Développement d'un outil de monitoring

PEYEN Kévin

COLLIGNON Bar

Rapport de projet

Master CHPS – 1e année

Année 2020-2021

Introduction

L'objectif du projet consiste en la réalisation d'un outil de monitoring qui permettra d'observer l'activité de plusieurs machines. Il sera développé en deux parties, la première sera les capteurs mesurant les métriques sur une machine et la seconde sera l'interface utilisée par l'utilisateur.

I. Couche réseau

Pour la communication entre nos capteurs et notre interface nous avons opté pour l'utilisation de socket. On acceptera l'IPV4 ou l'IPV6 et nous utiliserons un protocole TCP.

Du côté interface, il réagira comme un serveur, c'est-à-dire qu'il commence à initialiser un socket serveur sur une adresse pertinente et le port donnée en paramètre de la commande d'exécution du programme.
Ensuite il lance, dans un thread, une boucle infinie qui permettra d'accepter les clients à tout moment. À chaque client celui-ci créera un thread qui va recevoir les données du capteur qu'il lui est associé et les enregistrera dans une structure, en continu car la réception est un appel bloquant tant qu'il n'y a rien.

Du côté capteur, on commence à initialiser un socket sur l'adresse IP passé en paramètre de la commande et le port en second. Une fois connecté à notre interface, il récupère différentes informations. Dès que toutes les métriques voulu sont enregistrées, on les envoie une à une, dans un certain ordre pour que l'interface les reçoivent correctement. Par exemple, on envoie d'abord le nombre de cœurs du CPU avant leurs pourcentages d'utilisation, ce qui permet à l'interface de connaître le nombre de pourcentage qu'il va recevoir. Après 2 secondes on recommence les mesures et l'envoi, cela en continu.

II. Capteur

Notre capteur permet la récupération d'information sur une machine. On l'exécute sur la machine à observer, celui-ci se connecte via un socket a notre interface. Une fois fait, on commence les mesures.

D'abord, via le fichier /sys/devices/system/cpu/present, il va récupérer le nombre de cœurs présent sur le CPU de la machine.
Ensuite, via le fichier /proc/stat, on mesure et calcul l'activité sur chacun des cœurs.
Puis il termine ses mesurent sur la mémoire physique de la machine. Par le biais du fichier /proc/meminfo, il récupère la mémoire totale et la mémoire disponible.
Il envoi toutes ses métriques puis toute les 2 secondes il réitère son exécution.

III. Interface

Notre interface permettra la réception des différentes informations des capteurs connectés et de les afficher.

Le thread qui acceptera les clients aura accès à une variable partagé, passé en paramètre, qui contiendra la socket serveur, le nombre de client actuellement et un tableau dynamique de structure avec toutes les métriques. A chaque client que ce thread accepte, il incrémente le nombre de client, il réalloue suffisamment de place dans le tableau puis exécute un thread, thread_client, qui sera la boucle pour recevoir les informations de ce nouveau client à tout moment. On lui passe, par paramètre, seulement sa structure, qui correspond à celle dans le tableau dynamique. Pas de concurrence entre chaque thread_client car n'a accès qu'à leur emplacement dans le tableau.

Le thread principal quand a lui va initialiser le GUI, ncurses. Ce GUI sera découpé en 2 « fenêtres » en fonction de la taille de la console, ce qui permet le redimensionnement sans casser le visuel.
En parcourant le tableau dans la variable partagé, on affiche toutes les métriques de chaque client à la suite. On répartira entre la « fenêtre » de gauche et la « fenêtre » de droite selon leur ordre de connexion.
Toute les deux secondes le GUI se met à jour.

IV. Problème rencontré

Comme tout projet, nous avons rencontrer des nombreux problèmes au cours de notre travaille, tel que des soucis de linkage au niveau des librairies, ou encore des soucis d'accès a certains fichiers nécessitant un accès root. La plus part ont bien sur étaient résolu ou contournés au fur et a mesure de l'implémentation. Cependant, dans notre rendu, nous avons un bug, trouvé et connu, que nous avons compris, cependant nous ne savons pas comment le résoudre et d'où il pourrait venir.

Lorsque le serveur tourne, et l'exécution d'un capteur, le capteur envoie sans aucun soucis ses différentes informations. Cependant, lors de l'ajout d'un deuxième capteur, l'envoie d'informations du premier semble perturbé et le serveur ne les reçoit plus. Mais, en ajoutant un troisième, le deuxième continue a envoyer sans soucis. Suite a cela, nous avons ajouté jusqu'à 50 capteurs, et les 49 capteurs envoient sans soucis, et le serveur gère et affiche toutes les informations. Nous pensons d'après nos tests, que lors de l'acceptation du second capteur, la création du second thread n'est pas proprement crée et prend le capteur prend donc le contrôle du premier capteur, et suite a cela, les autres threads se créer sans aucun soucis.