PR6 – Programmation réseaux
Projet Mégaphone

Le but de ce projet est d'implémenter un serveur et un client pour le protocole *Mégaphone* décrit ci-après.

I) Le protocole *Mégaphone*

Le protocole *Mégaphone* est un protocole de communication de type client/serveur. Des utilisateurs se connectent à un serveur avec un client et peuvent ensuite suivre des fils de discussions, en créer de nouveaux ou enrichir des fils existants en postant un nouveau message (*billet*), s'abonner à un fil pour recevoir des notifications.
Plus précisément, un utilisateur peut s'inscrire auprès du serveur et une fois inscrit, il doit pouvoir faire les actions suivantes :

- poster un message ou un fichier sur un nouveau fil,
- poster un message ou un fichier sur un fil existant,
- demander la liste des n derniers billets sur chaque fil,
- demander les n derniers billets d'un fil donné,
- s'abonner à un fil et recevoir les notifications de ce fil.

Le serveur doit répondre aux demandes du client et également envoyer des notifications.
Dans un premier temps, les échanges entre les clients et le serveur sont décrits de façon globale. Le format exact des messages du client et du serveur sont ensuite décrits précisément.

a) Les échanges client/serveur

L'inscription Lors de sa première connexion au serveur, l'utilisateur doit s'inscrire auprès du serveur. Les étapes des échanges entre le client et le serveur sont les suivantes :

1. le client se connecte au serveur et envoie une demande d'inscription en donnant le pseudo de l'utilisateur.
2. Si le serveur accepte, il attribue un identifiant unique à l'utilisateur, l'envoie au client et ajoute l'identifiant et le pseudo à la liste des inscrits.
   Le serveur refuse l'inscription s'il manque le pseudo ou si celui-ci est trop court, donc si le message est mal formé.
3. Le client récupère l'identifiant de l'utilisateur.
4. Le client et le serveur terminent la connexion.

Poster un billet S'il est inscrit, l'utilisateur peut demander à poster un billet sur un fil existant ou sur un nouveau fil. Les étapes des échanges entre le client et le serveur sont les suivantes :

1. le client se connecte au serveur et envoie au serveur un billet à ajouter en précisant l'identifiant de l'utilisateur et, si nécessaire, le numéro de fil auquel ajouter le billet.
2. Le serveur vérifie que l'utilisateur est bien inscrit et, si c'est le cas, ajoute le billet au fil (qu'il crée si nécessaire) puis envoie un message au client pour lui signifier que l'ajout a été fait.
3. Le client réceptionne la réponse du serveur.
4. Le client et le serveur terminent la connexion.

Demander la liste des n derniers billets S'il est inscrit, l'utilisateur peut demander la liste des n derniers billets sur le fil numéro f. Les étapes des échanges entre le client et le serveur sont les suivantes :

1. le client se connecte au serveur et envoie au serveur une demande de la liste des n derniers billets sur le fil numéro f.
2. Le serveur vérifie que l'utilisateur est bien inscrit et, si c'est le cas, envoie un premier message annonçant le nombre de messages m qu'il va envoyer au client. Puis, il envoie m messages au client contenant chacun un billet et des informations sur ce billet ($m < n$ s'il y a moins de n messages sur le fil).
3. Le client réceptionne les $m + 1$ messages du serveur.
4. Le client et le serveur terminent la connexion.

S'abonner à un fil S'il est inscrit, l'utilisateur peut demander à s'abonner à un fil spécifique. Les étapes des échanges entre le client et le serveur sont les suivantes :

1. le client se connecte au serveur et envoie au serveur une demande d'abonnement à un fil donné.
2. Le serveur vérifie que l'utilisateur est bien inscrit et, si c'est le cas, abonne l'utilisateur au fil. Il répond au client en lui envoyant l'adresse d'abonnement (adresse de multidiffusion).
3. Le client enregistre l'adresse d'abonnement envoyée par le serveur.
4. Le client et le serveur terminent la connexion.
5. Le client s'abonne à l'adresse reçue et fait ce qu'il faut pour recevoir les messages diffusés sur cette adresse en parallèle du reste de ses activités.

Les notifications Lorsque l'utilisateur demande à s'abonner à un fil f :

- soit le fil f a déjà des abonnés et il existe donc une adresse de multidiffusion associée au fil f. Le serveur envoie au client cette adresse,
- soit le fil f n'a pas encore d'abonné, et le serveur attribue une nouvelle adresse de multidiffusion dédiée au fil f et l'envoie au client.

A intervalles de temps réguliers, sur chaque adresse de multidiffusion associée à un fil, le serveur envoie des notifications des derniers billets publiés sur le fil, *i.e.* ceux qui ont été publiés pendant le dernier intervalle de temps. Ce sera à vous de choisir la valeur de l'intervalle de temps qui ne doit pas être trop court (sinon messages multidiffusés trop fréquents).
Le client de l'utilisateur abonné à un ou plusieurs fils doit recevoir ces notifications **en parallèle** de ses autres activités.

Ajouter un fichier Il est possible d'ajouter un fichier qui fait moins de 2^{25} octets ($\simeq 33,5$ Mo) à un fil f afin qu'il puisse être téléchargé par les utilisateurs. L'utilisateur peut donc demander à ajouter au fil f un fichier qu'il transmet au serveur. Le serveur stocke alors le fichier et ajoute au fil f un billet tel que le texte du billet ne contient que le nom du fichier et sa taille.
Les étapes des échanges entre le client et le serveur sont les suivantes :

1. le client se connecte au serveur et envoie au serveur une demande de transfert d'un fichier à ajouter à un fil f.
2. Le serveur répond en donnant un numéro de port sur lequel il recevra les données du fichier transféré en UDP.
3. Le client et le serveur se déconnecte ensuite.

4. Le client transfert alors le fichier en UDP sur le port donné et le serveur reçoit et stocke le fichier, puis ajoute au fil f, un billet contenant uniquement le nom du fichier et sa taille (sous forme de chaîne de caractères).

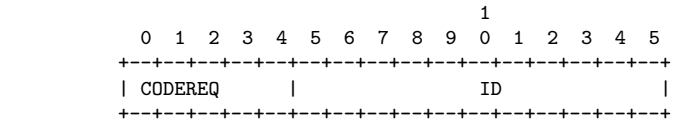
Attention, le serveur ne doit pas attendre indéfiniment le transfert. S'il ne reçoit rien après quelques secondes, il termine son attente et le transfert est annulé.

Télécharger un fichier Si un fichier apparaît sur un fil, il peut être téléchargé par les utilisateurs le demandant.
Les étapes des échanges entre le client et le serveur sont les suivantes :

1. le client se connecte au serveur et envoie au serveur une demande de téléchargement d'un fichier sur un fil f. Il fournit au serveur un numéro de port p, sur lequel il recevra en UDP, les données du fichier demandé.
2. Le serveur accepte la demande.
3. Le client et le serveur se déconnecte ensuite.
4. Le serveur envoie en UDP sur le port p du client les données du fichier.

b) Les messages du client

Les messages du client débuteront tous par l'entête suivant :



où

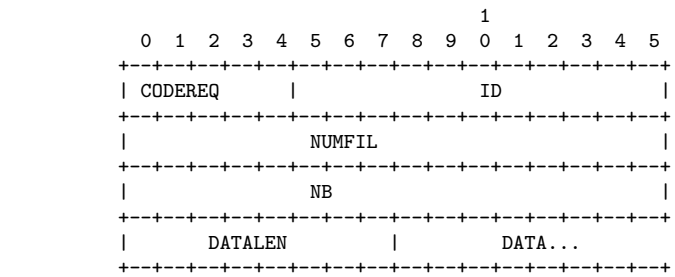
- `CODEREQ` correspond au code de la requête du client,
- `ID` à l'identifiant unique du client attribué par le serveur lors de l'inscription.

L'entête doit être au format **big-endian**.

L'inscription Lors de sa première connexion au serveur, l'utilisateur doit s'inscrire auprès du serveur en envoyant son pseudo. Son message d'inscription est donc formé de :

- l'entête avec `CODEREQ = 1` et `ID = 0`,
- suivi d'un pseudo codé sur 10 octets avec complétion par une suite de caractères `#` si le pseudo fait moins de 10 caractères.

Les autres messages du client auront tous le format suivant :



où l'entête est toujours la même et

- `NUMFIL` correspond à un numéro de fil au format big-endian,
- `NB` est un nombre (numéro de port ou nombre de messages) au format big-endian,
- `DATALEN` correspond à la taille du texte `DATA` qui suit.

Poster un billet Pour poster un billet sur le fil numéro f, le client doit envoyer au serveur un message avec :

- `CODEREQ = 2` et l'identifiant de l'utilisateur,
- `NUMFIL =` f,
- `NB =` 0,
- `LENDATA` contient la taille du texte à poster,
- `DATA` contient le texte du billet.

Si f vaut 0 alors c'est que l'utilisateur demande à poster sur un nouveau fil.

Demander la liste des n derniers billets Pour demander la liste des n derniers billets sur le fil numéro f, le client doit envoyer au serveur un message avec :

- `CODEREQ = 3` et l'identifiant de l'utilisateur,
- `NUMFIL =` f,
- `NB =` n,
- `LENDATA =` 0 et le champs `DATA` est vide.

Si n vaut 0 alors c'est que l'utilisateur demande tous les billets du fil.
Si f vaut 0 alors c'est que l'utilisateur demande la liste des n derniers billets de chaque fil.
Si n et f valent 0, cela signifie donc que l'utilisateur demande tous les billets de chaque fil.

S'abonner à un fil Pour s'abonner à un fil f, le client doit envoyer au serveur un message avec :

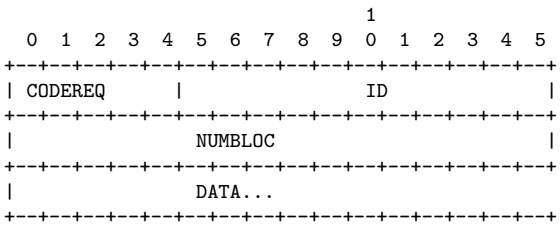
- `CODEREQ = 4` et l'identifiant de l'utilisateur,
- `NUMFIL =` f,
- `NB =` 0,
- `LENDATA =` 0 et le champs `DATA` est vide.

Ajouter un fichier Il est possible d'ajouter un fichier qui fait moins de 2^{25} octets ($\simeq 33,5$ Mo) à un fil f afin qu'il puisse être téléchargé par les utilisateurs.
Le client envoie au serveur un message avec :

- `CODEREQ = 5` et l'identifiant de l'utilisateur,
- `NUMFIL =` f,
- `NB = 0`,
- `LENDATA` contient le nombre de caractères composant le nom du fichier
- `DATA` contient le nom du fichier.

Pour envoyer en UDP les données du fichier, le client découpe les données en paquets de 512 octets numérotés de 1 au dernier. Il envoie ensuite au serveur ces paquets numérotés afin que le serveur puisse les réassembler dans l'ordre ensuite. Ainsi si le fichier fait 1034 octets, le client enverra 3 paquets, le paquet numéroté 1 aura les 512 premiers octets du fichier, le paquet numéroté 2 les 512 octets suivants et le paquet numéroté 3 les 10 derniers octets du fichier. Et si le fichier ne fait que 1024 octets, le client envoie également 3 paquets, les deux premiers paquets construits comme dans l'exemple précédent et le dernier paquet vide. Ainsi le serveur sait exactement quand toutes les données du fichier ont été reçues. Le dernier paquet numéroté p est un paquet comportant strictement moins de 512 octets de données. Il doit donc avoir reçu tous les paquets de 1 à p.

Chaque paquet UDP envoyé par le client aura le format suivant :



avec

- `CODEREQ = 5` et l'identifiant de l'utilisateur, entête au format big-endian,
- `NUMBLOC` contient le numéro du bloc de données au format big-endian,
- `DATA` contient au maximum 512 octets de données du fichier.

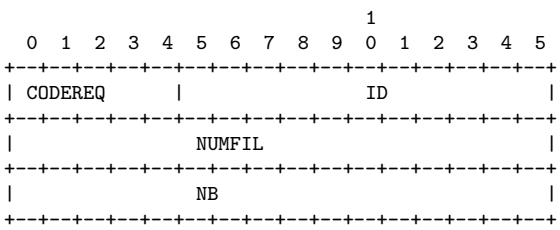
Télécharger un fichier Un utilisateur peut demander à télécharger un fichier apparaissant sur un fil f en envoyant un message avec :

- `CODEREQ = 6` et son identifiant,
- `NUMFIL` = f,
- `NB` contient ici le numéro de port sur lequel le client recevra en UDP les données du fichier demandé,
- `LENDATA` contient le nombre de caractères composant le nom du fichier
- `DATA` contient le nom du fichier à télécharger.

On suppose donc qu'un fil ne contient pas plusieurs fichiers avec le même nom.

c) Les messages du serveur

Le serveur commence par répondre à une requête du client.
Si la requête a pour code 1, 2, 3, 5, 6 ou si la requête est refusée, la réponse du serveur est un message ayant le format suivant :



où

- `CODEREQ` correspond au code de la requête du client,
- `ID` à l'identifiant unique de l'utilisateur attribué par le serveur lors de l'inscription,
- `NUMFIL` correspond à un numéro de fil au format big-endian,
- `NB` est un nombre (nombre de messages ou numéro de port) au format big-endian,

L'entête formé de `CODEREQ` et `ID` doit être au format big-endian.

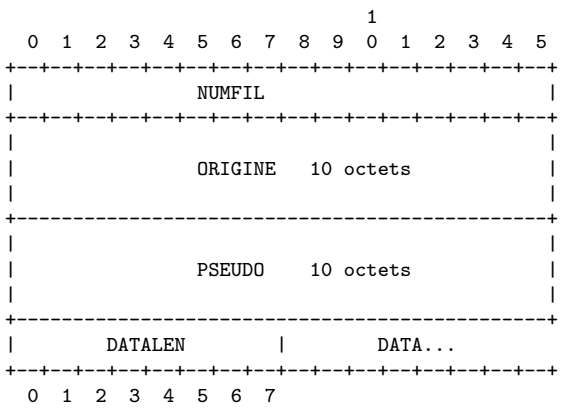
L'inscription Lors d'une demande d'inscription, le serveur attribue un identifiant unique au client qui est un nombre codé sur 11 bits. `ID` est alors égal à cette valeur, `CODEREQ` vaut 1 et `NUMFIL` et `NB` valent 0.

Poster un billet Afin que le client qui a envoyé une requête pour poster un billet sur le fil f sache que sa demande a été prise en compte par le serveur, celui-ci renvoie une réponse où les champs contiennent exactement les valeurs envoyées par le client, soit le code de sa requête (2), l'identifiant unique de l'utilisateur, la valeur 0 pour `NB` et le numéro du fil f où ajouter le billet sauf si le client demande à poster dans un nouveau fil et dans ce cas, c'est le numéro du nouveau fil qui est envoyé.

Demander la liste des n derniers billets Lorsque le client demande la liste des n derniers billets sur le fil f, le serveur envoie son premier message avec les champs `CODEREQ`, `ID` identiques à ceux contenus dans la requête du client.
Le champs `NUMFIL` est également identique s'il est strictement positif (*i.e.* il correspond à un numéro de fil). Si, par contre, il est nul dans la requête du client, alors il vaut le nombre de fils en cours dans la réponse du serveur.
si f est strictement positif, le champs `NB` est égal :

- au nombre de messages contenus dans le fil f si n est supérieur à ce nombre ou si $n = 0$,
- à n sinon.

Si f vaut 0, alors le champs `NB` est égal au nombre total de messages que le serveur va envoyer. Ce nombre est la somme du nombre de messages envoyés de chaque fil. Le nombre de messages envoyés de chaque fil suit la même régle que dans le cas où $f > 0$.
Le serveur envoie ensuite `NB` messages contenant chacun un billet. Le format des messages est le suivant :

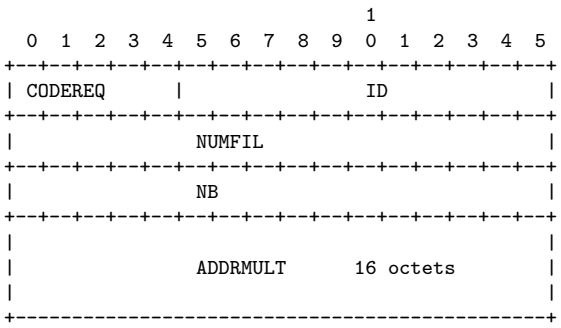


avec

- `NUMFIL` est le numéro du fil auquel appartient le billet au format big-endian,
- `ORIGINE` est le pseudo de l'initiateur du fil,
- `PSEUDO` est le pseudo de celui qui a posté le billet,
- `DATA` est le texte du billet de longueur `DATALEN`.

Erreurs Lorsque le serveur ne peut répondre à la requête du client, car il a rencontré un problème, il envoie un message d'erreur au client où `CODEREQ` vaut 31 et tous les autres champs sont à 0.

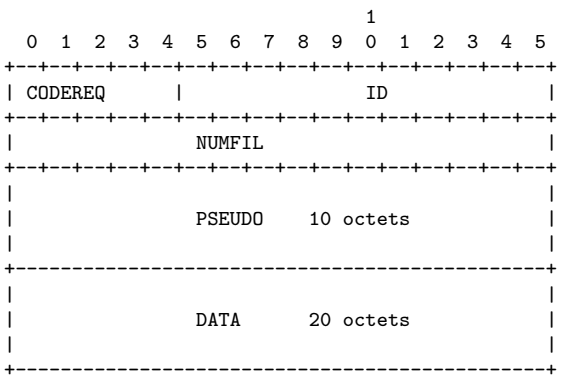
S'abonner à un fil Lorsque le client demande à s'abonner à un fil f, le serveur envoie au client une réponse au format suivant :



avec

- `CODEREQ = 4` et l'identifiant du client, entête au format big-endian,
- `NUMFIL` = f au format big-endian,
- `NB` est le numéro de port pour la multidiffusion au format big-endian,
- `ADDRMULT` est l'adresse IPv6 de multidiffusion en écriture big-endian.

Les notifications Pour chaque fil, s'il y a des abonnés, le serveur envoie à un intervalle de temps t constant, sur une adresse de multidiffusion dédiée, les billets publiés sur le fil dans l'intervalle de temps t. S'il y a eu k nouveaux billets publiés, le serveur envoie k messages, un pour chaque nouveau billet, de la forme suivante :



avec

- `CODEREQ = 4` et `ID = 0`,
- `PSEUDO` contient le pseudo du billet
- `DATA` contient les 20 premiers caractères du texte (le texte complété par des caractères nuls ('\0') s'il fait moins de 20 caractères).

Ajouter un fichier Lorsque le serveur reçoit une demande de transfert de fichier du client, il envoie une réponse avec `CODEREQ`, `ID` et `NUMFIL` ayant chacun la même valeur que dans la requête du client et `NB` égal au numéro de port sur lequel le serveur recevra les paquets UDP contenant les données du fichier.

Télécharger un fichier Lorsque le serveur reçoit une demande de téléchargement d'un fichier, il envoie une réponse avec `CODEREQ`, `ID` et `NUMFIL` et `NB` ayant chacun la même valeur que dans la requête du client pour signifier qu'il accepte la requête et va procéder au transfert.
Pour le transfert, le serveur procède de la même façon que le client (voir le paragraphe « Ajouter un fichier » dans la section décrivant les messages du client).

II) Le travail demandé

a) Réalisation

Le travail est à réaliser en trinôme. Il y a beaucoup à faire donc un binôme court le risque d'être débordé. Il n'y aura pas de dérogation à cette règle sauf exception qui doit être argumentée. Vous devez envoyer par mail à `anne.micheli@irif.fr` au plus tard le 12 mars 2023, la composition de votre équipe avec les autres membres de l'équipe en copie du mail. L'objet du mail doit être **[PR6] équipe projet**.
Vous devez écrire en C le serveur et le client correspondant au protocole *Mégaphone*. Votre serveur devra accepter les connexions IPv6 et IPv4.
Votre client devra pouvoir interagir avec le serveur minimal qui tourne sur lulu sur le port 7777. Ce serveur traite uniquement les requêtes de code 1, 2 ou 3 et les messages d'erreurs.
Les erreurs devront être gérées, c'est-à-dire qu'il faut réfléchir aux actions en cas de réception d'un message mal formaté ou d'un appel système erroné.
La propreté du code, sa lisibilité seront également pris en compte. Pensez à commenter votre code et à un mode verbeux pour la soutenance. Evitez le code spaghetti, mutualisez le code...
Vous pouvez discuter entre vous du fonctionnement du protocole, du format des messages... mais vous ne pouvez pas vous échanger ou montrer du code. Le plagiat est **strictement interdit**.

b) Pour aller plus loin

En plus du projet minimal décrit ci-dessus, toute amélioration utilisant les notions du cours sera prise positivement. Mais attention à tout d'abord terminer la partie obligatoire. Par ailleurs, l'ajout ne doit pas avoir d'impact sur la partie minimale.
Vous pourrez, par exemple, travailler une des extensions suivantes :

- messages d'erreur ciblés du serveur lorsque la requête du client est mal formatée,
- sauvegardes régulières des données du serveur (listes des inscrits, des fils de discussions, des adresses d'abonnements...) afin de pouvoir les récupérer en cas de crash du serveur,
- résistance du serveur aux attaques (par exemple, un utilisateur qui fait trop de transfert de fichiers ou qui fait des requêtes multiples pour envoyer un fichier et n'envoie jamais rien...),
- gestion des messages privés.

Cette liste n'est pas exhaustive. Vous pourrez proposer de nouvelles fonctionalités en les décrivant précisément (notamment le format des messages échangés) dans un fichier `extension.md`.

c) Modalités de rendu

Dès que votre trinôme est constitué, vous devez créer un dépôt git **privé** sur le gitlab de l'UFR [1].
Tous les enseignants du cours devront y avoir accès en tant que **Reporter**, c'est-à-dire, Pierre Charbit, Aldric Degorre, Colin Gonzalez, Anne Micheli, Florian Renkin et Yaëlle Vinçont. Le dépôt devra contenir un fichier `authors.md` contenant la liste des membres de l'équipe (nom, prénom, numéro étudiant et pseudo(s) sur le gitlab).
Votre dépôt final devra contenir vos fichiers source ainsi qu'un `Makefile` pour la compilation et le fichier `extension.md` le cas échéant. La compilation et l'exécution devront fonctionner sur les machines de l'UFR.
Ce rendu donnera lieu à une soutenance qui se déroulera en fin de période d'examens. Des précisions seront apportées ultérieurement.

1. https ://gaufre.informatique.univ-paris-diderot.fr/

Attention, les soumissions (commits) sur le gitlab rendront compte en partie de votre implication dans le projet. Si nous découvrons le jour de la soutenance en regardant sur le gitlab qu'une personne n'a pas participé au projet, sa note sera 0 au projet.

d) Discord

Un serveur discord a été mis en place afin que vous puissiez discuter entre vous, nous poser des questions pour clarifier le sujet... Vous trouverez le lien sur Moodle. Seules les réponses données par les enseignants sur discord feront foi. Il faut donc que vous posiez les questions concernant le projet sur ce serveur.
Vous pourrez également annoncer que votre serveur tourne en précisant son adresse et port, afin que d'autres groupes puissent le tester avec leur client. De plus, vous pourrez préciser les extensions ajoutées avec le format pour que des clients puissent les tester. Il est très important que vos applications interagissent avec d'autres implémentations, donc pensez a minima, à faire des tests inter-groupes.