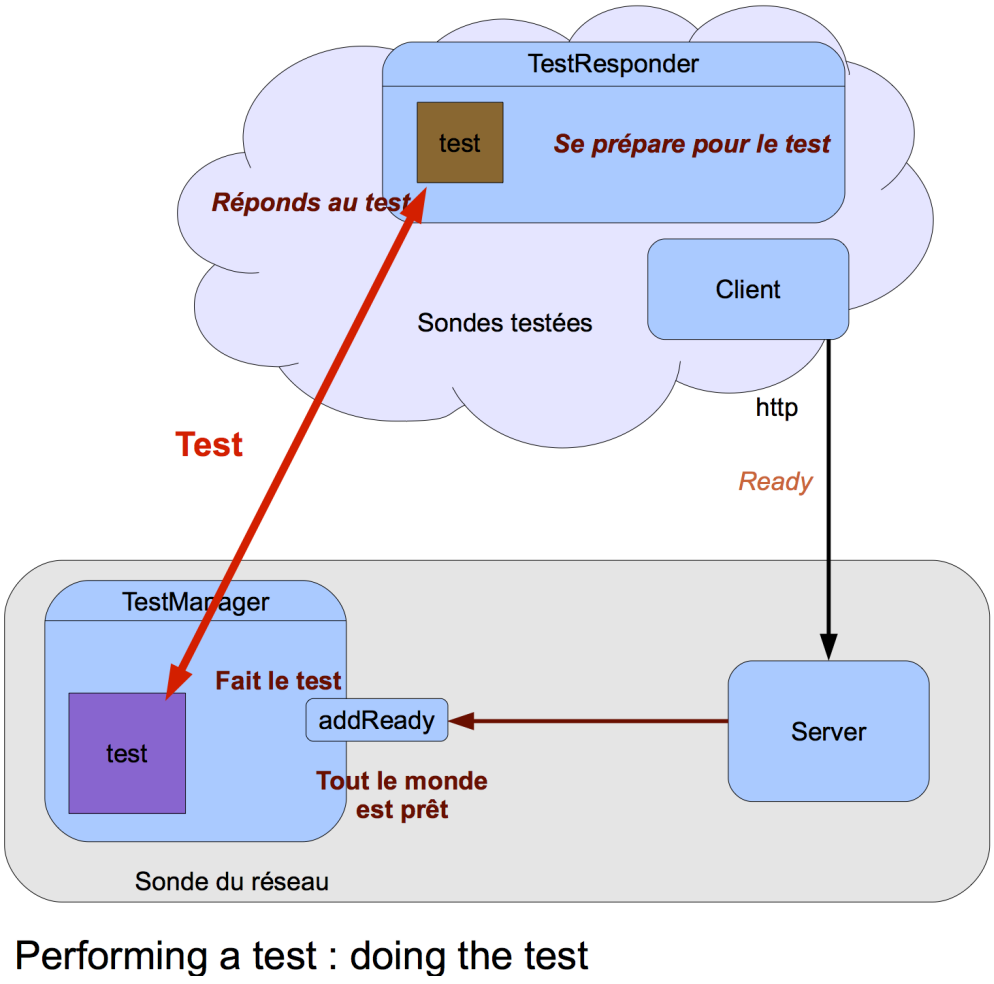


FIGURE 7 – Exécution d'un test - Etape 2

- Le `TestManager` envoie un message "Over" à toute les sonde participant au test.
- Ces dernières répondent un message "Result" contenant les résultats qu'elles ont obtenus.

- A la réception de tout les résultats, le `TestManager` construit un rapport qu'il retourne au `CommanderServer`, lequel était bloqué jusqu'à la réception de ce rapport.
- Le `CommanderServer` Répond à l'interface qui met à jour son affichage.

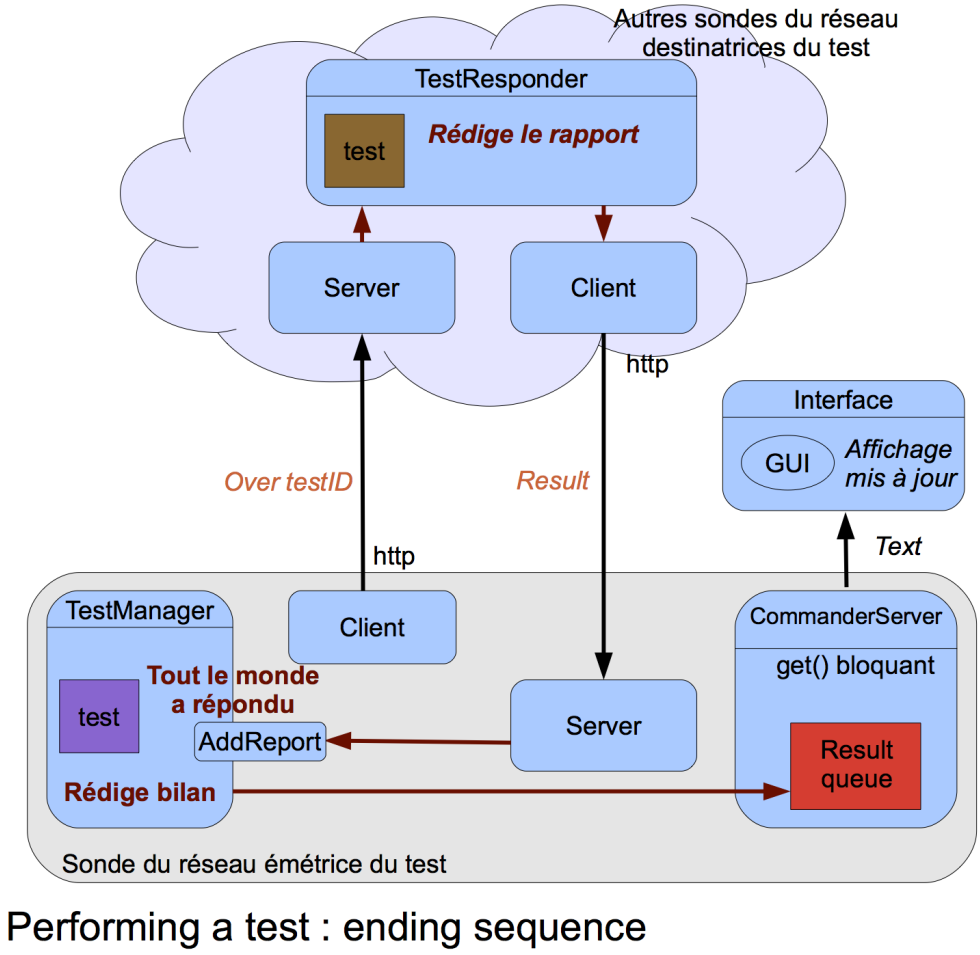


FIGURE 8 – Exécution d'un test - Etape 3

3.3 Tables de fonctionnement

`Server`	`ActionMan`		
Message	**Action**		
Add	Add	$\longrightarrow$	Ajout dans la table / Message "Hello" si nécessaire
Hello	Add	$\longrightarrow$	Ajout dans la table
Bye	Remove	$\longrightarrow$	Supprime la sonde de la table
	Quit	$\longrightarrow$	Diffuse un message "Bye" / Coupe toute les connexions sauf la sienne
Prepare	Prepare	$\longrightarrow$	Lance le test / Bloque jusqu'à la fin du test

`CommanderServer`	
Message	
Add	Requête "GET" vers la nouvelle sonde pour connaitre son ID / Diffuse un message "Add"
Delete	Envoie un message "Bye"
Do	Met une action "Do" dans le pile d'actions

Annexe : Liste des fonctions Scilab

A Organisation du code