Faculté des sciences

L00 – Travail de session

Produit par :
Xavier Breton (23 201 795)
Philippe Lacasse (23 181 469)

Travail présenté à :
Patrice Roy

Dans le cadre du cours :
IFT 611 – Conception de systèmes temps réel

Courriels :
Xavier Breton : brex1001@usherbrooke.ca
Philippe Lacasse : lacp2116@usherbrooke.ca

Point de contact principal : Philippe Lacasse

Mise en contexte et utilité du projet

Le projet consiste à transformer un Raspberry Pi en un pare-feu matériel dédié à la sécurisation d'un réseau local. L'objectif est de positionner l'appareil entre le modem et les équipements du réseau afin de filtrer le trafic entrant et sortant. L'utilité principale est d'offrir une couche de protection supplémentaire pour les objets connectés domestiques, souvent vulnérables. L'intérêt d'intégrer des composantes de système temps réel dans ce projet réside dans la gestion de la latence. Un pare-feu ne doit pas seulement bloquer les menaces, il doit le faire de manière déterministe pour éviter que le traitement des paquets ne devienne une source d'instabilité pour les flux sensibles comme la voix sur IP ou le jeu en ligne. Pour ce faire, nous utiliserons un Raspberry Pi 4 Model B.

Contraintes temps réel et stratégies de test

La contrainte principale est la latence de traversée: l'inspection de chaque paquet doit se faire dans une fenêtre de temps prévisible, sous la barre des 1 ms. En cas de dépassement, le système adopte un comportement de fail-safe consistant à bloquer le paquet concerné et à enregistrer l'événement. Cette approche garantit la stabilité temporelle du système et permet une validation a posteriori du respect des contraintes par analyse des journaux. Un non-respect de cette contrainte entraînerait une gigue excessive, rendant la connexion instable. Une autre contrainte concerne la priorité des flux, où les paquets de gestion du système doivent passer avant les flux de données massifs pour garantir que l'administrateur garde le contrôle, même en cas de congestion. Pour valider ces contraintes, nous utiliserons des outils comme `hping3` pour mesurer le temps de réponse précis sous différentes conditions. Nous testerons le système en situation de charge, par exemple en simulant un téléchargement intensif tout en vérifiant si la latence des paquets prioritaires reste stable. Ces tests permettront de confirmer que le pare-feu réagit de manière représentative à un usage réel.

Sécurité, qualité et suivi

La qualité du système repose sur sa capacité à rester opérationnel malgré les aléas. En cas de plantage des services de filtrage, un mécanisme de surveillance (watchdog) sera mis en place pour redémarrer automatiquement les règles de sécurité. Pour assurer le respect des contraintes à long terme, nous prévoyons des journaux de performance qui documenteront l'impact de chaque nouvelle règle ajoutée. Cela évitera qu'une complexification du filtrage ne finisse par briser les exigences de vitesse d'exécution initiales.

Livrable L01 : Preuve de concept et analyse des performances temporelles

Le livrable L01, prévu pour la mi-chemin, servira de preuve de concept technique centrée sur la capacité de transit déterministe.

- **Isolation matérielle** : Nous mettrons en place l'isolation d'un cœur de processeur (via le paramètre noyau `isolcpus`) afin de lui dédier exclusivement le traitement des interruptions réseau et du chemin critique de traitement des paquets.
- **Journalisation :** Les opérations de journalisation, intrinsèquement non déterministes en raison des accès E/S et du formatage, seront strictement séparées du chemin critique. Elles seront exécutées sur un autre cœur de processeur non isolé, via un thread dédié de priorité inférieure. Cette séparation garantit que les activités de logging ne peuvent en aucun cas interférer avec les contraintes temporelles du traitement réseau.
- **Infrastructure de test :** Utilisation d'un générateur de trafic pour injecter des flux synthétiques. L'objectif est de documenter la latence de base (*baseline*) et d'identifier la gigue (variation de latence) induite par le matériel.
- **Mesures TR initiales :** Ce livrable présentera des histogrammes de latence (générés par `cyclictest`) permettant de valider la stabilité du processeur isolé. En complément, nous effectuerons des tests de transit "à vide" avec `hping3` pour établir la latence de référence du réseau sans filtrage. Cette double approche permettra de confirmer que l'infrastructure matérielle est prête à supporter la logique métier tout en respectant notre cible de déterminisme.

Livrable L02 : Produit fini et ordonnancement déterministe

Le livrable L02 démontrera la capacité du système à agir comme un pare-feu actif sous des contraintes temporelles strictes. Le moteur de filtrage s'appuiera sur une approche de liste blanche, structurée de manière séquentielle pour permettre l'analyse du pire cas :

- **Structure logique de la table de filtrage (ACL) :**
  - Règle 1 (Priorité haute) : Inspection d'état (*Stateful*) pour autoriser immédiatement les sessions déjà établies.
  - Règles 2 à 999 : Autorisations spécifiques basées sur les adresses IP et les ports (ex. : accès IoT, consoles de jeu, serveurs locaux).
  - Règle 1000 (Défaut) : Blocage systématique de tout trafic non répertorié.
- **Filtrage et blocage dynamique (WCET) :** Le système analysera les en-têtes de paquets. Nous porterons une attention particulière au `WCET` (*Worst-Case Execution Time*), soit le temps de traitement le plus long observé lorsqu'une règle complexe est déclenchée (Le paquet est conforme à aucune règle). La contrainte est de maintenir ce traitement sous les 1 ms, peu importe la taille de la table de filtrage.
- **Régulation du débit (Rate Limiting) et Priorisation :** Si une anomalie (DoS) est détectée, le système doit réagir immédiatement. Nous utiliserons l'ordonnancement temps réel de Linux (`SCHED_FIFO` ou `SCHED_RR`) pour que le processus de régulation ait toujours la priorité sur les tâches non critiques.

Produit final

Le produit final consiste en un nœud réseau actif basé sur un Raspberry Pi 4, agissant comme pont filtrant entre deux zones réseau tout en garantissant un respect strict des contraintes temporelles.

1. Spécifications techniques

- **Plateforme matérielle :** Raspberry Pi 4 Model B.
- **Environnement logiciel :** Linux (Raspberry Pi OS) avec patch PREEMPT_RT (pour le support de l'ordonnancement SCHED_FIFO et meilleur contrôle du noyau).
- **Isolation matérielle :** Utilisation de l'argument noyau `isolcpus` pour dédier un cœur de processeur exclusivement au traitement des paquets.
- **Gestion des interruptions :** Configuration de l'affinité matérielle (`smp_affinity`) pour diriger les interruptions réseau vers le cœur isolé.
- **Ordonnancement :** Application de politiques temps réel pour protéger le chemin critique des tâches de fond (logging, maintenance).
- **Mécanisme Fail-Safe :** Blocage automatique du paquet et journalisation si le temps de traitement excède le seuil critique de 1 ms.

2. Fonctionnalités et validation

- **Déterminisme du filtrage :** Garantie d'un temps d'exécution au pire cas (WCET) stable, indépendamment de la charge système globale.
- **Priorisation des flux :** Isolation de la bande passante pour le trafic de gestion (SSH/Admin) afin de maintenir le contrôle lors de congestions ou d'attaques DoS.
- **Résilience (Watchdog) :** Mise en œuvre d'un watchdog logiciel assurant le redémarrage immédiat des services de sécurité en cas de défaillance.
- **Analyse de performance :** Production d'histogrammes de latence comparatifs via `cyclictest` et `hping3` pour prouver la stabilité du système sous charge.

Séparation du travail

Xavier Breton

Livrable L01 :

- **Configuration noyau :** Implémentation de l'isolation de cœur (`isolcpus`) et paramétrage du démarrage du Raspberry Pi.
- **Tests de stabilité :** Exécution des mesures `cyclictest` et production des histogrammes de latence brute.
- **Administration :** Création et tenue du registre des fichiers et des auteurs.

Livrable L02 :

- **Structure de filtrage :** Implémentation technique de la table de filtrage (ACL) et optimisation de la recherche de règles.
- **Sécurité matérielle :** Mise en œuvre et test du Watchdog logiciel pour la relance des services en cas de panne.
- **Documentation :** Rédaction des procédures de déploiement et de configuration système.

Philippe Lacasse

Livrable L01 :

- **Infrastructure réseau :** Configuration des interfaces de transit et mise en place de la topologie de test (pont réseau).
- **Baseline réseau :** Exécution des tests de latence "à vide" (sans logique métier) avec `hping3` pour établir la référence de traversée.
- **Administration :** Coordination des rencontres et rédaction des procès-verbaux (PV).

Livrable L02 :

- **Développement temps réel :** Implémentation des règles de filtrage dynamique et gestion des priorités via `SCHED_FIFO`.
- **Tests de charge :** Simulation de trafic intensif et validation du respect du WCET (Worst-Case Execution Time) sous la barre de 1 ms.
- **Documentation :** Analyse des résultats de performance et rédaction du rapport final d'analyse TR.