NOMS DE MEMBRES DU GROUPE

- **Pakou Yanou Rachel Emmanuel**
- **Tchio Tsopfack Bryan**
- **Tioning Tchamdeu Daniella**
- **Ngobo Masieta Clara**
- **Noah David Hervé**
- **Woupala Loïc**
- **Dongmo Jennifer**
- **Fossi Tadjuidje Calix**

Dans notre projet nous avons voulut utiliser

Contexte et justification

Cette application est un programme code en langage C et en utilisant SDLC pour la création de son interface graphique, son objectif principal est de permettre à plusieurs utilisateurs d'un même réseau de communiquer entre eux

Objectif du projet

-Communication P2P dans un réseau

Portées et limites

1. Portées

-Permet de communiquer avec n'importe qui dans le réseau qu'utilise l'application

2. Limites

L'application est :

-Très peu sécurisé

-Uniquement sur Windows

-Incapable de savoir les utilisateurs en ligne

-Incapable de vérifier qu'un message est parvenu a son destinataire

Public Cible

-Des personnes qui souhaitent discuter dans un réseau sans avoir besoin de dépenser leur forfait internet

-Spécifications des besoins

Conception détaillé

Le projet sera constitué d'un dossier

-Src : qui contiendra le fichier principal avec dll de sdl

-Network : est un dossier qui aura les fonctions utiliser pour écrire et recevoir des messages dans le réseau

-Interface : qui est le dossier qui va contenir les fonctions pour l'interface de l'application

-Docs : qui vont contenir les documents utiles du projets

Gestion de projet

-Le projet a été repartir en plusieurs sous partie

-La partie réseau

-La partie visuelle

-La partie logique

Conception des différents tests

-Pour vérifier que l'application fonctionne nous avons procédé a différentes tests :

-S'envoyer un message a soi même pour voir le comportement

-Envoyer u message à un groupe de personnes

-Vérifier comment l'interface réagi selon la résolution de l'appareil

Conclusion

En résumé, nous concluons que pour que notre projet soit optimale et fonctionnel pour communiquer avec d'autres udp, certes il n'es pas le meilleur pour la messagerie mais nous l'utiliserons pour faciliter l'interaction de plusieurs utilisateurs dans le groupe

Depuis quelques semaine nous avons essayer de mettre notre projet sur GITHUB , chaque membre a mis son compte git

Nous avons poursuivis sur l'application faire quelques modifications , dans le programme C

FONCTIONS PERSONNALISÉES DU PROJET (ALGorithme LAN P2P Messenger)

1. load_or_create_uuid(char* uuid) Rôle : Charge l'UUID depuis le fichier "uuid.txt" s'il existe. Sinon, génère un nouvel UUID aléatoire et le sauvegarde dans le fichier.

2. create_pseudo_txt() Rôle : Crée le fichier "pseudo.txt" s'il n'existe pas.

3. save_pseudo(const char* pseudo) Rôle : Sauvegarde le pseudo actuel dans le fichier "pseudo.txt".

4. pseudo_from_uuid(const char* base, const char* uuid, char* final_pseudo) Rôle : Génère un pseudo unique en ajoutant un suffixe hexadécimal déterministe basé sur l'UUID.

5. ask_pseudo() Rôle : Demande le pseudo à l'utilisateur au démarrage (si absent), génère la version unique et l'affiche.

6. peer_exists(const char* uuid) Rôle : Vérifie si un utilisateur (peer) est déjà enregistré dans la liste des peers.

7. send_broadcast(const char* uuid) Rôle : Envoie un message UDP de type "HELLO" en broadcast pour annoncer la présence de l'utilisateur sur le réseau.

8. udp_listener(LPVOID arg) Rôle : Écoute les messages UDP entrants (HELLO), détecte les nouveaux peers et les enregistre.

9. send_tcp(const char* ip, const char* msg) Rôle : Envoie un message TCP à une adresse IP donnée et attend une confirmation “ACK”.

10. tcp_listener(LPVOID arg) Rôle : Serveur TCP local qui reçoit les messages entrants, les enregistre dans un fichier et renvoie “ACK”.

11. add_unicast(const char* ip, const char* msg) Rôle : Ajoute un message dans la file d’attente des messages à envoyer (avec gestion de confirmation).

12. cleanup_confirmed() Rôle : Supprime de la file les messages déjà confirmés pour libérer la mémoire.

13. sender_thread(LPVOID arg) Rôle : Thread responsable de : - envoyer les broadcasts périodiques - tenter d’envoyer les messages en attente - gérer les confirmations

14. print_menu() Rôle : Affiche le menu principal de l’application.

15. menu_loop() Rôle : Gère l’interface utilisateur et les actions : consultation messages, liste utilisateurs, changement pseudo, envoi message.

16. main() Rôle : Point d’entrée du programme. Initialise Winsock, les

verrous, les threads, puis lance la boucle du menu.