Rapport TER

Interface générique pour le suivi de l'exécution d'une application distribuée

Membres du groupe :
Jalal Azouzout
Ali Ba faqas
Leila Bourouf
Saghar Yazdani

Encadrante :
Hinde BOUZIANE

12 mai 2023

Table des matières

1 Introduction et positionnement du sujet

1.1 Rappel du sujet du projet

Programmation d'une interface pour le suivi de l'exécution d'une application distribuée

Encadrante : Hinde Bouziane (hinde.bouziane@lirmm.fr)

Nous souhaitons pourvoir suivre l'évolution de l'état d'un ensemble de processus composant une application distribuée (les processus sont interconnectés via un réseau TCP/IP). Nous ciblons en particulier des applications répondant à une problématique donnée dans le cadre de l'algorithmique répartie (exclusion mutuelle, élection, diffusion, etc.).

Dans le cas par exemple de l'exclusion mutuelle, plusieurs processus partagent une même ressources et lorsqu'un processus utilise cette ressource, aucun autre ne peut l'utiliser en même temps. Dans ce cas, nous souhaitons visualiser qui est en train d'utiliser la ressource à tout moment de l'exécution. L'application à réaliser doit donc fournir une interface aux processus pour que chacun puisse envoyer son état courant (pour l'exemple : utilisation ou non utilisation de la ressource). Cette interface serait générique, pouvant être utilisée pour différentes applications et différents types d'états.

Le langage à utiliser est C/C++. Dans les grandes lignes, les choses à faire seraient : - se familiariser avec deux ou trois algorithmes distribués pour défnir et comprendre ce qui serait à visualiser - étude de quelques bibliothèques spécialisées pour la réalisation d'une interface graphique et faire un choix adapté au problème énoncé - modélisation et implémentation - établir un guide d'utilisation, en particulier défnir ce que doit faire un utilisateur qui arrive avec une nouvelle application pour que cette dernière soit "branchée" à l'interface graphique - "branchement" à deux ou trois applications distribuées (fournies) pour la visualisation

Prérequis :
- systèmes d'exploitation - réseaux - l'algorithmique est un grand plus - aimer programmer en C/C++ facilitera grandement la réalisation de ce projet - remarque : ce projet nécessite des notions de programmation concurrente et répartie qui seront vues en parallèle dans le carde de l'UE HAI604I.

1.2 Présentation du problème : Visualisation de l'évolution d'une application distribuée

Contexte

Dans une application distribuée, plusieurs processus sont interconnectés grâce au protocole TCP. Ils peuvent être organisés de différentes manières, en anneaux, en un réseau complet, en arbres, ect. Aussi, ils échangent ou partagent des ressources entre eux, provoquant des changements d'état.

Exemples d'algorithmes distribués

Exclusion mutuelle : Les processus participants passent de la section critique à la section normale et vice versa. À tout moment, au moins un processus est dans la section critique.

Élection : Les processus s'échangent des messages pour élire un leader.

Coloration de graphes : Chaque processus (nœud) reçoit une couleur différente de ses voisins. L'état d'un processus est représenté par une couleur.

Objectif

L'objectif est de créer une interface qui illustre l'évolution d'une application distribuée. Cette interface doit non seulement permettre la visualisation des états des processus et les connexions entre eux, mais aussi fournir une API pour les développeurs d'applications distribuées. Cette API leur permettra d'envoyer les états de leur application à l'interface et de visualiser l'évolution de l'application en temps réel.

1.3 Motivation

Nous avons choisi ce projet car nous sommes intéressés par les applications distribuées et le réseau, ainsi que par ce domaine en général.

En effet, les applications distribuées sont de plus en plus utilisées dans le monde de l'informatique. De plus, grâce au module HAI604I programmation multitâche que nous avons suivi durant le semestre, nous avons acquis des connaissances en parallèle de notre projet, notamment en réseau, en programmation multi-thread, en système et en programmation multi-processus, que nous avons pu mettre en pratique dans notre projet.

1.4 Cahier des charges

1. Création d'un processus capable de changer d'état (0 et 1) et de l'afficher dans le terminal.
2. Recherche et sélection de la meilleure bibliothèque pour implémenter l'interface graphique adaptée au projet.
3. Développement d'un programme qui crée l'interface graphique et affiche un changement d'état (0e t 1) en boucle.
4. Conception d'un site qui se connecte à l'interface graphique, change son état en boucle et envoie l'état à l'interface via une connexion TCP.
5. Création de deux sites qui se connectent l'un à l'autre, chacun changeant d'état en boucle et envoyant l'état au serveur principal.
6. Création d'un programme nommé MainServeur, capable de recevoir plusieurs connexions de sites.
7. Finalisation du MainServeur et développement de deux sites ayant deux états (actif et en attente), avec envoi des états au serveur principal pour affichage graphique.
8. Recherche et amélioration de l'interface graphique pour afficher un graphe non orienté avec les nœuds et les couleurs, en utilisant la bibliothèque 'Cairo'.
9. Reproduction des étapes précédentes en ajoutant un graphe et des couleurs dans l'interface graphique.
10. Réalisation de tests pour afficher un graphe non orienté en forme d'anneaux avec n sites.
11. Écriture d'un script en bash permettant de lancer n sites.
12. Ajout d'une fonctionnalité pour que chaque site envoie l'adresse IP des serveurs auxquels il est connecté, et que le serveur principal gère ces adresses pour afficher les connexions.
13. Ajout d'un programme de type orchestrateur pour distribuer les adresses aux sites et créer un anneau ou un réseau complet.
14. Développement d'un programme qui se connecte au serveur principal, à l'orchestrateur et au site concerné, en prenant en compte les adresses reçues par l'orchestrateur.
15. Configuration de l'interface graphique pour gérer toutes les fonctionnalités ajoutées.
16. Réalisation de tests de l'interface graphique sur 2 exemples d'applications distribuées.
17. Amélioration de l'interface graphique pour afficher le maximum d'informations proprement, en prenant en compte le nombre de sites.

2 Technologies utilisées

2.1 Etude des technologies utilisées dans le cadre du projet

Choix du langage de programmation C :
Nous avons utilisé le langage de programmation C pour notre projet car il nous a été imposé dans le cahier des charges. Le C est un langage de programmation de bas niveau qui permet un contrôle fin sur les ressources du système et offre des performances exceptionnelles.

Choix de bibliothèque GTK :
GTK permet de créer facilement une interface graphique complète avec des widgets tels que des boutons, des zones de saisie, des listes, etc. Et de gérer son affichage.
Les signaux de GTK permettent de connecter facilement des fonctions de rappel aux événements des widgets (clic sur un bouton, entrée dans une zone de saisie, etc.)

Choix de la bibliothèque Cairo :
Cairo permet de dessiner des formes et du texte de manière vectorielle efficace. Il est utilisé dans notre interface pour dessiner les cercles et les lignes représentant les connexions réseau.

La combinaison de GTK et Cairo dans notre code, permet de créer une interface graphique complète et interactive, tout en séparant clairement l'affichage vectoriel du graphique de l'interface elle-même. Cela améliore la modularité et la clarté du code.

3 Les Développements Logiciel : fonctionnalités et modules

3.1 Les principaux modules du logiciel développé dans le cadre du projet

Le logiciel développé comprend trois modules clés pour fournir les fonctionnalités requises figure 1 :

1) Une application distribuée, déployée sur plusieurs machines, pour assurer l'évolutivité et le passage à l'échelle. Cette application exécutera en parallèle les différents processus métiers.

2) Une interface graphique permettant de suivre les évaluations et la progression de l'application distribuée. Elle affichera en temps réel l'état de chaque processus pour une visibilité globale.

3) Un module de communication API, basé sur le protocole TCP/IP fiable et efficace, qui aura pour rôle de transmettre l'état courant de chaque processus à l'interface graphique de suivi. Ceci garantira une mise à jour constante de l'affichage en synchronisant les données entre l'application distribuée et l'interface.

L'ensemble de ces modules permettra d'offrir une application fonctionnelle et efficace pour la visualisation de l'état des processus dans le cadre des algorithmes répartis.

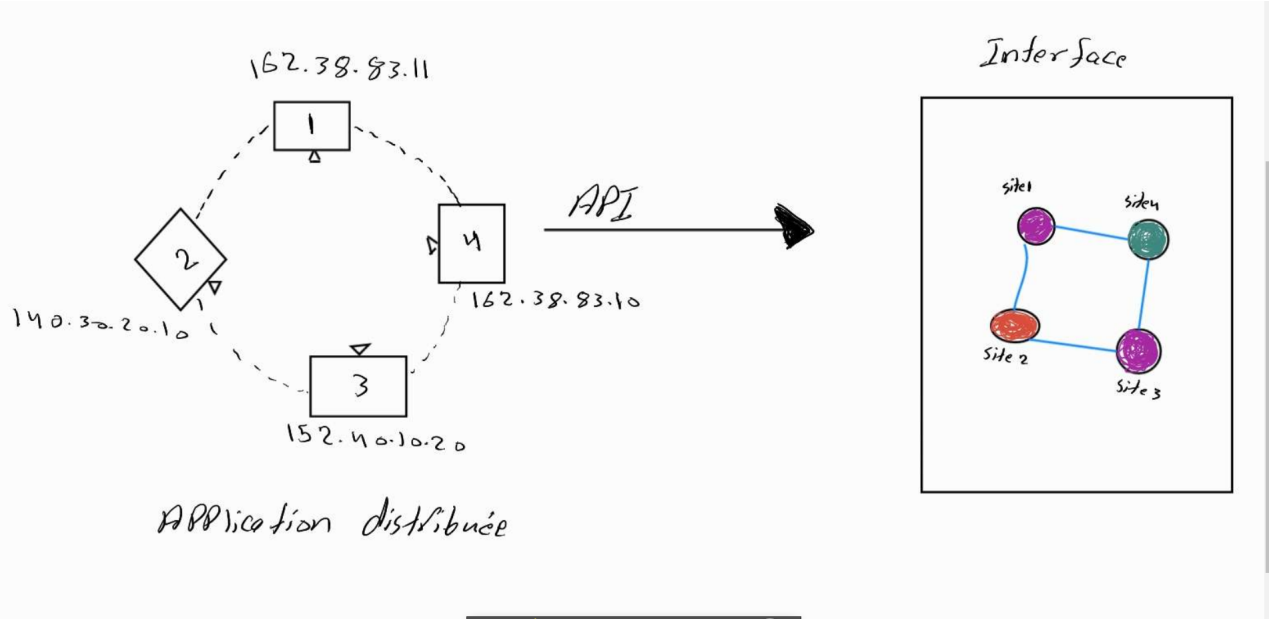

Figure 1 – Graphique de connexion processus / interface

3.2 Les fonctionnalités de l'interface graphique implémentée

L'interface graphique développée fournit les fonctionnalités suivantes :

• Elle affiche pour chaque site surveillé :
- Son adresse (IP et Port) pour identifier de manière unique chaque site
- Un identifiant numérique
- Son état actuel dans l'application distribuée
- Son état de connexion à l'interface : "connecté" ou "déconnecté".

• Une barre de défilement permet d'afficher :
- L'heure de la connexion à l'interface
- L'état actuel et l'état de la connexion à l'interface.

• Une barre de recherche permet de :
- Saisir l'identifiant d'un site spécifique
- Zoomer dessus afin de visualiser rapidement ses informations et les sites auxquels il est connecté.

4 Algorithmes

4.1 Les algorithmes implémentés

algorithme permetant d'afficher les nœuds sur un graphe circulaire

Pour représenter graphiquement les sites connectés de manière organisée, nous avons mis en place un simple algorithme de placement circulaire.

Il calcule les coordonnées polaires (x, y) de chaque nœud du graphe en anneau de la manière suivante :

Tout d'abord, nous définissons :

- center x et center y : coordonnées du centre du cercle
- radius : rayon du cercle, distance entre le centre et les nœuds
- k : indice du nœud, avec k allant de 0 à n-1 et n le nombre total de nœuds

angle-step représente l'angle entre deux nœuds consécutifs. Il est calculé avec : angle-step = 2π / n
où 2π est la circonférence totale et n le nombre de nœuds.

Les coordonnées (x,y) du nœud k s'obtiennent ainsi :

x = center x + radius * cos(k * angle-step)
y = center-y + radius sin(k * angle-step)
Ceci positionne le nœud k à l'angle k * angle-step par rapport au centre.

L'algorithme utilise des fonctions trigonométriques de base pour attribuer aux nœuds des coordonnées suivant leur indice leur permettant d'être représentés de manière organisée sur un graphe circulaire.

5 Discussion des résultats

5.1 Préparation pour tester l'interface

Dans le but de tester notre interface graphique, nous avons implémenté 2 applications distribuées :

Une application en anneau, où chaque site est relié uniquement à ses 2 voisins. Cette topologie permet de tester les fonctionnalités de visualisation circulaire de notre interface.

Une application en réseau complet, où chaque site est connecté à tous les autres. Cette topologie plus complexe permet de vérifier que l'interface affiche correctement des graphes denses.

Pour construire ces applications distribuées, nous avons développé Un programme d'orchestration fonctionnant comme un serveur central, chargé de Connecter les sites entre eux selon la topologie choisie (anneau ou complète).

Fonctionnalité de l'orchestration

Au démarrage, le programme crée un socket d'écoute configuré pour recevoir les connexions entrantes des sites clients. Une fois prêt, le programme attend la connexion des différents sites clients. Pour chaque connexion acceptée, il ajoute l'adresse du site client à une liste de adresses.

Ensuite, selon le choix de l'utilisateur (anneau ou complet), le programme envoie la configuration appropriée à chaque site client :

Pour un réseau en anneau : Il envoie à chaque site client de id i l'adresse du site client précédent qui a l'id (i-1). Pour le premier site client, il envoie les informations du dernier site afin de créer l'anneau.

Pour un réseau complet : Il envoie l'adresse de chaque site client à tous les sites clients.

5.2 Limites de l'interface

-L'interface peut suivre l'état d'applications composées d'un maximum de 256 sites.

-Plus le nombre de sites affichés est élevé, plus il devient difficile de voir clairement les connexions entre les sites, en particulier dans les réseaux complets comme montre la figure6.

-À mesure que le nombre de sites affichés augmente, il devient de plus en plus difficile d'avoir une visibilité claire sur les indices et l'état de chaque site. L'affichage devient encombré et les détails peuvent passer inaperçus comme dans la figure 8

solutions apportées :

- Pour le problème de visualisation des connexions entre les sites, nous avons ajouté la fonctionnalité de pouvoir zoomer sur un site en particulier, afin de visualiser rapidement les sites auxquels il est connecté.

- Pour résoudre le problème de visibilité des indices et des états , nous avons implémenté une barre de défilement qui permet d'afficher clairement ces informations, même lorsqu'un grand nombre de sites sont présents.

Améliorations pouvant être apportées à l'interface : Pour que l'interface puisse suivre l'état d'une application composée de plus de 256 sites, une fonctionnalité de zoom avant/arrière pour l'affichage de l'ensemble du réseau peut être implémentée.

5.3 Bancs d'essais utilisés pour les tests du logiciel

Après avoir préparé tous les éléments nécessaires pour tester notre interface, nous avons effectué plusieurs tests sur des applications locales. Nous avons ensuite exécuté ces mêmes tests sur des applications distribuées.

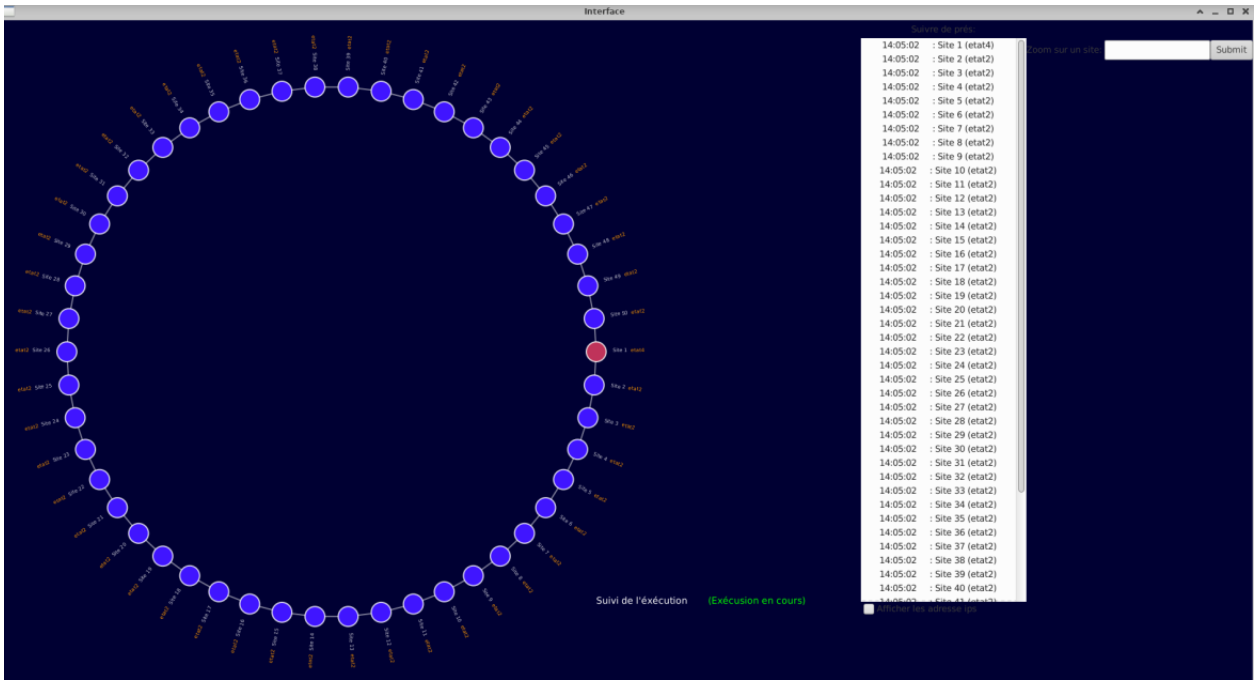

Figure 2 – Réseau en Anneau

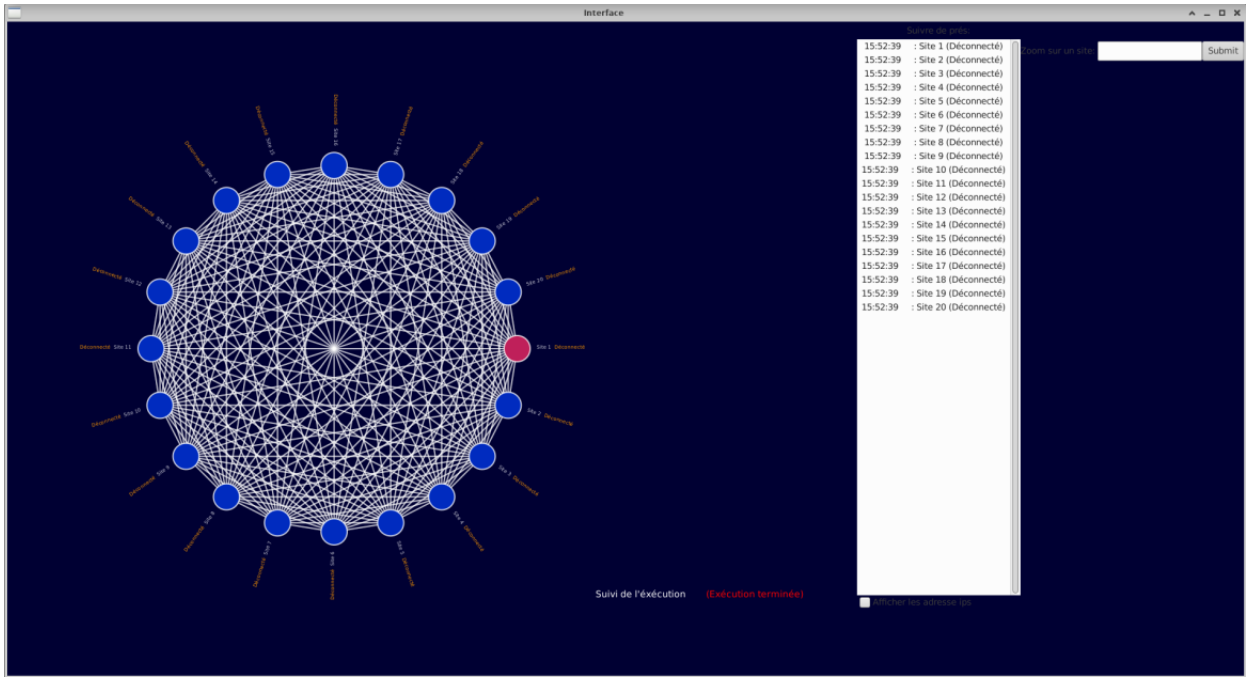

Figure 3 – Réseau complet de 20 sites

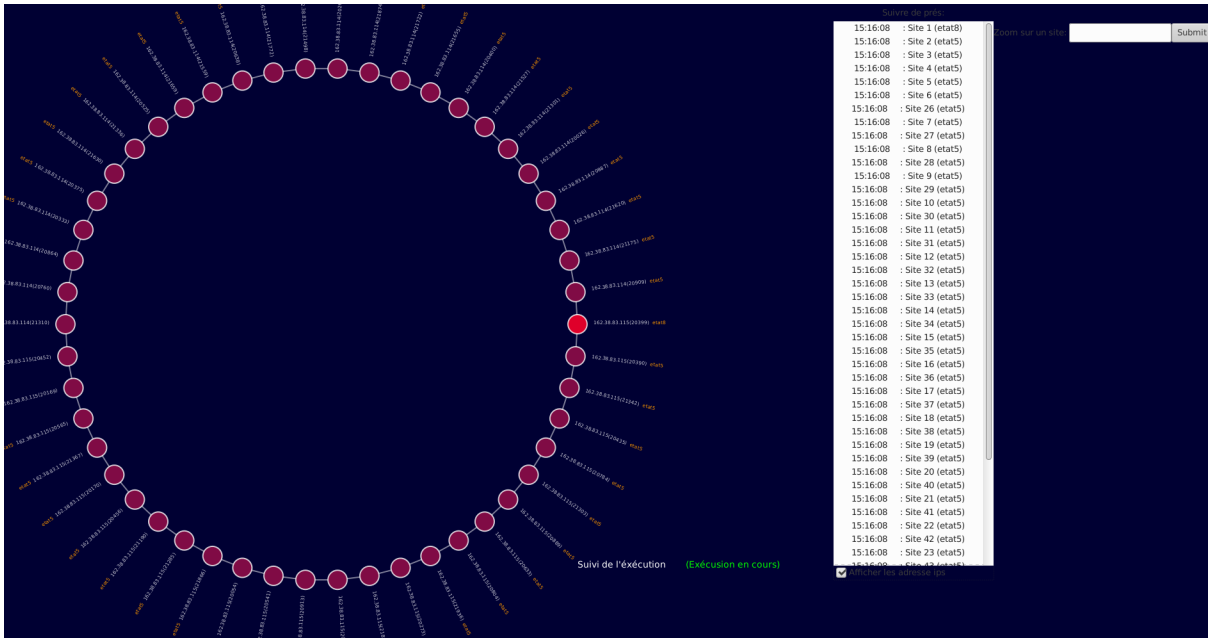

Figure 4 – Anneau de 50 sites

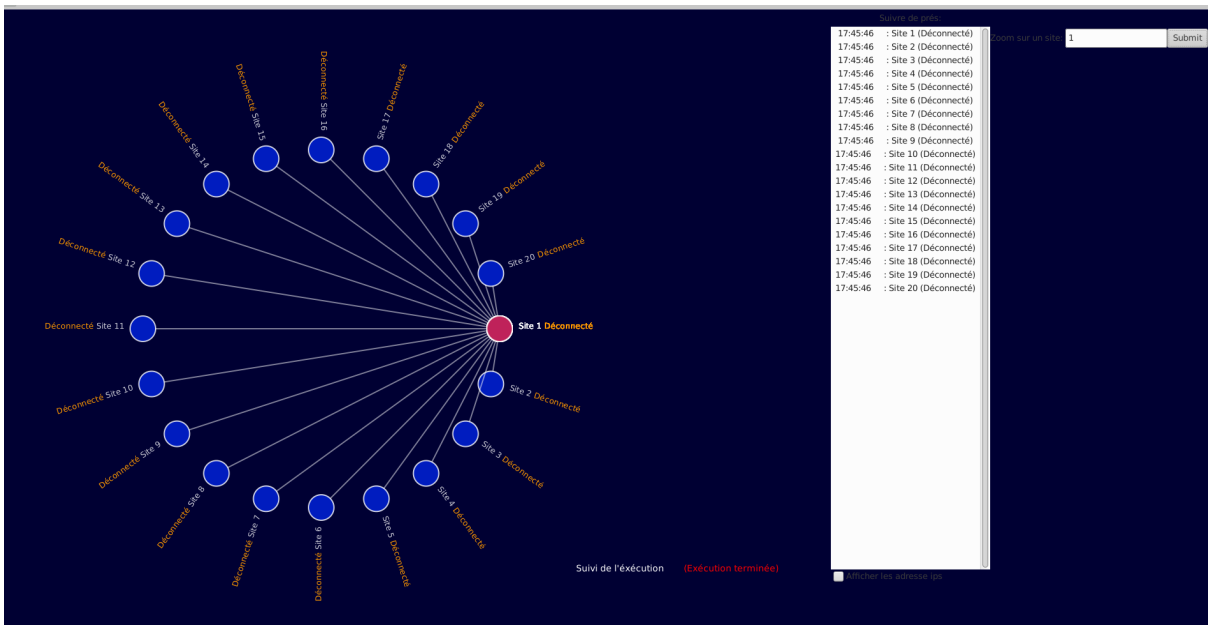

Figure 5 – Zoom sur le site n°1 dans un réseau complet

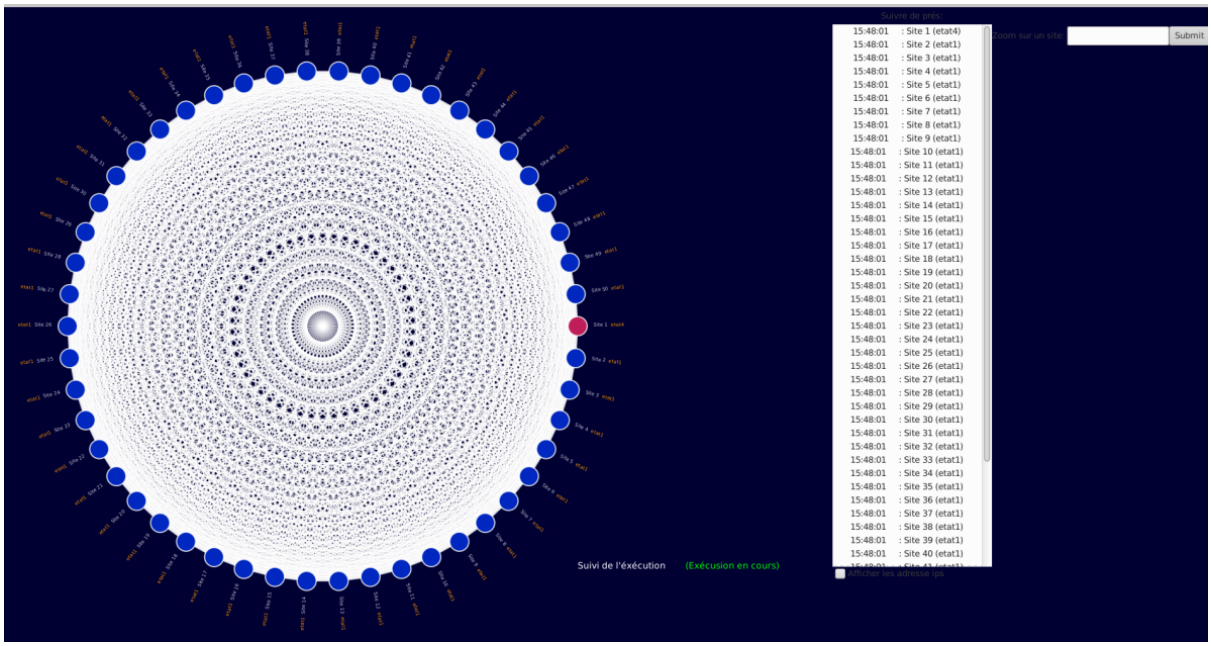

Figure 6 – Réseau complet de 50 sites

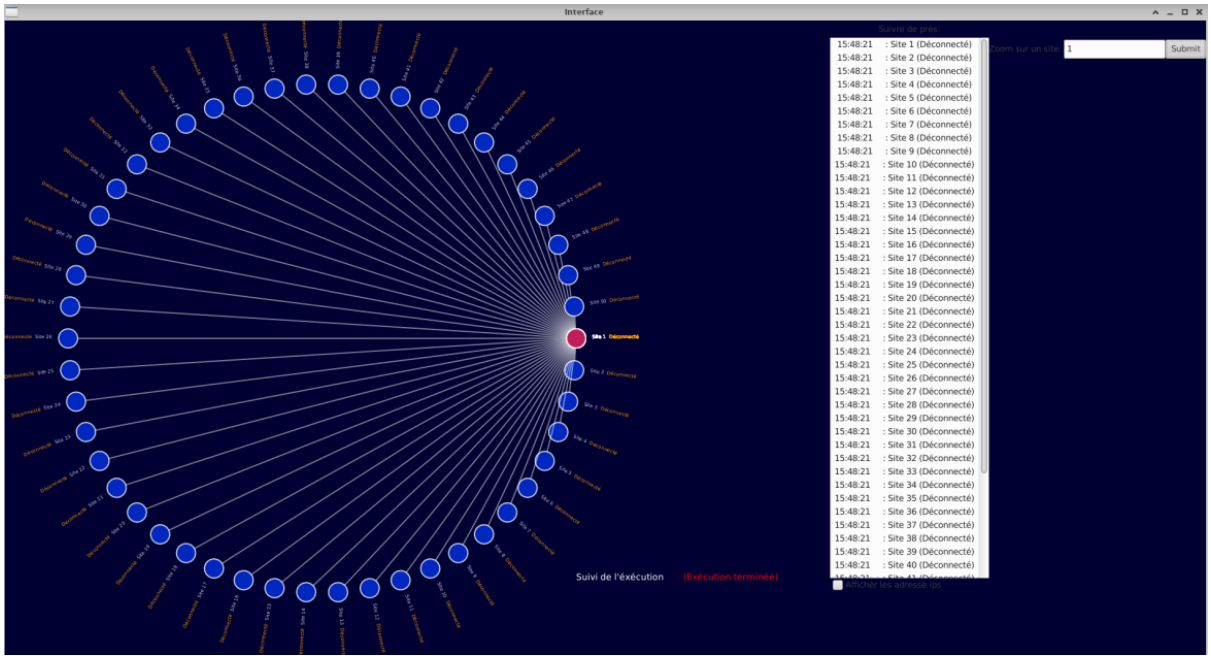

Figure 7 – Zoom site n°1 de Réseau complet

En examinant la figure 6, nous avons constaté que les connexions n'étaient pas claires. Cependant, la figure 7, qui représente la même application, met en évidence l'importance de la fonctionnalité de zoom sur un seul site.

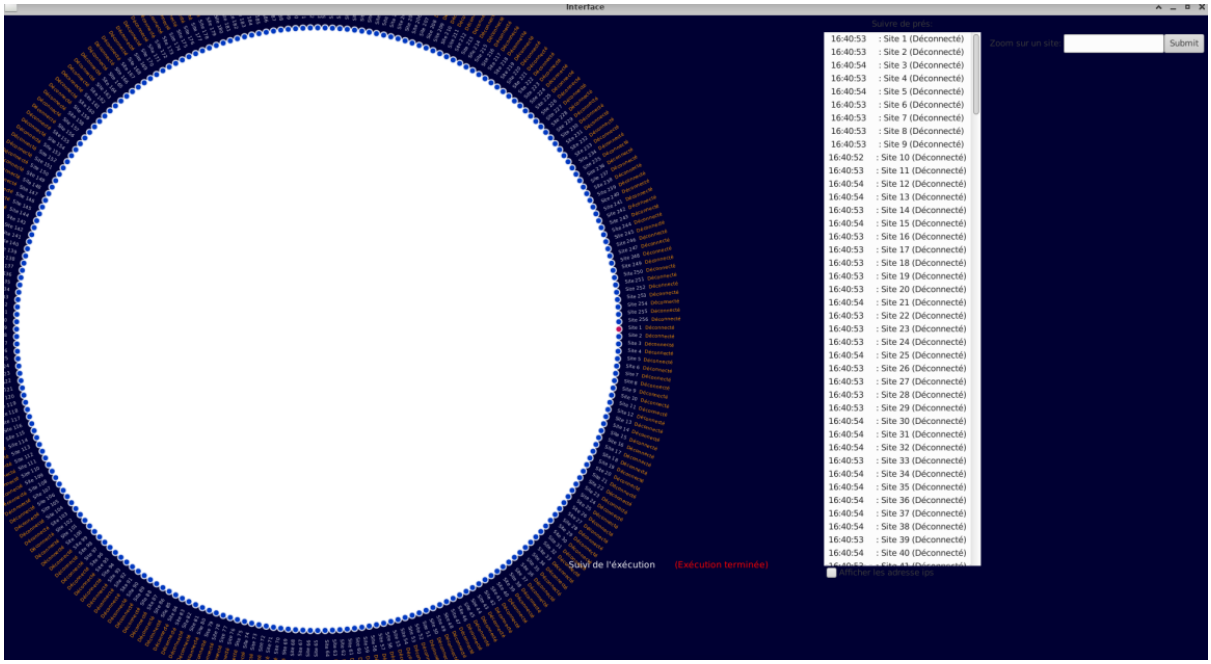

Figure 8 – Réseau complet de 256 sites

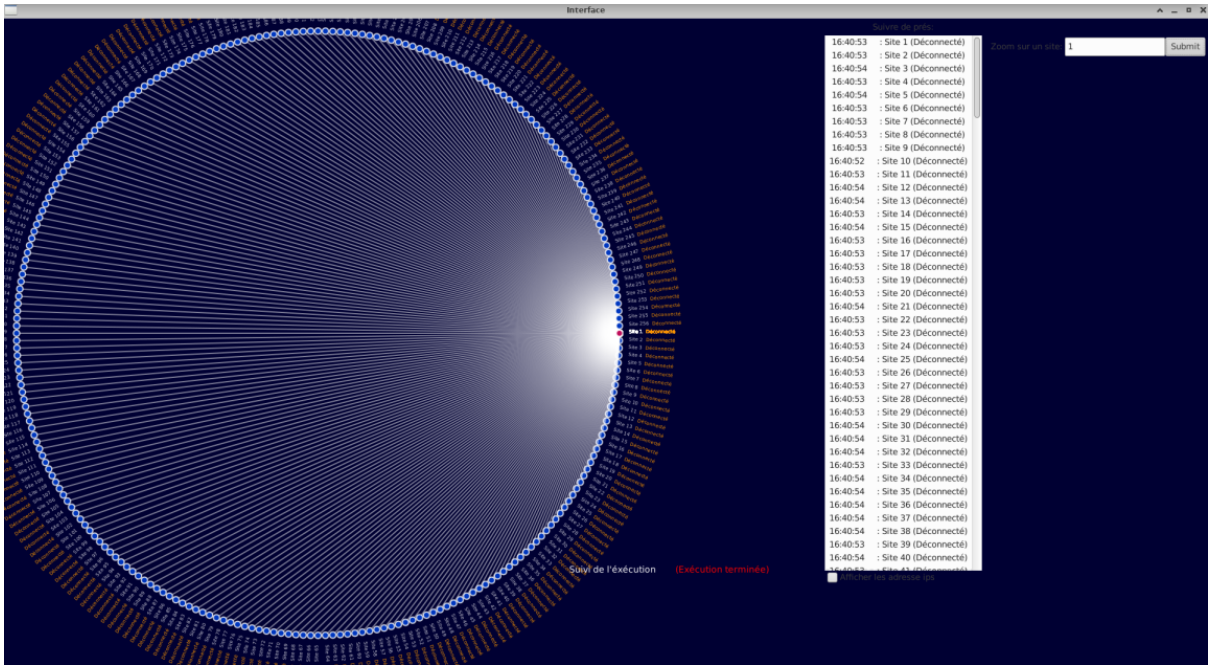

Figure 9 – Zoom site n°1 de Réseau complet

La figure 8 montre l'importance de la barre de défilement qui affiche bien les indices et les états des sites.

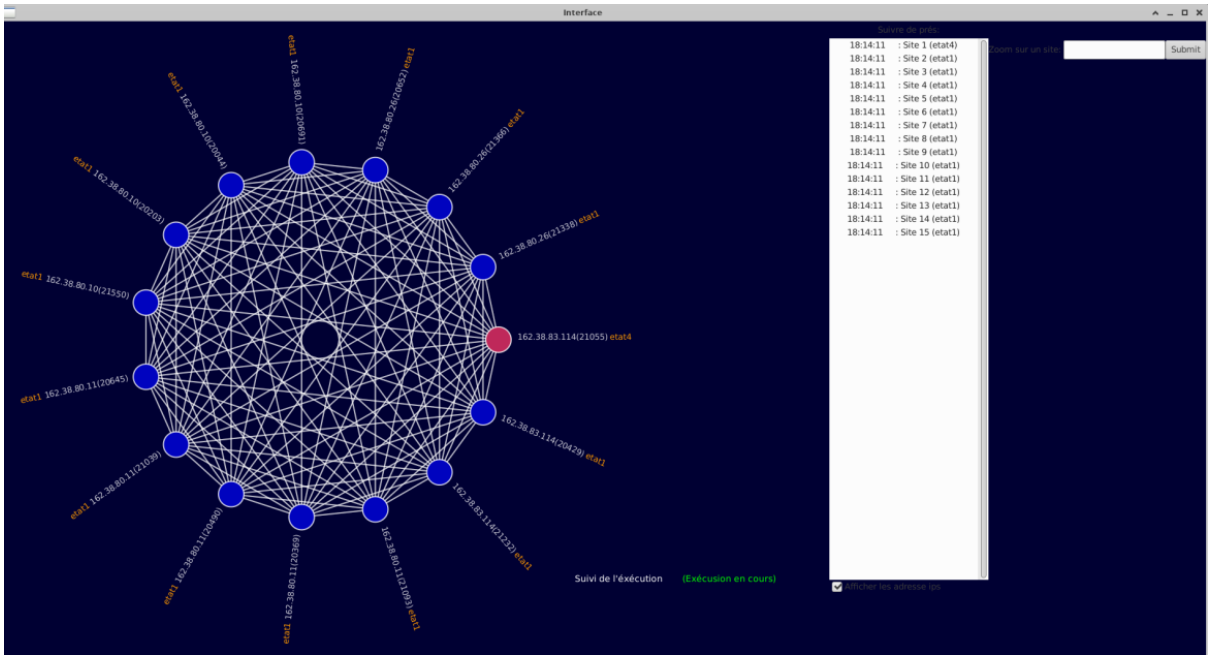

Figure 10 – Réseau distribué avec quatre machines.

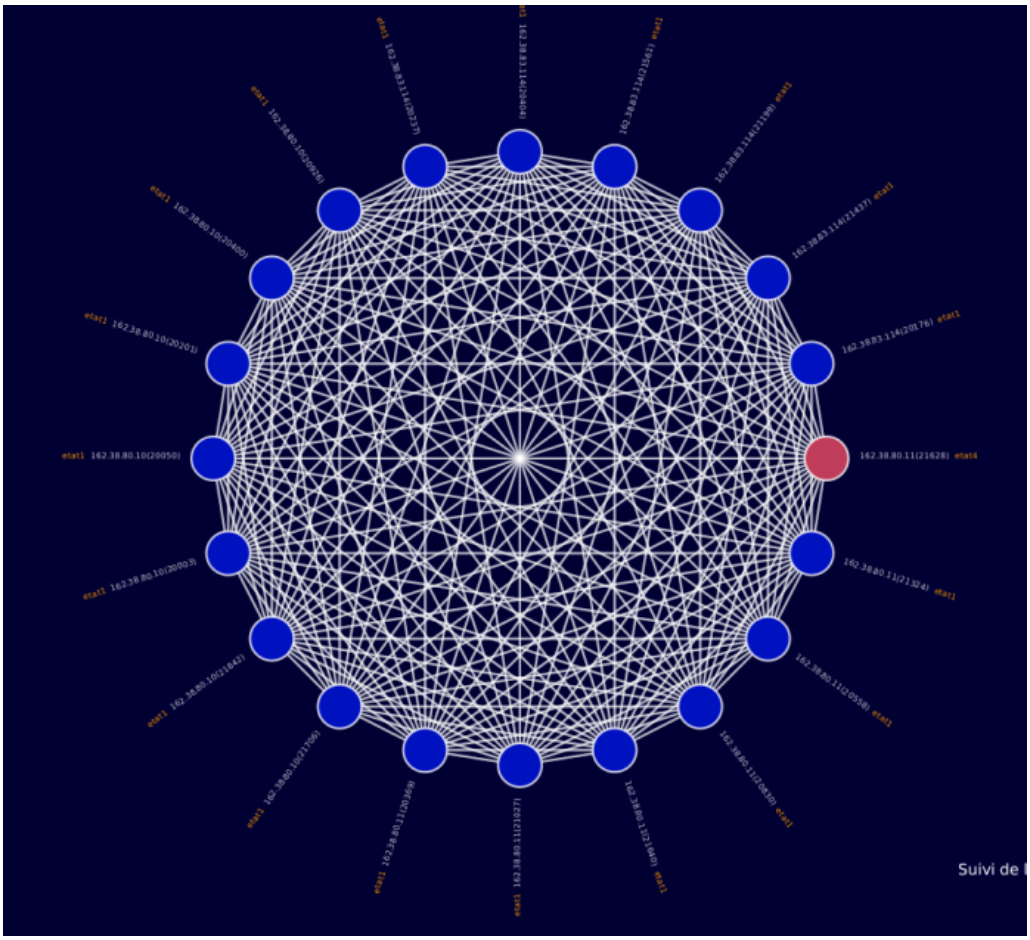

Figure 11 – Réseau distribué avec trois machines

Nous observons que dans les applications distribuées, les adresses IP des machines étaient différentes, comme le montrent les figures 10 et 11.

6 Gestion du Projet

6.1 Gestion de projet et documents de planification

DIAGRAMME DE GANTT :

Durant le projet, nous avons choisi d'effectuer un diagramme de Gantt, figure 12, qui nous a permis de définir les tâches à accomplir, et de plannifier tout le travail à effectuer dans le délai imparti.

GESTION DE PROJET :

Nous avons adopté une organisation bien structurée pour mener à bien notre projet. Nous avons programmé des réunions hebdomadaires pour lesquelles nous nous sommes réunis deux demi-journées par semaine, le mardi et le vendredi après-midi, à la bibliothèque de la faculté. Ces réunions nous ont permis de travailler ensemble, de discuter de l'état d'avancement et des difficultés rencontrées, ainsi que d'échanger des idées et des recherches.

Pour les échanges en dehors des réunions, nous avons utilisé Discord, une plateforme de communication en ligne flexible et adaptée au travail collaboratif. Cette solution nous a permis de communiquer rapidement sur les différentes tâches à réaliser, de partager des ressources et de rester en contact de manière permanente.

Nous avons également utilisé GitHub pour le partage et la synchronisation de notre code. Cette plateforme nous a permis de travailler simultanément sur le projet, de fusionner facilement nos contributions et de garder une trace détaillée de l'historique de développement.

Lien du dépot Github : `https://github.com/AliMohammed232000/S6New`

En somme, cette organisation hebdomadaire combinant des réunions physiques et des échanges en ligne a prouvé son efficacité pour assurer la coordination nécessaire à la bonne réalisation de notre projet tout en conservant une certaine flexibilité.

6.2 Changements majeurs en cours de projet

Au niveau du choix des technologies de développement :
Initialement, nous avions développé l'interface graphique de ce programme avec la bibliothèque GTK et le générateur d'interface GLAD. Cependant, nous nous sommes rendu compte que GLAD ne correspondait pas parfaitement à nos besoins :

- Difficultés à modifier l'interface après coup

- Code généré parfois verbeux et non optimal

- Limites pour des interfaces complexes et personnalisées

Nous avons donc décidé de nous passer de GLAD pour développer directement notre interface graphique avec GTK uniquement, en codant manuellement les widgets et leur disposition.

Cette approche nous a permis :

- Plus de flexibilité et de personnalisation

- Un meilleur contrôle et une meilleure compréhension du code

- Une interface plus facilement maintenable et extensible

Bien que GLAD nous ait fait gagner du temps au début, nous avons constaté que pour correspondre parfaitement à nos besoins spécifiques, développer entièrement notre interface graphique avec GTK s'est avéré une meilleure solution.

Au niveau des membres du groupe :
Il y a eu une diminution du nombre de participants au projet suite au retrait de 2 membres.
Cela a :
- Réduit les ressources disponibles
- Nécessité de redistribuer certaines tâches
- Fait perdre un peu de diversité

Les membres restants ont dû fournir un effort supplémentaire pour compenser le retrait des 2 participants.

Malgré tout, le projet a pu aboutir grâce à l'implication renforcée des membres restants.

7 Le guide d'utilisation

Voici le guide d'utilisation pour bien utiliser l'interface, il est conseillé de commencer par la création du fichier de configuration de l'interface graphique puis l'implémentation de la connexion dans votre programme à l'aide du fichier "interface.h" fournis. Prenez soin de bien choisir votre au nombre d'états et aux couleurs associées, ce qui facilitera la compression durant le fonctionnement de votre programme.

7.1 Initialisation de l'interface : Creation du fichier de configuration

Avant de connecter votre programme, ou de lancer l'interface, il faut d'abord configurer le fichier .txt placé dans un dossier "files", s'il n'existe pas, créé-le.

Ensuite entré ses informations dans le fichier : 1.Ajouter le nombre de sites lancé 2.Nombre d'états 3. Choisir si les couleurs associées aux états sont générées (0) ou choisies par vous (1). 4. n lignes d'état avec ou sans le numéro des couleurs au format, id-etat : nom-état valeur-de-couleur.

Si vous avez choisi d'initialiser les couleurs par vous-même, il faudra ajouter 3 chiffres allant de 0.0 à 1.0.

Pour le noir : 0.0 0.0 0.0 ; Pour le blanc : 1.0 1.0 1.0

Exemple avec choix des couleurs :
#Nombre de sites
4
#Nombre d'états
4
#Choix(O ou 1) 1
#Liste d'états
5 :attente 0.1 1.0 0.0
1 :fermer 0.1 0.0 0.0
3 :traitement 0.1 1.0 1.0
4 :intialisation 0.3 1.0 0.3

Exemple avec couleurs générées :
#Nombre de sites
4
#Nombre d'états
4
#Choix(O ou 1) 0
#Liste d'états
5 :attente
1 :fermer

3 :traitement
4 :intialisation

7.2 Connection du processus à l'interface

Après avoir configuré le fichier, vous allez devoir implémenter l'api dans votre programme.

Veillez bien à suivre les étapes dans l'ordre dans votre programme, sinon il ne fonctionnera pas.

7.2.1 Initialisation de la connexion

Premièrement, il faut inclure le fichier interface.h dans votre programme.Inclure "interface.h"

Deuxiemement, il faut initialiser un struct qui contiendra l'adresse de la machine où est situé le MainServer, et un port que vous choisiraient.

Exemple :struct Server inter ; inter.addr = "127.0.0.1" ;inter.port = 3000 ;

7.2.2 Etablir la connexion à l'interface

Ensuite, pour établir la connexion avec le MainServer, il faudra ajouter cette fonction qui prend en parametre la struct précédemment définie et qui retour un int de valeur 1 si la connexion est établie, -1 sinon.

int connectConnectionInterface(struct Server* inter) ;

7.2.3 Envoyer de l'id du processus

Attention : Assurez-vous bien que la connexion au MainServeur est bien établie.

Apres avoir établi la connexion, pour que l'interface puisse identifier le processus courant qui lui enverra les données, il faudra lui envoyer un identifiant, ainsi vous devez ajouter cette fonction à la suite, qui prend en paramètre l'adresse de la struct et le numéro d'identifiant et retourne un entier, -1 si erreur lors de la connexion, 1 sinon.

int sendId(struct Server* inter, int id) ;

Remarque : Se connecter à l'interface et envoyer l'identifiant du site sont les deux premières étapes que votre programme doit accomplir.

7.2.4 Envoyer d'une stucture de donnée

Pour indiquer à l'interface quel processus est connecté à celui-là via une connexion TCP, vous devez envoyer les sockaddr_in de toutes les connexions une par une à l'aide de cette fonction :

int sendStruct(struct Interface* inter, struct sockaddr_in id) ;

Ainsi vous devrez ajouter cette fonction qui prend en paramètre l'adresse de la struct Server et une structure de type sockaddr_in, -1 si erreur lors de l'envoi du message, 1 sinon.

7.2.5 Envoyer un état à l'interface

Maintenant que la connexion est établie avec le MainServer, Vous pouvez ajouter une ou plusieurs fonctions "sendStatus" à des points-clés où votre programme change d'états, et qui enverra une valeur de type int que vous avez précédemment définie dans le fichier de configuration.

La fonction prend en paramètre l'adresse de la struct et le code précédemment initialisé associé à l'état de votre programme et retourne un entier, 1 si le code est bien reçu, -1 sinon.

int sendStatus(struct Server* inter, int code) ;

7.2.6 Fermeture de la connexion

int closeConnectionInterface(struct Server* inter) ;

Pour fermer la connexion avec l'interface, il faut ajouter cette fonction qui prend en paramètre l'adresse de la struct initialisée et qui retourne un entier, -1 si erreur lors de la connexion 1 sinon.

7.3 Visualisation

Maintenant, nous allons voir comment utiliser l'interface graphique pour visualiser les processus.

7.3.1 Compilation du programme

Pour compliler il faut utiliser le fichier compile.script
commande : bash compile.script

7.3.2 Lancement du programme

Pour ne pas avoir de problème, prenez bien soin de lancer en premier l'interface grâce à la commande "/MainServeur Port-utiliser chemin-fichier-config" puis votre programme.

7.3.3 Zoom sur un processus

Pour pouvoir zoomer sur l'activité d'un processus en particulier, vous pouvez en remplissant la barre de saisie située en haut à droite, le numéro du processus puis appuyé sur submit pour afficher son activité.

8 Bilan et Conclusions

Les objectifs principaux de ce projet ont été atteints :

• Les réseaux en anneau et réseaux complets fonctionnent correctement avec jusqu'à 256 nœuds et les états peuvent être visualisés via notre interface graphique.

• Différentes fonctionnalités ont été ajoutées à l'interface pour améliorer la visualisation des connexions : sélection de nœuds particuliers, affichage des adresses IP au choix, etc.

• Le projet a été testé avec succès sur un réseau distribué de plusieurs machines sans rencontrer de problèmes majeurs.

Parmi les améliorations possibles à apporter :

• Implémenter un zoom complet du réseau pour mieux visualiser l'ensemble des nœuds.

Au niveau du travail d'équipe :

• La collaboration avec les autres membres de l'équipe a été bénéfique et a permis de surmonter efficacement les difficultés rencontrées.

• L'encadrement de Madame Hinde Bouziane a été précieux, en nous guidant dans la bonne direction et en nous encourageant tout au long du projet.

En conclusion, ce projet a atteint ses buts principaux et nous a permis d'acquérir de nombreuses compétences en algorithmie réseau, programmation C, architecture client-serveur et travail d'équipe. Il reste cependant des améliorations possibles que nous pourrions apporter.

9 Bibliographie

[1] **Cairo :** https ://www.cairographics.org/documentation/

[2] **GTK :** https :https ://www.gtk.org/docs/

10 Annexes

Tache	Date debut	Durée (semaine)	Date final
Implémenter un processus en langage C qui change d'état et la création d'une interface qui permet d'afficher l'état d'un processus	18/01/2023	1	25/01/2023
La création d'un site qui change son état en boucle et envoie périodiquement cet état à l'interface	31/01/2023	1	07/02/2023
Interconnection de plusieurs sites, chacun modifiant périodiquement son état et envoyant cet état au main serveur	03/02/2023	1	10/02/2023
Le main serveur reçoit plusieurs connexion de site avec deux états distincts, "en attente" et "actif", qui envoient périodiquement leur état au main serveur	07/02/2023	1	14/02/2023
Réaliser les même étapes d'avant avec les graphes et les couleurs,Implémenter le graphe avec n sites	24/02/2023	1	03/03/2023
La créationun programme de orchestrateur , un programme site et Script de lancement,implémentation de réseau complet+l'anneau	22/03/2023	2	05/04/2023
Amélioration d'interface et la création de l'API	21/04/2023	1	28/04/2023
Ecriture du guide et le rapport	07/05/2023	1	14/05/2023

Figure 12 – Diagramme de Gantt