Rapport Technique : Simulation du Réseau Token Ring

Réalisé par :
NAIT ELHAJ ALI KHADIJA, EL ASRI YASSINE, ABDELLAH ABARCHIHI
INPT - Cloud et IoT

January 21, 2025

Contents

1 Introduction

Ce rapport détaille notre travail réalisé dans le cadre du projet d'atlier d'ingenierie d'un réseau Token Ring. IL a pour objectif de présenter le travail demandé, qui consiste à développer une simulation d'un réseau Token Ring.
Notre travail a été réalisé en groupe, en utilisant le langage de programmation C pour le développement et GitHub pour la collaboration et le partage de code.

2 Description du travail:

2.1 Étude du protocole Token Ring

Avant de commencer l'implémentation, nous avons étudié le fonctionnement du réseau Token Ring. Dans ce type de réseau :

- Le jeton circule dans un anneau de nœuds.
- Un nœud peut insérer des données dans le jeton lorsqu'il le reçoit, puis le passe au nœud suivant.
- Si un nœud est en panne, des mécanismes doivent être mis en place pour éviter le blocage du réseau.

2.2 Planification et répartition des tâches

Le projet a été divisé en trois parties pour permettre une répartition équitable entre les membres du groupe :

1. **Initialisation des Structures et Mise en Place de l'Architecture du Réseau.** (réalisé par ***EL ASRI YASSINE***)
2. **Gestion des Interactions et Simulation des Étapes du Réseau .** (réalisé par ***NAIT ELHAJ ALI KHADIJA***)
3. **Ajout de Données et Intégration de la Simulation**. (réalisé par **ABDELLAH ABRECHIHI**)

3 Algorithme Utilisé

L'algorithme utilisé pour la simulation est décrit ci-dessous :

Listing 1: Algorithme du réseau Token Ring

```
while (true) {
    attendre le jeton;
    if (donn es    transmettre) {
        transmettre les donn es;
        passer le jeton au suivant;
    } else {
        passer le jeton au suivant;
    }
}
```

4 Implémentation

4.1 Langage Utilisé

Nous avons choisi le langage C pour ses performances et sa capacité à gérer les structures de données bas niveau.

4.2 Code Source

Voici le code complet du simulateur :

Listing 2: Code C pour la simulation du réseau Token Ring

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

// Structure repr sentant une station
typedef struct Station {
    int id;                     // Identifiant de la station
    bool has_token;             // Indique si la station a le jeton
    struct Station* next;       // Pointeur vers la station suivante
    int data_to_send[10];       // Donn es    transmettre
    int data_count;             // Nombre de donn es    transmettre
} Station;

// Structure du r seau Token Ring
typedef struct {
    Station* stations;          // Tableau des stations
    int num_stations;           // Nombre de stations
    Station* current_token_holder; // Station avec le jeton
} TokenRingNetwork;

// Fonction pour cr er une station
Station* create_station(int id) {
    Station* station = (Station*)malloc(sizeof(Station));
    station->id = id;
    station->has_token = false;
    station->next = NULL;
    station->data_count = 0;
    return station;
}

// Fonction pour initialiser le r seau Token Ring
TokenRingNetwork* create_token_ring_network(int num_stations) {
    TokenRingNetwork* network = (TokenRingNetwork*)malloc(sizeof(TokenRingNetw
    network->num_stations = num_stations;
    network->stations = (Station*)malloc(num_stations * sizeof(Station));

    for (int i = 0; i < num_stations; i++) {
        network->stations[i] = *create_station(i);
        if (i > 0) {
```

```
            network->stations[i - 1].next = &network->stations[i];
        }
    }
    network->stations[num_stations - 1].next = &network->stations[0];
    network->current_token_holder = &network->stations[0];
    network->current_token_holder->has_token = true;

    return network;
}

// Fonction pour simuler une  tape  du r seau
void simulate_step(TokenRingNetwork* network) {
    Station* current_station = network->current_token_holder;

    if (current_station->data_count > 0) {
        printf("Station %d transmet des donn es: %d\\n",
                current_station->id, current_station->data_to_send[0]);
        for (int i = 0; i < current_station->data_count - 1; i++) {
            current_station->data_to_send[i] = current_station->data_to_send[i
        }
        current_station->data_count--;
    }

    current_station->has_token = false;
    current_station->next->has_token = true;
    network->current_token_holder = current_station->next;
}

// Fonction pour ajouter des donn es    une station
void add_data_to_station(TokenRingNetwork* network, int station_id, int data)
    if (station_id >= 0 && station_id < network->num_stations) {
        Station* station = &network->stations[station_id];
        if (station->data_count < 10) {
            station->data_to_send[station->data_count] = data;
            station->data_count++;
            printf("Donn e %d ajout e    la station %d\\n", data, station_id
        } else {
            printf("Station %d pleine\\n", station_id);
        }
    } else {
        printf("Station invalide\\n");
    }
}

// Fonction principale
int main() {
    int num_stations = 5;
    TokenRingNetwork* network = create_token_ring_network(num_stations);

    add_data_to_station(network, 0, 100);
    add_data_to_station(network, 2, 200);
```

```
    for (int i = 0; i < 10; i++) {
        printf("\\n tape %d :\\n", i + 1);
        simulate_step(network);
    }

    free(network->stations);
    free(network);

    return 0;
}
```

4.3 Instructions d'Exécution

1. Compiler le fichier avec GCC :

```
gcc token_ring.c -o token_ring
```

2. Exécuter le programme :

```
./token_ring
```

5 Résultats d'Exécution

Voici un exemple des résultats obtenus après l'exécution du programme principal :

```
Donnée 100 ajoutée à la Station 0
Donnée 200 ajoutée à la Station 1
Donnée 300 ajoutée à la Station 3

Étape 1:
Station 0 transmet des données: 100
Station 0 passe le jeton à Station 1

Étape 2:
Station 1 transmet des données: 200
Station 1 passe le jeton à Station 2

Étape 3:
Station 2 passe le jeton à Station 3

Étape 4:
Station 3 transmet des données: 300
Station 3 passe le jeton à Station 4

Étape 5:
Station 4 passe le jeton à Station 0
```

```
Étape 6:
Station 0 passe le jeton à Station 1

Étape 7:
Station 1 passe le jeton à Station 2

Étape 8:
Station 2 passe le jeton à Station 3

Étape 9:
Station 3 passe le jeton à Station 4

Étape 10:
Station 4 passe le jeton à Station 0
```

6 Conclusion

Ce projet nous a permis de mieux comprendre le protocole Token Ring, de développer une simulation en C et de collaborer efficacement avec GitHub.