Automate Poignée de main WS2P

03/12/2020

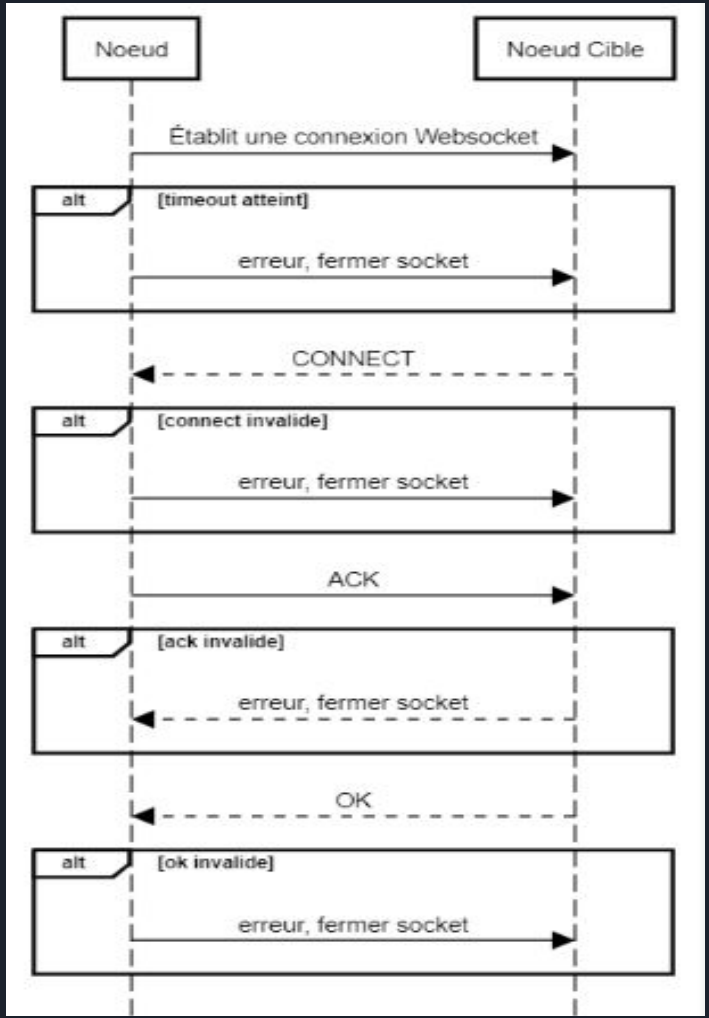
Contexte

Ouverture d'une connexion WS2P:

Le Noeud ayant ouvert la connexion envoie un message CONNECT puis attend un ACK pour répondre un OK, mais le noeud recevant la connexion Websocket procède de manière identique: il envoie un CONNECT et attend un ACK correspondant pour répondre OK.

Deux demi poignées de main indépendantes se déroulent symétriquement via la même socket.

>> Diagramme de séquence d'une demie poignée de main: Le noeud cible effectue lui aussi cet échange.



Problème

Comme indiqué par Elois lors de la réunion du 31/11/2020, la difficulté réside dans le fait que deux communication simultanées et indépendantes ont lieu dans la même socket. Ainsi, on doit s'attendre à recevoir tout type de message à un instant donné, puisqu'on peut aussi bien recevoir une réponse liée à sa demi-poignée de main qu'à celle de l'autre demie poignée de main.

Toutefois cette difficulté n'existe pas vraiment en Elixir où un tel automate peut être implémenté par l'utilisation d'un Genserver OTP.

Solution proposée

On utilisera un genserver pour implémenter l'automate à états suivants, pour chaque poignée de main

Chaque état correspond à des vérifications effectuées, et l'envoi d'un message.

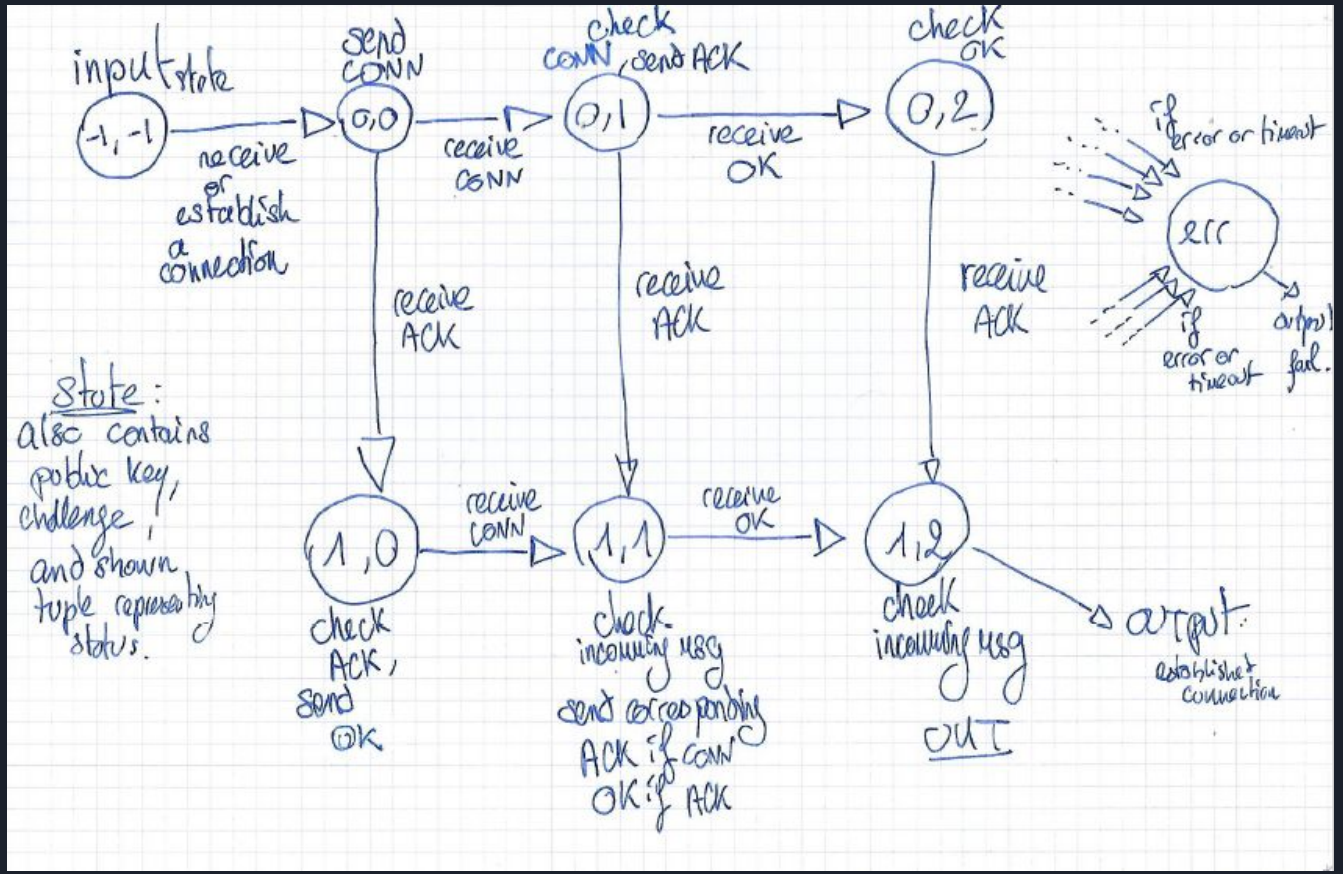
Chaque transition correspond à la réception d'un message.

Le state du Genserver contient l'état de l'automate, ainsi que des informations telles que les deux pubkey et les challenges.

Tout échec des vérifications ou timeout échu conduit à une erreur.

Une structure conditionnelle devra être mise en place dans deux états pour envoyer un message en fonction du message entrant.

Solution proposée



Format des messages

cf https://git.duniter.org/nodes/common/doc/-/blob/ws2p_v2/rfc/0004_ws2p_v1.md

En JSON pour récupérer les données, en RAW pour signer.

CONNECT : CONNECT, clé publique, challenge et signature

le challenge est le texte ajouté à la string à signer dans le ACK et le OK qui suivront

la signature est une signature ED25519 vérifiable (utilise une clé privée associée à la clé publique)

ACK: ACK, clé publique, et signature

la signature est réalisée grâce au challenge fourni dans le CONNECT

OK: OK, clé publique, signature

la signature est réalisée grâce au challenge fourni dans le CONNECT