PR6 – Programmation réseaux
Projet *AuctionP2P*

Le but de ce projet est de créer une application pair à pair sur IPv6 implémentant un protocole d'enchères que nous nommerons *AuctionP2P*.
La première partie décrit le protocole sans être précis sur le format des messages échangés. Les messages sont décrits dans la partie II).

I) Le protocole *AuctionP2P*

Le protocole *AuctionP2P* est un protocole d'enchères entre pairs. Chaque pair peut débuter une vente aux enchères et la vente est lancée avec un prix initial. N'importe quel pair est alors libre d'enchérir, c'est-à-dire de proposer un prix plus élevé que le prix initial. Si la proposition est validée, l'enchère suivante devra être plus élevée, et ainsi de suite... La vente s'arrête lorsqu'il n'y a plus aucune nouvelle proposition depuis un certain temps.
Pour fonctionner le protocole *AuctionP2P* nécessite plusieurs paramètres.
Tout d'abord, la communication entre les pairs utilise plusieurs adresses :

- une adresse *liaison* : adresse IPv6 de multidiffusion avec son port pour rejoindre le système d'enchères entre pairs,
- une adresse *adressePerso* par pair : adresse IPv6 et port d'écoute pour la communication en mode TCP et en mode UDP,
- une adresse *encheres* : adresse IPv6 et port de multidiffusion pour la communication entre tous les pairs.

Par ailleurs, en plus de l'adresse personnelle *adressePerso*, un pair devra posséder un identifiant unique dans le système d'enchères et une paire de clés (privée, publique) au format ED25519.
Étant donné que le protocole est pair à pair, les pairs doivent arriver à un consensus pour valider une transaction. En effet, si, par exemple lors d'une vente, deux pairs envoient simultanément la même proposition de prix, les autres pairs vont recevoir les deux propositions dans un ordre qui peut varier d'un pair à l'autre. Pourtant, il faut décider quelle proposition est acceptée et laquelle est rejetée.

a) Le processus de consensus

Certains messages multidiffusés demandent à être validés par tous les pairs afin qu'à tout moment, chaque pair ait les mêmes informations que les autres. Le consensus ici est en fait l'unanimité !
Chaque pair va donc avoir besoin d'engendrer une paire de clés (privée, publique) au format ED25519. Pour valider un message, un pair signe alors le message avec sa clé privée. Chaque pair doit pouvoir vérifier un message signé par un autre pair et doit donc posséder les clés publiques de tous les pairs.
Lorsqu'un pair Q multidiffuse un message `mess` qui doit être validé par consensus, il supervise le processus de validation entre pairs et est nommé le *superviseur*. Le processus de consensus suit alors le déroulement suivant :

1. le superviseur Q attend, sur son adresse *adressePerso* en mode UDP, la réception d'un message de validation du message `mess` par chaque pair du système d'enchères (message de code 1, voir section II)) et vérifie chaque validation,
2. si la vérification d'un message signé par un pair P échoue, Q fait comme s'il n'avait pas reçu de validation de P,

3. si parallèlement aux étapes précédentes, Q reçoit un message concurrent, il réagit comme expliqué à la section f),
4. une fois les validations des pairs présents récupérées, le pair Q signe le message avec sa clé privée et concatène dans un message, le message, sa signature et les validations reçues des autres pairs, puis multidiffuse sur l'adresse *enchères*, ce message (messages de code 2 et 20, voir section II)).
5. à la réception du message de Q, chaque pair vérifie que le message `mess` a bien été signé par chaque pair du systèmes. Si la vérification réussie, le pair considère que le message est validé.

Aux étapes 1 et 5, en cas de non réception de la réponse attendue dans les t_1s [1], le protocole décrit dans la section g) est suivi. À ces étapes, si la vérification d'un message par un pair (le superviseur Q à l'étape 1 et un pair à l'étape 5) échoue, celui-ci fait comme s'il n'avait pas reçu le message et suit donc également le protocole décrit dans la section g).
On ne demande pas de gérer la situation où une vérification ne réussit jamais, puisque cela ne devrait pas arriver. En effet, les clés publiques et privées n'étant jamais modifiées, il n'y a pas de raison que la vérification échoue indéfiniment.

b) Rejoindre un système d'enchères

Lorsqu'un pair P souhaite rejoindre un système d'enchères, le protocole à suivre est le suivant :

1. le pair P multidiffuse sur l'adresse *liaison*, une demande pour rejoindre le système d'enchères (message de code 3, voir section II)) ,
2. si après t_2s [2], il n'a pas reçu de réponse, il reprend l'étape 1 de multidiffusion, sauf si c'est la troisième fois qu'il ne reçoit rien et dans ce cas il considère qu'il n'y a pas de système d'enchères formé. Le pair P forme alors le système d'enchères et attend de nouveaux pairs sur l'adresse *liaison*,
3. si le système d'enchères est déjà actif, les pairs du système d'enchères, à la réception de la demande de liaison, répondent en **unicast** au pair P en lui communiquant leur adresse personnelle *adressePerso* en mode TCP (message de code 4, voir section II)),
4. le pair P choisit un pair Q du système d'enchères parmi les pairs ayant répondu et se connecte alors via TLS à l'adresse *adressePerso* en mode TCP de Q. Une fois connecté, P transmet à Q son identifiant, son adresse *adressePerso* et sa clé publique (message de code 5, voir section II)) et Q lui répond en lui envoyant un nouvel identifiant si l'identifiant proposé par P est déjà pris (messages de code 50 et 51). La connexion reste ouverte pour l'étape 6,
5. Q se connecte alors, via TLS, à l'adresse *adressePerso* en mode TCP de chaque autre pair du système, pour transmettre l'identifiant, l'adresse *adressePerso* et la clé publique de P (message de code 6, voir section II)),
6. Q envoie à P en mode TCP un message contenant l'adresse *encheres* et des informations sur chaque pair du système, c'est-à-dire son identifiant, son adresse *adressePerso* et sa clé publique (message de code 7), voir section II)), puis termine la connexion.

Un pair P doit pouvoir rejoindre un système à n'importe quel moment. Une fois les étapes ci-dessus réussies, P doit participer aux ventes en cours bien qu'il n'était pas présent au début.

1. par exemple 5s
2. par exemple 5s

c) Déroulement d'une vente aux enchères

N'importe quel pair du système peut initier une vente aux enchères. Pour commencer, le pair doit attribuer un numéro à sa vente. Chaque pair possède un compteur interne `cpt` du nombre de ventes qu'il a initiées. Le numéro de la nouvelle vente est alors la concaténation de l'identifiant du pair et de `cpt+1`, produisant ainsi un numéro de vente unique.
Il multidiffuse alors sur l'adresse *encheres*, un message avec le numéro de la vente et prix initial de l'enchère (message de code 7), voir section II)). Chaque pair peut alors enchérir en multidiffusant sur l'adresse *encheres*.
A chaque nouvelle vente, le pair qui initie la vente prend un rôle spécifique, celui de superviser les validations d'enchères sur cette vente, ainsi que la fin de la vente. Ce pair est nommé le *superviseur* de la vente.
Lorsqu'une enchère est multidiffusée par un pair P (message de code 9, voir section II)), le superviseur S multidiffuse à nouveau le message d'enchère de P (message de code 10) et engage le processus de consensus pour la validation de ce message.

d) Finalisation d'une vente aux enchères

Si plus de t_3s [3] se sont écoulées depuis la dernière validation d'une enchère ou depuis le lancement d'une vente, sans aucune nouvelle proposition d'enchère, le superviseur multidiffuse sur l'adresse *encheres* un message avec le numéro et dernier prix de la vente (message de code 11, voir section II)). Si de nouveau, aucune enchère n'est proposée après t_3s, la vente aux enchères est terminée. Le superviseur de la vente multidiffuse alors un message contenant l'identifiant du gagnant, s'il y en a un, et le prix final de l'enchère (message de code 12) et engage un processus de consensus afin de valider le message multidiffusé et donc, la fin de la vente.

e) Quitter le système d'enchères

Lorsqu'un pair P souhaite quitter le système d'enchères, il multidiffuse sur l'adresse *encheres* un message de fin (message de code 13, voir section II)), ce qui engage un processus de consensus avec P comme superviseur. Ensuite, chaque pair retire P du système d'enchères.

f) Gestion des messages concurrents ou non valides

Lors du processus de consensus, le superviseur peut recevoir un message concurrent. C'est le cas lors de la validation d'une enchère, le superviseur peut recevoir une autre proposition d'enchère par un pair P. Il rejette l'enchère en multidiffusant sur l'adresse *enchere* un refus de cette enchère (message de code 14, voir section II)).
Le superviseur peut également recevoir à tout moment une enchère avec un prix inférieur au prix courant de la vente. Dans ce cas, il rejette également l'enchère en multidiffusant sur l'adresse *enchere* un refus de cette enchère (message de code 15).

g) Cas des réponses UDP perdues

Le protocole de transport UDP n'étant pas fiable, il se peut qu'un message UDP, multidiffusé ou en unicast, se perde. Si des réponses en UDP sont attendues par le pair P à un message `mess` qu'il a envoyé précédemment, le protocole, en cas de non réponse de certains pairs, est le suivant :

1. le pair P réemet en mode UDP sur les adresses *adressePerso* des pairs n'ayant pas répondu, le message `mess`,

3. par exemple 30s

2. si après t_1s, il reste encore des pairs n'ayant pas répondu, il réemet comme à l'étape 1,
3. si après t_1s, il reste encore des pairs n'ayant pas répondu, ces pairs sont alors considérés absents et P suit le protocole décrit en section h) concernant les pairs absents.

Ce protocole n'est pas suivi dans deux cas particuliers :

- si le message `mess` est une demande pour rejoindre le système d'enchères (voir section b)),
- ou lors de l'attente d'une nouvelle enchère (voir section d)).

h) Gestion des pairs ne répondant plus

disparition d'un superviseur Il se peut que lors du processus de consensus, le superviseur S disparaisse brusquement. Dans ce cas, le processus ne peut continuer, la transaction en cours doit être annulée. Un message annulant la transaction et contenant l'identifiant du pair disparu est multidiffusé sur l'adresse *encheres* (message de code 16, voir section II)) et un processus de consensus est engagé. Il est probable que plusieurs pairs lancent simultanément ce processus de consensus. C'est le pair avec l'identifiant le plus petit qui sera in fine le superviseur du processus de consensus.
Si le pair P_1 reçoit de la part d'un autre pair P_2, une demande de validation de l'annulation d'une transaction, alors qu'il a fait la même demande :

- et l'identifiant de P_1 est plus petit que celui de P_2, alors P_1 ignore la demande de P_2,
- et l'identifiant de P_1 est plus grand que celui de P_2, alors P_1 valide le message de P_2 et multidiffuse un message pour annuler sa demande de validation (message de code 17).

disparition de pairs non superviseurs Si c'est un ou plusieurs pairs, non superviseurs de la transaction en cours, qui ne répondent pas, alors ces pairs sont considérés absents par le pair P attendant la réponse. P multidiffuse alors sur l'adresse *encheres* un message pour retirer du système ces pairs absents (message de code 18, voir section II)) et engage un processus de consensus où P est le superviseur. Il y a une exception lors de la demande d'un pair P pour entrer dans le système. Si le pair Q choisi par P pour l'intégrer au système ne répond plus, P n'étant pas encore inséré ne peut avertir le système de la disparition de Q. Il reprend tout simplement le protocole pour rejoindre le sytème d'enchères à son début.

i) La gestion interne des pairs

La gestion interne des pairs est laissé à votre appréciation, notamment concernant l'interactivité avec l'utilisateur pour l'initialisation d'une vente et les enchères.

II) Les formats de messages

On commence par décrire les différents champs qui composeront les messages :

- le code du message, noté `CODE`, permet d'identifier le type de message et est un entier non signé sur 1 octet,
- l'identifiant, noté `ID`, est un entier non signé sur 2 octets,
- la longueur d'un message, notée `LMESS`, est un entier non signé sur 1 octets,
- la longueur d'une signature, notée `LSIG`, est un entier non signé sur 1 octets,
- le message, noté `MESS`, à transmettre sous forme d'une suite de caractères,
- la signature, notée `SIG`, est une suite de caractères,
- une adresse IP, notée `IP`, est un entier de 16 octets correspondant à l'adresse IP,

- un port, noté `PORT`, est un entier non signé sur 2 octets,
- une clé, notée `CLE`, qui correspond à la suite de caractères définissant la clé,
- `NUMV` est un entier non signé sur 4 octets, correspondant au numéro d'une vente,
- `PRIX` est un entier non signé sur 4 octets, correspondant au prix d'une vente,
- `NB` est un entier non signé sur 2 octets.

Tous les entiers doivent être inclus dans les messages au **format réseaux** (big-endian).
Chaque pair devra produire sa paire de clés publique et privée au format ED25519. La longueur d'une clé publique ED25519 est de 60 octets. Si on ajoute les textes d'entêtes et de fin, le fichier contenant la clé a alors une taille de 113 octets. C'est ces 113 octets qui sont inclus dans les messages et correspondent au champ `CLE`.

a) Le processus de consensus

Dans les messages ci-dessous, lorsque vous travaillerez sur la version sans sécurité du projet, la longueur de la signature `LSIG` sera égale à 0 et donc le champ `SIG` sera absent du message.

1. À l'étape 1, le superviseur Q attend que chaque pair P lui envoie le message `mess` validé, c'est-à-dire qu'il attend un message du type :

   ```
   +++++++++++++++++++++++++++++++++++++++++++
   | CODE = 1 | ID | LMESS | MESS | LSIG | SIG |
   +++++++++++++++++++++++++++++++++++++++++++
   ```

   où `ID` est l'identifiant du pair P, `LMESS` la longueur du message `mess`, `MESS` le message `mess`, `LSIG` la longueur de la signature du message `mess`, `SIG` la signature du message `mess`.

4. À l'étape 4, le message envoyé par Q aux pairs commence par :

   ```
   +++++++++++++++++++++++++++++++++++++++++++++++++
   | CODE = 2 | ID | LMESS | MESS | LSIG | SIG | NB |
   +++++++++++++++++++++++++++++++++++++++++++++++++
   ```

   où `ID` est l'identifiant de Q, `LMESS` la longueur du message `mess`, `MESS` le message `mess`, `LSIG` la longueur de la signature du message `mess`, `SIG` la signature du message `mess` et `NB` le nombre de pairs ayant signé le message `mess` (excepté Q).
   Étant donné qu'il est préférable qu'un message UDP ne soit pas fragmenté lors de la transmission, il faut ici distinguer selon le nombre de pairs qui participent au consensus. Il y a donc deux cas selon que le nombre de pairs, hors Q, ayant signé soit inférieur ou égal à 3 ou supérieur :

   - si `NB` $\leqslant 3$, alors le message est complété avec la concaténation des identifiants, longueurs et signatures du message `mess` des pairs. Donc pour chaque pair, on concatène au message, un message au format suivant :

     ```
     ++++++++++++++++++
     | ID | LSIG | SIG |
     ++++++++++++++++++
     ```

   - si `NB` > 3, le premier message multidiffusé est complété comme décrit ci-dessus. Puis pour les pairs suivants, on envoie autant de messages nécessaires qu'il faut pour que toutes les signatures soient multidiffusées. Chaque message a l'entête suivant :

     ```
     ++++++++++++++++++++++
     | CODE = 20 | ID | NB |
     ++++++++++++++++++++++
     ```

où `ID` est l'identifiant de Q et `NB` le nombre de signatures contenus dans le message. Le message est alors complété avec la concaténation des identifiants, longueurs et signatures du message `mess` des `NB` pairs.

b) Rejoindre un système d'enchères

Dans les messages ci-dessous, lorsque vous travaillerez sur la version sans sécurité du projet, vous pourrez remplacer les 113 octets du champs `CLE` par une suite de 113 caractères nuls (`'\0'`). Lorsqu'un pair P souhaite rejoindre un système d'enchères :

1. à l'étape 1, le message envoyé par P est le suivant :

```
++++++++++++
| CODE = 3 |
++++++++++++
```

3. à l'étape 3, les pairs répondant à P envoient le message au format suivant :

```
++++++++++++++++++++++++++++
| CODE = 4 | ID | IP | PORT |
++++++++++++++++++++++++++++
```

où `ID` est l'identifiant du pair, `IP` l'adresse IP de l'adresse *adressePerso* du pair et `PORT` le numéro de port de cette adresse.

4. à l'étape 4, P envoie un message à Q au format suivant :

```
++++++++++++++++++
| CODE = 5 | INFO |
++++++++++++++++++
```

où `INFO` est le message au format suivant :

```
++++++++++++++++++++++
| ID | IP | PORT | CLE |
++++++++++++++++++++++
```

où `ID` est l'identifiant de P, `IP` et `PORT` l'adresse IP et le numéro de port de l'adresse *adressePerso* de P et `CLE`, sa clé publique.
Si l'identifiant poposé par P n'est pas déjà pris par un autre pair, Q répond par le message :

```
++++++++++++
| code = 50 |
++++++++++++
```

Sinon, Q répond par le message :

```
++++++++++++++++++
| code = 51 | ID |
++++++++++++++++++
```

où `ID` est l'identifiant attribué à P.

5. à l'étape 5, Q envoie à chaque pair du système, le même message qu'à l'étape 4 mais où `CODE` vaut 6 et où l'identifiant de P a éventuellement été modifié (si l'identifiant proposé par P était déjà utilisé) :

```
++++++++++++++++++
| CODE = 6 | INFO |
++++++++++++++++++
```

6. à l'étape 6, Q envoie à P une message avec le préfixe suivant :

```
++++++++++++++++++++++++++++++++
| CODE = 7 | ID | IP | PORT | NB |
++++++++++++++++++++++++++++++++
```

où `ID` est l'identifiant de Q, `IP` l'adresse IP de l'adresse *encheres*, `PORT` le numéro de port de l'adresse *enchere*, `NB` le nombre de pairs du système (Q compris).
A ce préfixe, les `INFO` sur chaque pair du système sont concaténées.

c) Déroulement d'une vente aux enchères

Pour initier une vente, un pair S multidiffuse le message au format suivant :

```
++++++++++++++++++++++++++++++
| CODE = 8 | ID | NUMV | PRIX |
++++++++++++++++++++++++++++++
```

où `ID` est l'identifiant de S, `NUMV` le numéro attribué à la nouvelle vente et `PRIX` le prix initial de cette vente.
Pour enchérir sur cette vente, un pair P multidiffuse alors un message au format suivant :

```
++++++++++++++++++++++++++++++
| CODE = 9 | ID | NUMV | PRIX |
++++++++++++++++++++++++++++++
```

où `ID` est l'identifiant de P, `NUMV` le numéro de la vente et `PRIX` le nouveau prix que propose P.
Le superviseur S de la vente multidiffuse alors ce message mais avec le code 10 (au lieu de 9).

d) Finalisation d'une vente aux enchères

Le superviseur S multidiffuse un premier message sur l'adresse *encheres* au format suivant :

```
++++++++++++++++++++++++++++++
| CODE = 11 | ID | NUMV | PRIX |
++++++++++++++++++++++++++++++
```

où `ID` est l'identifiant de S, `NUMV` le numéro de la vente et `PRIX` le prix de la dernière enchère validée s'il y en eu au moins une, ou le prix intial sinon.
Pour conclure, S multidiffuse le message suivant :

```
++++++++++++++++++++++++++++++
| CODE = 12 | ID | NUMV | PRIX |
++++++++++++++++++++++++++++++
```

où `ID` est l'identifiant du gagnant de la vente s'il y en a un, sinon de S, `NUMV` le numéro de la vente et `PRIX` le prix de la dernière enchère validée s'il y en eu au moins une, ou le prix intial sinon.

e) Quitter le système d'enchères

Le message de fin d'un pair P a le format suivant :

```
++++++++++++++++++
| CODE = 13 | ID |
++++++++++++++++++
```

où `ID` est l'identifiant de P.

f) Gestion des messages concurrents ou non valides

Pour rejeter une enchère d'un pair P, le superviseur multidiffuse un message au format suivant :

```
++++++++++++++++++++++++++
| CODE | ID | NUMV | PRIX |
++++++++++++++++++++++++++
```

où `ID` est l'identifiant de P, `NUMV` le numéro de la vente et `PRIX` le prix proposée par P et `CODE` vaut 14 si l'enchère est rejetée car en concurrence avec une autre enchère et 15 sinon.

g) Gestion des pairs ne répondant plus

disparition d'un superviseur Si un superviseur S disparaît brusquement, le message d'annulation de la vente et de disparition du pair S est le suivant :

```
++++++++++++++++++++++++
| CODE = 16 | ID | NUMV |
++++++++++++++++++++++++
```

où `ID` est l'identifiant de S, `NUMV` le numéro de la vente.
Si deux pairs P_1 et P_2 envoie un même message de code 16, alors si P_2 a un identifiant plus grand que P_1, il annule son message en multidiffusant ce même message mais avec le code 17.

disparition de pairs non superviseurs Le message pour retirer du système des pairs disparus, multidiffusé par un pair P, a l'entête suivant :

```
++++++++++++++++++++++
| CODE = 18 | ID | NB |
++++++++++++++++++++++
```

où `ID` est l'identifiant de P et `NB` est le nombre de pairs absents.
Les `NB` identifiants des pairs disparus sont ensuite concaténés au message.

III) Le travail demandé

a) Réalisation

Le travail est à réaliser en trinôme. 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 31 mars 2025, 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 l'application implémentant le protocole *AuctionP2P*. Elle devra fonctionner sur IPv6.
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.
Il faudra également veiller à ce que vos applications ne bloquent pas éternellement. Pensez par exemple au cas du pair qui intègre un système mais ne se déclare jamais prêt. Cela signifie que vos applications doivent décider à un moment, qu'un pair n'est plus actif si cela fait trop longtemps qu'elles attendent un message.
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... Et faites attention aux fuites de mémoire, de descripteurs...
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) Organisation du travail

Le projet est conséquent. Il est donc conseillé de procéder par étape :

1. commencez par écrire une version simple du protocole, sans sécurité (sans signature et en utilisant TCP au lieu de TLS), pour 4 pairs au maximum dans le système d'enchères et qui considère de plus qu'aucun pair ne disparaît, que les messages UDP ne se perdent pas et qu'il n'y a pas plusieurs ventes aux enchères simultanément,
2. puis ajouter la gestion des pairs qui disparaîssent, des messages UDP qui se perdent ainsi que la gestion des ventes en parallèle,
3. puis sécurisez les échanges TCP et le processus de consensus avec les signatures,
4. à la fin, ajouter la gestion d'un système avec plus que 4 pairs.

Le protocole n'étant pas prévu de fonctionner lorsqu'un pair supervise simultanément deux actions, il faudra toujours s'assurer qu'un pair ne supervise au plus qu'une action.

c) Modalités de rendu

Dès que votre binôme est constitué, vous devez créer un dépôt git **privé** sur le gitlab de l'UFR [4]. Tous les enseignants du cours devront y avoir accès en tant que **Reporter**, c'est-à-dire, Astyax Nourel, Fabien de Montgolfier, Aldric Degorre, Peter Habermehl et Anne Micheli. 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. 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.
Attention, les soumissions (commits) sur le gitlab rendront compte en partie de votre implication dans le projet. Si, par exemple, 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) Forum Moodle

Un forum Moodle 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 cette liste feront foi. Il faut donc que vous posiez les questions concernant le projet sur cette liste.
Vous pourrez également annoncer qu'un pair tourne en précisant l'adresse *liaison*, afin que d'autres groupes puissent le tester avec leur application. Il est très important que vos applications interagissent avec d'autres implémentations, donc pensez *a minima* à faire des tests inter-groupes.

e) Soutenances

Les soutenances se dérouleront la semaine du 26 mai 2025 en 2027.
Il faudra donc que vous veniez avec un **ordinateur**. Si cela n'est pas possible, nous devons absolument être prévenus quelques jours avants.
Vous devrez nous faire une démonstration de votre projet qui devra tourner sur le réseau de l'UFR, en lançant des pairs sur des machines distinctes. Il est donc impératif que votre ordinateur pour la soutenance puisse **se connecter au réseau de l'UFR**.

4. https ://moule.informatique.univ-paris-diderot.fr/