Rapport d'intégration — Routeur Dynamique Rust

1. Présentation du programme

Ce projet implémente un protocole de routage dynamique en Rust, permettant à plusieurs routeurs de découvrir automatiquement les meilleurs chemins dans un réseau local. Le protocole repose sur l'échange périodique de messages de type "hello" entre voisins, la mise à jour des tables de routage et la gestion automatique des pannes et changements de topologie.

Lien du projet : https://github.com/Cessouille/rust-router

---

2. Protocole : Sécurité, Robustesse, Efficacité

Sécurité

- Le protocole n'accepte que les annonces provenant du réseau local (interfaces filtrées).
- Les messages sont sérialisés et désérialisés de façon stricte, limitant les risques d'injection ou de corruption.
- Les routes ne sont jamais annoncées en boucle grâce à la technique du split horizon (un routeur n'annonce pas à un voisin une route apprise de ce voisin).

Robustesse

- **Détection de panne** : Si un voisin ou un chemin n'est plus annoncé pendant 15 secondes, la route est supprimée automatiquement.
- **Gestion des liens** : Les interfaces de type loopback ou NAT sont ignorées pour éviter les faux positifs.
- **Mise à jour automatique** : Les changements de topologie (ajout/retrait de routeurs ou de liens) sont pris en compte sans intervention manuelle.

Efficacité

- **Calcul du plus court chemin** : Le protocole choisit toujours le chemin avec le moins de sauts (hops), avec tie-breaker sur l'adresse IP la plus basse.
- **Optimisation des annonces** : Seules les routes pertinentes sont annoncées à chaque voisin.
- **Suppression rapide des routes obsolètes** : Les routes non rafraîchies sont supprimées du système et de la table de routage.

---

3. Performances

Des métriques de performance sont enregistrées à chaque cycle du protocole dans un fichier `router_perf.log` :

- **Temps d'envoi des messages hello**
- **Temps de réception des messages hello**
- **Nombre de messages hello reçus**
- **Durée totale d'un cycle du protocole**

Ces logs permettent d'analyser la réactivité du protocole, la stabilité du réseau et la charge générée par le routage dynamique.
Des exemples de ces logs sont présentés ci-dessous :

```
========================== CYCLE ==========================
[25/06/2025 09:41:54] Time to send hellos: 369.26µs
[25/06/2025 09:41:56] Time spent receiving hellos: 2.01s
[25/06/2025 09:41:56] Number of hellos received: 2
[25/06/2025 09:41:56] Full protocol loop duration: 2.02s
```

---

4. Structure et points clés du code

- **Rust moderne** : Utilisation des threads, des mutex (`Arc<Mutex<...>>`) pour le partage d'état, et des timers précis (`Instant`).
- **Sérialisation** : Les messages sont échangés en JSON via UDP.
- **Gestion des interfaces** : Détection automatique des interfaces valides pour l'annonce et l'écoute.
- **Table de routage** : Mise à jour automatique de la table de routage système via des commandes `ip route`.
- **Logs** : Toutes les métriques de performance sont stockées dans un fichier, facilitant l'analyse a posteriori.

---

5. Conclusion

Ce routeur dynamique Rust offre une solution robuste, efficace et relativement sécurisée pour la gestion automatique des routes dans un réseau local.
Il s'adapte dynamiquement aux changements de topologie, tolère les pannes et fournit des outils de mesure pour évaluer ses performances en conditions réelles.