L'Institut National des Postes et Télécommunications

Le processus d'une machine d'un reseau Token Ring

Projet d'atelier

Réalisé par :
— Majda Bendifi
— Meryem El Mansouri
— Nohaila Bentaleb

Encadrant :
— Mr. Bensaid Hicham

Année universitaire :
— 2024/2025

Table des matières

1 Introduction

Le but de ce projet est de créer un simulateur d'un réseau Token Ring. Ce type de réseau est une topologie à anneau où un jeton circulant dans une seule direction permet la transmission des données. Chaque station transmet uniquement lorsqu'elle détient le jeton.

Ce rapport explique le fonctionnement de l'algorithme, la conception du simulateur, les commandes Git utilisées, ainsi que le code source développé.

2 Algorithme

Voici le fonctionnement logique d'un noeud dans un réseau Token Ring :

```
while (true) {
    attendre_le_jeton();
    if (a_des_choses_a_transmettre) {
        transmettre_la_trame_a_transmettre();
        passer_le_jeton_a_la_station_suivante();
    } else {
        passer_le_jeton_a_la_station_suivante();
    }
}
```

Listing 1 – Algorithme 1 : Fonctionnement d'un noeud Token Ring

3 Conception du simulateur

Le simulateur est implémenté en langage C. Il permet de paramétrer le nombre de noeuds et de suivre l'état du réseau au fil des ítérations.

3.1 Fonctionnalités principales

— Paramétrage dynamique du nombre de noeuds.
— Gestion du jeton et transmission de trames.
— Simulations avec gestion des pannes.

4 Code source

Voici le code source du simulateur :

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX_NOEUDS 10
#define JETON_INIT 1

typedef struct {
    char jeton[3];
    int id_emetteur;
```

```
} TrameJeton;

bool est_noeud_actif(int id_noeud) {
    return id_noeud != 2;
}

void simulation_anneau_jeton(int nb_noeuds) {
    int canaux[MAX_NOEUDS][2];
    pid_t pids[MAX_NOEUDS];

    for (int i = 0; i < nb_noeuds; i++) {
        if (pipe(canaux[i]) == -1) {
            perror("Erreur lors de la creation du canal");
            exit(1);
        }
    }

    for (int i = 0; i < nb_noeuds; i++) {
        pid_t pid = fork();
        if (pid < 0) {
            perror("Erreur lors du fork");
            exit(1);
        }

        if (pid == 0) {
            TrameJeton jeton_recu;

            while (1) {
                read(canaux[i][0], &jeton_recu, sizeof(jeton_recu));
                printf("Noeud %d : Jeton recu de %d\n", i, jeton_recu.
                    id_emetteur);

                if (!est_noeud_actif(i)) {
                    printf("Noeud %d est en panne. Passage du jeton.\n
                        ", i);
                } else {
                    if (rand() % 2) {
                        printf("Noeud %d : Envoi des donnees.\n", i);
                        sleep(1);
                    } else {
                        printf("Noeud %d : Aucune donnee a transmettre
                            .\n", i);
                    }
                }

                int noeud_suivant = (i + 1) % nb_noeuds;
                jeton_recu.id_emetteur = i;
                write(canaux[noeud_suivant][1], &jeton_recu, sizeof(
                    jeton_recu));

                sleep(1);
            }
            exit(0);
```

```
62          } else {
63              pids[i] = pid;
64          }
65      }
66
67      TrameJeton jeton = { {1, 0, 0}, -1 };
68      printf("Demarrage de la simulation de l'anneau a jeton avec %d
           noeuds.\n", nb_noeuds);
69      write(canaux[0][1], &jeton, sizeof(jeton));
70
71      for (int i = 0; i < nb_noeuds; i++) {
72          wait(NULL);
73      }
74  }
75
76  int main() {
77      int nb_noeuds;
78
79      printf("Entrez le nombre de noeuds (max %d) : ", MAX_NOEUDS);
80      scanf("%d", &nb_noeuds);
81
82      if (nb_noeuds <= 0 || nb_noeuds > MAX_NOEUDS) {
83          printf("Le nombre de noeuds doit etre entre 1 et %d.\n",
               MAX_NOEUDS);
84          return 1;
85      }
86
87      srand(getpid());
88
89      simulation_anneau_jeton(nb_noeuds);
90
91      return 0;
92  }
93
94  }
```

Listing 2 – Code C du simulateur

5 Exemple d'exécution

Pour illustrer le fonctionnement du simulateur, voici un exemple d'exécution du programme avec 3 noeuds. Les messages affichés permettent de suivre la simulation du réseau Token Ring.

5.1 Commandes d'exécution

Le programme peut être compilé et exécuté avec les commandes suivantes :

```
1  gcc -o token_ring token_ring.c
2  ./token_ring
```

Listing 3 – Compilation et exécution du programme

5.2 Exemple de sortie

Voici un exemple des messages affichés lors de l'exécution avec 3 noeuds :

```
Demarrage de la simulation de l'anneau a jeton avec 4 noeuds.
Noeud 0 : Jeton recu de -1
Noeud 0 : Aucune donnee a transmettre.
Noeud 1 : Jeton recu de 0
Noeud 1 : Envoi des donnees.
Noeud 2 : Jeton recu de 1
Noeud 2 est en panne. Passage du jeton.
Noeud 3 : Jeton recu de 2
Noeud 3 : Envoi des donnees.
...
...
```

Listing 4 – Exemple de sortie d'exécution

Observez les messages affichés pour suivre la simulation du réseau Token Ring. Chaque noeud vérifie s'il a des données à transmettre et, si c'est le cas, effectue une transmission avant de passer le jeton au noeud suivant.

6 Utilisation de Git et GitHub

Pour collaborer efficacement, nous avons utilisé Git et GitHub. Le développement du projet a ètè organisé de manière à permettre une gestion fluide et structurée des contributions.

6.1 Initialisation du projet

Le projet a été initialisé avec les commandes suivantes :

```
git init
git add .
git commit -m "Initialisation du projet"
git branch -M main
git remote add origin <url_du_d\'ep\^ot>
git push -u origin main
```

Listing 5 – Initialisation du dépôt

Cela a permis de créer le dépôt principal sur GitHub pour centraliser le projet.

6.2 Collaboration

Les membres ont utilisé Git pour synchroniser leurs modifications et travailler sur des fonctionnalités. Voici les commandes principales utilisées :

```
git pull # R\'ecup\'erer les modifications depuis le d\'ep\^ot distant

git add fichier_modifi\'e.c
git commit -m "Ajout de la g\'en\'eration de donn\'ees al\'eatoires"
git push
```

Listing 6 – Commandes de collaboration

6.3 Gestion des branches et intégration

Les branches ont ˜eté utilisées pour isoler les modifications. Voici un exemple des ˜etapes suivies pour créer, modifier et fusionner une branche :

```
# Cr\'eer une branche pour la documentation
git branch doc
git checkout doc

# R\'edaction de la documentation
git add rapport.tex
git commit -m "Ajout de la section sur l'utilisation de Git et GitHub"
git push -u origin doc

# Fusion dans la branche principale
git checkout main
git merge doc
git push
```

Listing 7 – Créer et fusionner des branches

6.4 Suivi des contributions

L'historique des contributions sur GitHub montre l'implication de chaque membre. Les pull requests ont ˜eté utilisées pour discuter des modifications importantes et les valider avant leur intégration.

7 Conclusion

Ce projet nous a permis d'apprendre à utiliser Git et GitHub pour collaborer efficacement. Les branches ont aidé à organiser les contributions de manière structurée. Ce travail pourrait être amélioré en automatisant les tests via GitHub Actions pour un développement encore plus fluide.