Université du Littoral Côte d'Opale
EILCO ING3
Année universitaire 2020-2021

Enseignant : Patrick Sondi
Etudiants :

OBJETS CONNECTES/OBJETS INTELLIGENTS
(Projets de mise en œuvre et mise en pratique)

Projet 1 : Application de gestion d'un système capteurs-collecteur

I.1 Contexte

Le projet vise à réaliser un noeud_capteur et un nœud_collecteur ainsi que les applications permettant de les faire fonctionner dans le cadre d'une application de collecte de données. Les fonctionnalités attendues seront décrites pas-à-pas à travers le sujet de TP.

Afin de vous aider à réaliser ce premier projet, il est mis à votre disposition les sources d'un programme en C/C++ qui réalise les fonctions de base dont vous aurez besoin. Vous êtes libre de réaliser le projet du TP en C/C++ également ou de tout réimplémenter dans le langage de votre choix.

I.2 Description des sources fournies

Les sources fournies comprennent :

1) Une source C/C++ nœud.cpp qui implémente un modèle de nœud et qui servira à réaliser le capteur et le collecteur. Ce fichier utilise une librairie de base (net_aux) pour gérer les communications. Un fichier Makefile vous permet de compiler l'ensemble.

2) Un dossier jokes qui contient des fichiers de blagues détenus par le nœud en question.

Le fonctionnement du programme est le suivant :

Il est possible une instance de nœud dans chaque machine après avoir mis dans jokes sa liste de blagues initiale. Notons N1 et N2 les nœuds lancés depuis les machines M1 et M2. Si on donne à N1 l'adresse IP de la machine M2, N1 utilise son interface client (lancé dans une thread) pour aller se connecter à N2. N1 ajoute alors N2 dans sa liste de voisins, puis demande à N2 de lui transmettre sa liste de blagues. N1 vérifie les blagues de N2 qui existent déjà dans sa base (dossier jokes) et ne rajoute que celles qui n'y sont pas. Vu que N1 s'est connecté à N2, ce dernier connais désormais N1 et peut donc le contacter à son tour pour récupérer sa liste de blagues.

I.3 Test des sources fournies

Testez le programme fourni depuis une même machine (N1=127.0.0.1:8001 et N2=127.0.0.1:8002). Pensez à créer un dossier jokes spécifique pour chaque nœud N1 et N2 et à y mettre quelques blagues communes et quelques blagues différentes.

I.4 Réalisation des communications

1) Modifiez le code nœud.cpp pour obtenir un modèle de nœud_collecteur.cpp. Ce dernier n'a plus de dossier jokes, mais il dispose d'un dossier database_N1 pour chaque capteur N1 déployé dans le réseau où il stocke respectivement toutes les données provenant de ce capteur. Votre code noeud_collecteur.cpp doit créer ce dossier à chaque fois qu'un capteur est créé (ou au moment où il se connecte au collecteur pour la première fois).

2) Modifiez le code nœud.cpp pour obtenir un modèle de noeud_capteur.cpp. Ce dernier n'a plus de dossier jokes, mais il dispose d'un dossier data_N1 (s'il a l'ID N1). Ce dossier est créé automatiquement par le capteur au moment de son lancement.

3) En fonctionnement, le nœud_capteur envoie une notification au nœud_collecteur pour qu'il vienne récupérer les données qui s'y trouve. Le noeud_collecteur se connecte au noeud_capteur dès qu'il peut et récupère les données. Ce dernier efface toutes les données envoyées au noeud_collecteur lors d'une telle session. Modifiez les instructions du programme nœud_capteur afin d'exprimer une notification de la présence de données et celui de nœud_collecteur afin qu'il demande et récupère ces données. Si le nœud_collecteur reçoit deux fois la même donnée, il ne doit l'enregistrer qu'une seule fois. Réalisez également l'effacement par le noeud_capteur des données transmises.

I.5 Gestion du stockage côté capteur

A une périodicitié à définir T, le nœud_capteur crée un fichier data d'une certaine taille S qui correspond à des données collectées par l'ensemble de capteurs qu'il héberge. On suppose que le dossier data est plein lorsque le nombre de fichiers qui y sont stockés atteint une valeur N. Avant que ce nombre soit atteint le nœud_capteur doit solliciter le noeud_collecteur afin qu'il récupère les fichiers. Si cette valeur est atteinte et qu'il y a de nouveaux fichiers générés, le nœud capteur ne doit pas les stocker, mais il doit compter tous les fichiers générés pendant ce temps.

1) Proposez une (ou des) relation(s) entre T, S et N si l'on suppose que le noeud_collecteur réagit immédiatement lorsqu'il est sollicité pour récupérer les données et que le temps de transmission de l'ensemble de données est très petit par rapport à T.

2) Proposez une (ou des) nouvelle(s) relation(s) entre T, S, N, Dc et Dt si l'on suppose que le noeud_collecteur réagit avec un délai Dc lorsqu'il est sollicité pour récupérer les données et que le temps de transmission de l'ensemble de données est Dt.

3) Modifiez le code de noeud_capteur avec T=2s, S quelconque et N=20. Faites en sorte que le noeud_capteur notifie la présence de données à chaque fois qu'il y a 15 fichiers dans le dossier data.

I.6 Gestion de la batterie

1) Soit B=1000 le niveau maximum initial de la batterie de chaque noeud_capteur a sa création. Chaque création de fichier data consomme 50 et chaque transmission d'un fichier data consomme 20. Chaque envoi d'une notification de disponibilité de données consomme 10. Utilisez une variable pour représenter l'évolution du niveau de batterie de chaque capteur et modifiez le code de noeud_capteur partout où il faut pour que la valeur de B évolue avec le temps.

2) Le noeud_collecteur dispose d'un dossier configs contenant 3 fichiers de configuration (full_config, eco_config et min_config). Créez-les. A chaque action, le noeud_capteur vérifie la valeur de B. Lorsque celle-ci atteint 600, il réclame un fichier eco_config au noeud_collecteur, et lorsqu'il atteint 300, il demande le fichier min_config. Modélisez ce comportement dans les codes de noeud_capteur et noeud_collecteur.

3) Chaque réception de fichier de configuration consomme 30. Mettez à jour le code de noeud_capteur.

I.6 Gestion des statistiques

1) Mettez en place une variable pour compter tous les fichiers générés par un nœud_capteur depuis son lancement

2) Mettez en place une variable pour compter tous les fichiers perdus par le nœud_capteur depuis son lancement

3) Lorsque le niveau de batterie atteint 50, tous les fichiers générés et non encore transmis sont comptés comme perdus, et le capteur ne doit plus les transmettre

4) Mettez en place une variable pour compter tous les fichiers transmis par le noeud_capteur au noeud_collecteur

5) Mettez en place une variable pour compter tous les fichiers reçus par le noeud_collecteur pour chaque noeud_capteur

6) Proposez un moyen pour calculer le taux de perte interne à chaque noeud_capteur. Réalisez-le

7) Proposer un moyen pour calculer le taux de perte dans la liaison noeud_capteur – noeud_collecteur. Réalisez-le

8) Modifiez les codes du noeud_capteur et du nœud_collecteur afin que les deux programmes affichent leurs statistiques avant de s'arrêter de fonctionner.

I.7 Résultats

Réalisez le fonctionnement de ce système avec deux capteurs (initalisés avec B=1000 et B=700) et un collecteur sur une période de 300 secondes. Analysez les résultats.