Présentation du test technique

Sommaire

Liste de technos IPC

Pour le choix de la libraire/protocole utilisé pour l'IPC, voici une liste de quelques options envisagées triées dans un ordre de bas niveau vers haut niveau d'abstraction.

Named pipe

L'utilisation de named pipe UNIX est basique pour des IPC locales, mais ne permettra pas de solution en réseau. De plus, cette solution restant bas niveau, elle peut demander davantage de développements.

Socket TCP ou UDP

L'utilisation directement de sockets TCP ou UDP en C++ est possible, cela permet des IPC aussi bien local qu'en réseau. Cette solution bas niveau permet une grande flexibilité et optimisation, mais demande plus de développement.

boost.asio

L'utilisation des sockets peut être faite avec la libraire boost.asio. Cela permet de simplifier le développement d'une solution basé sur les socket TCP ou UDP. Cela ajoute une dépendance sur la librairie boost, mais cela reste modéré. Cependant, cette libraire reste encore bas niveau et nécessite d'implémenter manuellement le protocole de communication.

ZeroMQ

La libraire ZeroMQ permet de continuer d'utiliser le principe de socket TCP ainsi que d'autre solutions de transport (ipc, inproc, …), pour ses IPC ou des communication multi-thread. Elle peut donc être utilisée en local ou en réseau. De plus, elle offre un plus haut niveau d'abstraction que la librairie boost, car elle implémente plusieurs pattern request/reply, push/pull, publish/subscribe…
Il s'agit d'une solution implémentée en C++ avec des interfaces disponible dans divers langages.
Cette solution ne nécessite pas la mise en place de broker, ce qui simplifie sa mise en place, mais peut la rendre plus sensible à la perte de messages.
Enfin, elle n'implémente pas les protocoles SSL/TLS pour chiffrer et sécuriser les communications.

nanomsg NNg

La libraire nanomsg NNg est une solution proche de ZeroMQ avec une implémentation en C. Elle semble plus récente que ZeroMQ ce qui peut la rendre intéressante à essayer. Cependant, la documentation et communauté autour de celle-ci semblent encore limités. Elle permet d'implémenter le protocole TLS 1.2.

D-Bus

D-Bus est une solution logiciel pour la communication inter processus sous Linux. Cet outil est implémenté en C et est nativement disponible sur beaucoup de distribution Linux. Ce logiciel permet ainsi de bénéficier directement de messagerie déjà existante sous Linux par exemple avec systemd. Cependant, bien que cela soit possible, cette solution est moins adaptée pour des communications en réseau ou des OS différents comme Windows.

Protocole MQTT

Le protocole MQTT (couche application) permet l'utilisation de publish/subscribe à partir du protocole TCP. Il est principalement utilisé dans le monde de l'IoT. Il permet d'utiliser un applicatif léger pour les clients, mais nécessite la mise en place de broker. Enfin, le protocole supporte l'utilisation du protocole TLS pour sécuriser la messagerie.

Protocole AMQP

AMQP est un protocole (couche application) de messagerie qui permet l'utilisation de pattern de type publish/subscribe. Celui-ci nécessite la mise en place d'un broker, mais permettra une meilleure robustesse notamment à la perte de message. Enfin, il supporte le protocole TLS. Plusieurs implémentations sont possibles en C++, soit avec un broker RabbitMQ (AMQP-CPP), soit avec une solution Apache Qpid (Qpid Proton).

Framework gRPC

gRPC est un framework utilisé pour les Remote Procedure Call. Il se base sur le protocole HTTP et utilise Protobuf pour la sérialisation des données. Il permet l'utilisation d'appels simples ou de stream en mode synchrone ou asynchrone. Cette solution ne nécessite pas la mise en place de broker et supporte le protocole TLS. Le framework est directement disponible en C++, mais aussi dans beaucoup d'autre langages.

Choix de la techno IPC

Dans le cadre du test technique, afin d'avoir une solution qui peut rapidement être mise en place, je vais écarter les solutions trop pas niveau (TCP/IP direct, librairie boost).

L'objectif du test étant de faire un peu de développement plutôt que de la configuration de système, je vais écarter les solutions nécessitant la mise en place et configuration d'un broker (MQTT, AMQP).

D-Bus est un cas à trop particulier orienté local et Linux.

Le protocole gRPC bien qu'il offre des mécaniques de streaming qui pourrait être utilisées par le serveur afin d'envoyer ses métriques, cela reste un protocol RPC. Il faudra donc que les clients initient l'envoie des données via un appel. Cependant le sujet du test décrit d'avantage une mécanique de type publish/subscribe, ce qui ne correspond pas vraiment à une solution de type RPC.

Enfin entre ZeroMQ et nanomsg NNg, je vais privilégier l'utilisation de ZeroMQ pour 2 raisons. Tout d'abord, le projet est plus mature et offre une meilleure documentation pour une mise en place rapide. De plus, son implémentation en C++ plutôt qu'en C le rend plus proche de la description du test qui souhaite utiliser le C++. Enfin le sujet de suivi de métriques ne me semble pas critique et nécessiter la mise en place d'un protocole sécurisé type TLS.

J'ajoute ici que lors du choix de la techno, il est important également de vérifier la licence associée a chaque projet. Ici, dans ce cas, de ZeroMQ est sous licence MPL 2.0. Ce qui signifie que la librairie peut être utilisée dans des projets même commerciaux tel quel, cependant si des modifications sont apportées directement à la libraire, il faudra les publier sous la même licence.

Choix du format des messages

La libraire ZeroMQ ne fournit pas d'outils permettant la sérialisation/désérialisation de messages structurés. Il faut donc choisir une technologie en plus pour cela. J'ai opté pour une solution simple qui est l'utilisation du format Json. En effet, celle-ci offre plusieurs avantages, elle est très standard, simple à mettre en place en C++ avec la libraire nlohmann et surtout le format est directement lisible ce qui simplifie le debug dans le cadre du développement d'un petit projet rapide.

Cependant, d'autres options auraient pu être envisagées tel que MessagePack ou Protobuf. Ces deux solutions permettent de palier aux défauts du Json qui sont l'absence d'optimisation de la taille des messages et des performances moins bonne. Pour un projet plus complet et abouti, il serait sans doute recommandé de changer la méthode de sérialisation des messages pour un de ces outils.

Possibilités d’améliorations

L’objectif de ce test était de faire un petit développement C++ rapide. Si l’objectif est d’obtenir une vraie application aboutie, plusieurs axes d’améliorations sont envisageables.

Tout d’abord, avant de recréer une solution de zéro, il peut être intéressant de rechercher si des solutions déjà existantes sont disponibles, comme par exemple des applications telles que collectd, munin-monitoring,… Cela offre l’intérêt d’un développement plus abouti plus rapidement. Cependant, cela peut être plus limitant si les besoins sont trop spécifiques et évolutifs.

Sinon concernant le projet fournit en tant que tel, voici quelques pistes d’améliorations :

- - Mise en places de tests
- - Revoir s’il n’y a pas des gestions d’exceptions manquantes et meilleures gestion de celle-ci
- - Gestion de log plutôt que la sortie standard
- - Ajouter un fichier de configuration au serveur afin de sélectionner les informations que l’on souhaite publier ainsi que la fréquence de publication
- - Ajouter d’autre informations que le CPU et la RAM en fonction des besoins comme par exemple :
  - - Informations détaillé lié a un ou des processus spécifique important
  - - Informations sur l’espace et les accès disque.
  - - information sur les interruptions systèmes
  - - Autre information plus spécifique par exemple liée aux périphériques du serveur
- - En fonction des besoins cela peut être utile de mettre en place un stockage des données directement par le serveur si aucun client n’est connecté (pour une récupération ultérieure).
- - La partie affichage des données peut également être revue. Si l’analyse en temps réel est importante la solution basée sur gnuplot doit être revue par exemple en utilisant une libraire graphique C++ comme Qt.