Projet Programmation et Génie Logiciel

Projet FourmyLaby Rust : Soutenance

Rappel du principe du jeu

- Jeu en **grille 2D** dans un labyrinthe (généré automatiquement)
- Jeu de "colonie de fourmis" **multijoueur**
- Architecture **client-serveur TCP/IP**
- Le serveur doit gérer **plusieurs parties ainsi que plusieurs joueurs**
- Communication en **JSON**, suivant un **certain protocole**

plusieurs sockets, plusieurs parties en même temps...

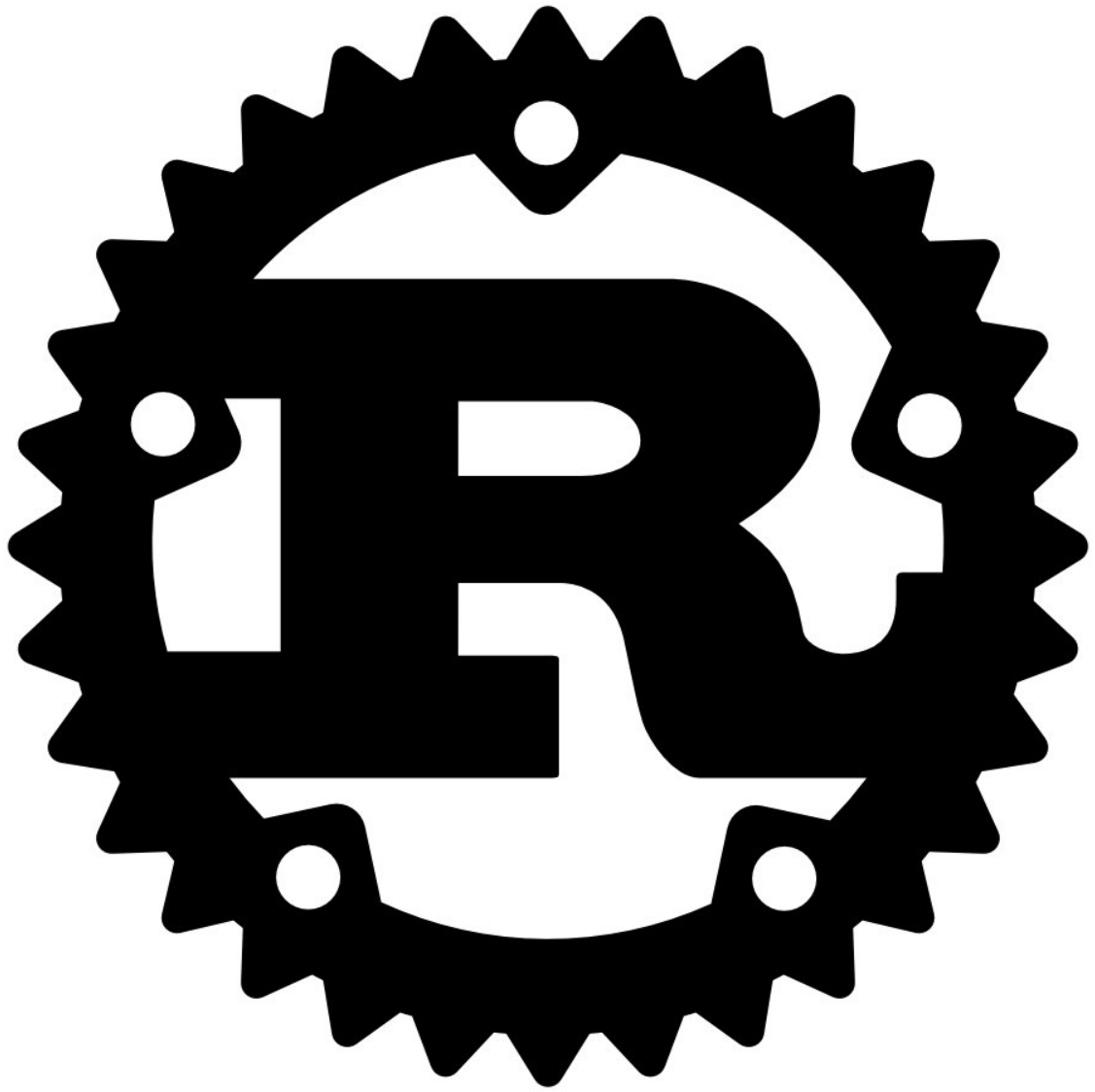
Des histoires de langage de programmation système

Le soucis des langages système vus jusqu'à maintenant (C, C++)

- **Gestion des ressources** qui risque de devenir complexe (plusieurs parties, plusieurs sockets, ...)
- Plusieurs threads donc des histoires de **synchronisation** qui ne risque pas d'arranger les choses (race conditions, deadlocks)

On risque de se retrouver avec un projet qui va devenir d'autant plus **complexe**, que l'on va manipuler d'objets.

Que faire ?



The Rust Programming Language

Rust

Un langage taillé pour notre projet

- mécanisme original mais efficace de gestion des ressources (**système d'ownership strict**)
- **évite les problèmes classiques** des langages usuels (fuites mémoire, dangling reference, race conditions, aliasing, etc.)
- langage **concurrent** (modèle similaire à l'Erlang)

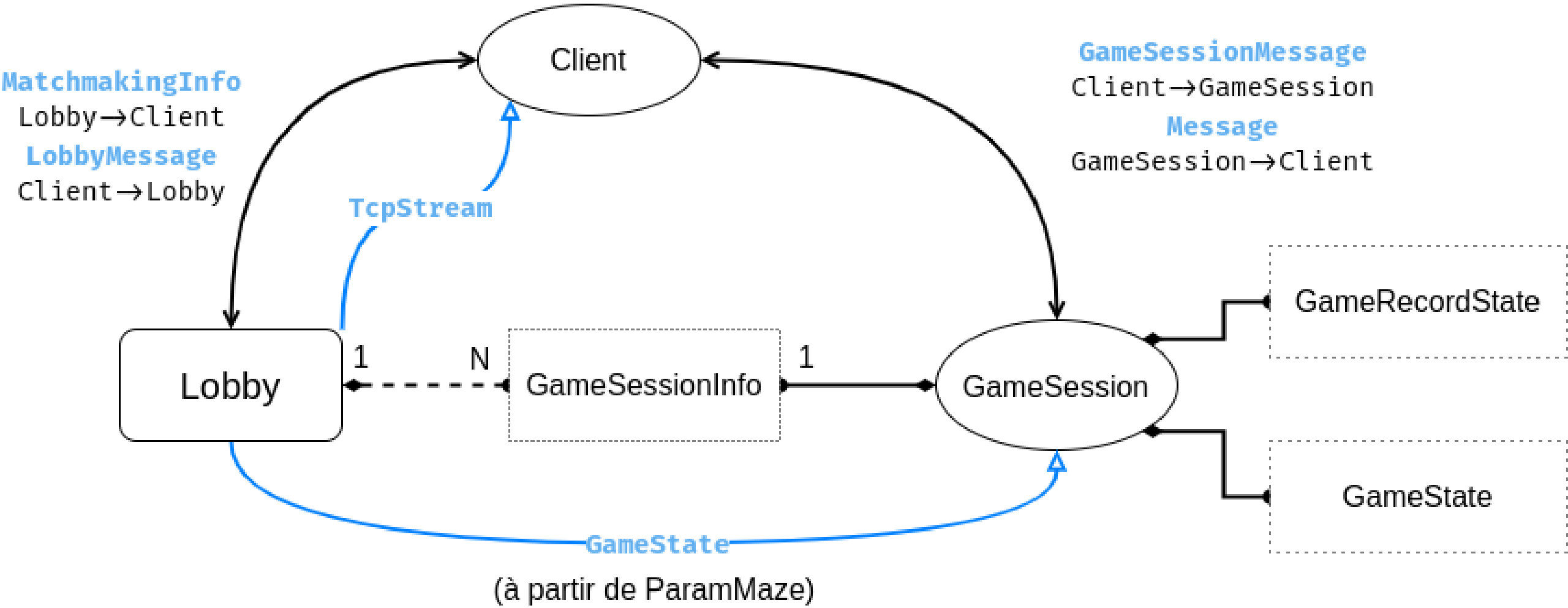
- “ langage de programmation fiable, concurrent, pratique ”

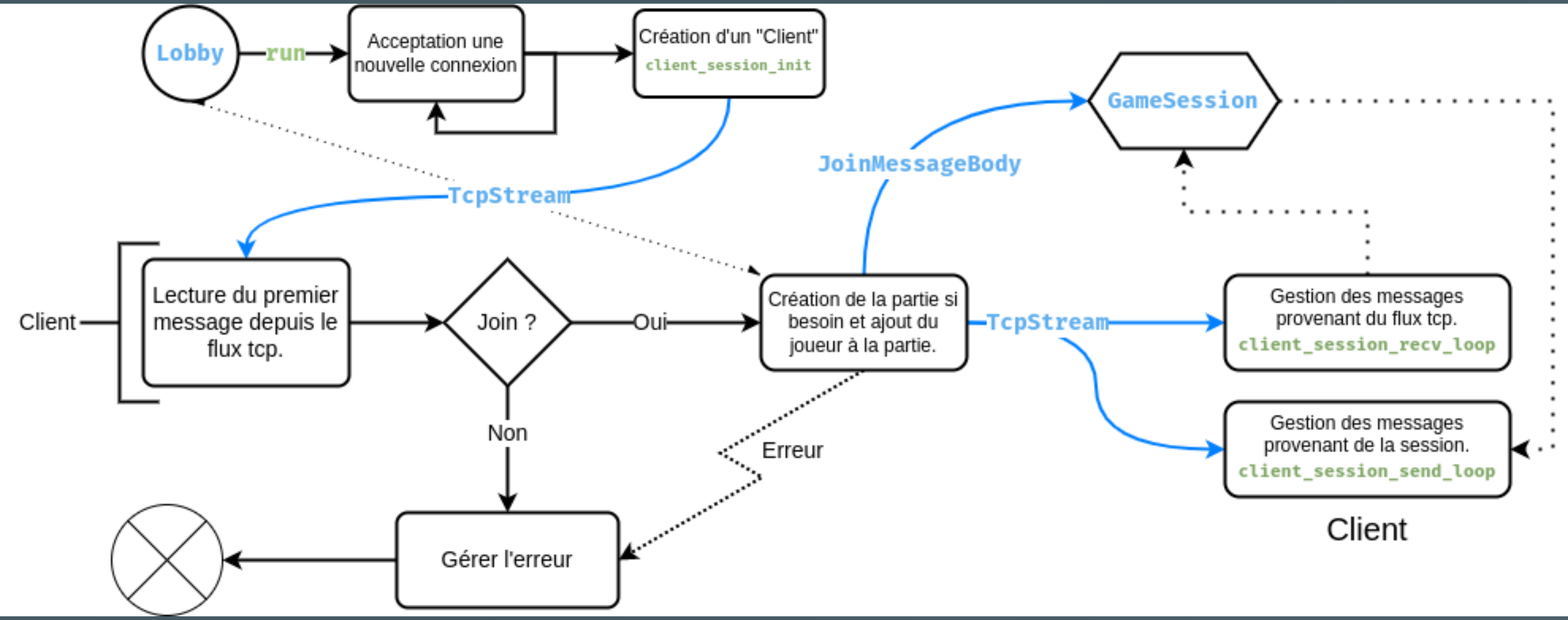
Témoignages

“ Speaking of languages, it's time to halt starting any new projects in C/C++ and use Rust for those scenarios where a non-GC language is required. For the sake of security and reliability. the industry should declare those languages as deprecated
Mark Russinovich (CTO Microsoft Azure) ”

“ To date, there have been zero memory safety vulnerabilities discovered in Android’s Rust code.
Jeffrey Vander Stoep (Google Security Team) ”

Parlons du projet


Client
MatchmakingInfo
Lobby→Client
LobbyMessage
Client→Lobby
TcpStream
GameSessionMessage
Client→GameSession
Message
GameSession→Client
Lobby
1
N
GameSessionInfo
1
GameSession
GameRecordState
GameState
GameState
(à partir de ParamMaze)


Lobby
run
Acceptation une nouvelle connexion
Création d'un "Client"
client_session_init
GameSession
JoinMessageBody
TcpStream
Client
Lecture du premier message depuis le flux tcp.
Join ?
Oui
Non
Création de la partie si besoin et ajout du joueur à la partie.
TcpStream
Gestion des messages provenant du flux tcp.
client_session_recv_loop
Gestion des messages provenant de la session.
client_session_send_loop
Erreur
Gérer l'erreur
Client

Un choix d'implémentation particulier :

Système concurent par passage de message

- “ Ne pas partager la mémoire, à la place, partager la mémoire en communiquant (Andrew Gerrand, Golang) ”
- le partage mémoire implique beaucoup de **Mutex** donc **complique** beaucoup de code, et **n'est pas approprié ici au multithreading**
- on pourrait sacrifier l'aspect multithread, mais ça ne résoud pas vraiment le **problème de gestion multiples de parties/sockets**
- le Rust est **adapté** pour le passage de message (module **mpsc**)

Le projet en Rust

Le langage Rust a apporté plusieurs choses au projet :

- Garanties de fiabilité : le code est sûr et robuste
- Support des *tagged unions* et *pattern matching* :
  - simplifie le traitement des messages
- très pratique framework de sérialisation (**serde**)
- mécanisme de gestion d'erreur fléxible (**Result<T, E>**)
- très puissant ensemble d'outils (**cargo**, **rustdoc**, **compilateur**, ...)

Organisation

Gitlab de l'université

Pas de difficulté sur ce projet en Rust
mais...
éparpillée pour le méta-projet Labyrinthe de Fourmis

- Problèmes de communication avec les autres binômes
- Aucun client/simulateur de joueur n'a été rendu disponible
- Code des librairies de génération de labyrinthe reçue tardivement

Conclusion

Le Rust a donné des résultats positifs dans le développement du projet, de plusieurs manières (sûreté du langage, outils, etc.).

Nombreuses opportunitées d'emploi/stage en Rust, étant donné l'intérêt grandissant pour le langage.

> “ Rust does best when we're ambitious
> Niko Matsakis (Amazon AWS) ”

Demo