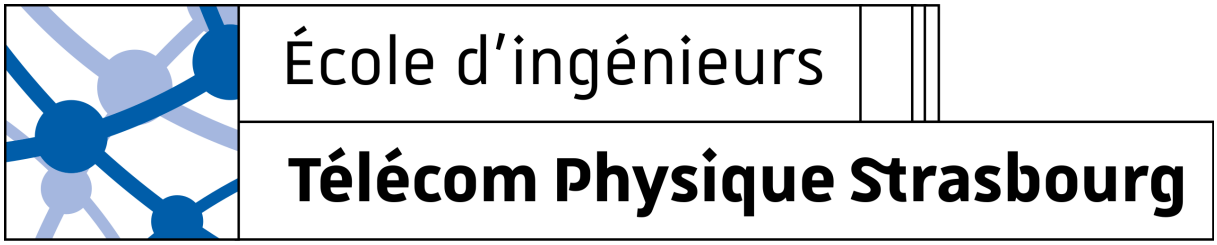
Compte Rendu de Projet

Projet RIO

21 mai 2023



Auteurs : Julien Brelot, Johan Verdon, Bergamini Enzo

Table des matières

1 Introduction

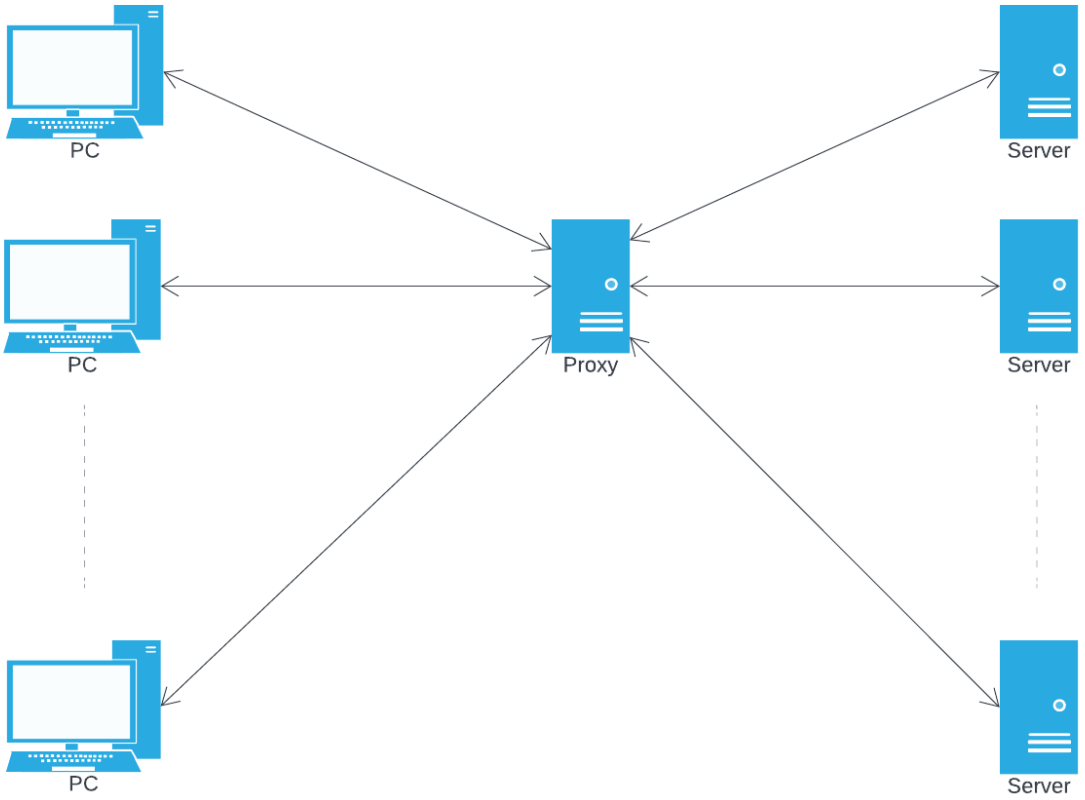
Ce rapport décrit notre réalisation de projets dans le cadre du projet de découverte de la filière *Réseau et Internet des Objets* au sein de l'école d'ingénieurs *Télécom Physique Strasbourg*. Notre projet porte sur la conception d'un échange de texte avec correction d'erreurs de transmission, en utilisant un *reverse proxy* comme nœud intermédiaire.

L'objectif principal de notre projet était de mettre en place un système de communication fiable entre un émetteur et un récepteur *TCP (Transmission Control Protocol)*, en intégrant un proxy inversé pour brouiller les données et détecter/corriger les erreurs introduites lors de la transmission.

Nous avons développé notre solution en utilisant le langage *C* et la bibliothèque *socket*.

2 Communication réseau

Les échanges et retransmissions fonctionnent de manière similaire à TCP. Le serveur acquitte chaque paquet non corrompu et déplace la fenêtre d'envoi des données en conséquence. Si un acquittement négatif est reçu ou si le temps d'attente pour l'acquittement du dernier paquet envoyé est dépassé, les données de la fenêtre courante sont renvoyées. Si un certain nombre de tentatives de renvois sont dépassées, l'utilisateur a la possibilité d'envoyer un nouveau message, tout en étant informé que toutes les données n'ont peut-être pas été envoyées.



2.1 Fonctionnement du client

Le client se connecte au proxy en utilisant le protocole TCP. Si le programme n'arrive pas à se connecter, il réessaie x fois avec un délai d'attente entre chaque essai. L'authentification se fait en communiquant le pseudo au proxy, qui se prépare à recevoir des paquets

du client. Ensuite, l'utilisateur est invité à saisir le nom d'un client ou l'adresse IP d'un client avec qui il souhaite communiquer. Le chat commence alors, sans bloquer l'interface pour permettre à l'utilisateur de saisir une commande pour quitter le programme. Lorsqu'on quitte, le proxy est informé pour qu'il prévienne l'autre client.

2.2 Fonctionnement du serveur

Le serveur écoute sur un port spécifié en entrée (par défaut, le port `DEFAULT_EXCHANGE_PORT_SERVER`). Il attend une unique entité (le proxy) et accepte la connexion. Le serveur vérifie l'intégrité des données dans le paquet et les corrige si possible. Il envoie un acquittement au client ayant envoyé le paquet si les données sont intègres ou corrigées, puis achemine le paquet vers le destinataire en le renvoyant vers le proxy. Si les données ne sont pas intègres, un paquet du type `DT_NAK` est envoyé à l'expéditeur. Chaque paquet entre via le proxy et chaque paquet sortant est renvoyé vers le proxy.

2.3 Fonctionnement du proxy

Le proxy prend en entrée un fichier contenant une liste de serveurs au format `adresseIP:portEcoute`
Par exemple :

```
8.8.8.8:80
1.1.1.1:53
...
```

Le proxy est configuré pour se connecter à plusieurs serveurs, chaque serveur ayant une charge. Lorsqu'un client se connecte, il est attribué au serveur ayant la charge la moins élevée à ce moment-là.

Le proxy introduit un bruit dans chaque paquet d'échange (`DT_CEX` ou `DT_FEX`) pour éventuellement corrompre les données. Il effectue également un filtrage en gardant en mémoire les clients avec leur pseudo et le serveur par lequel transitent leurs envois. Lorsqu'un client indique qu'il souhaite arrêter un échange de messages ou mettre fin à la connexion, le proxy informe directement le client destinataire d'également arrêter.

3 Détection et correction d'erreurs

Nous avons réalisé une implémentation basée sur les codes polynomiaux CRC (Cyclic Redundancy Check). Ces codes jouent un rôle essentiel dans la détection d'erreurs de transmission dans les systèmes de communication.

Les codes polynomiaux CRC sont des algorithmes de contrôle d'intégrité qui permettent de détecter les erreurs lors de la transmission de données. Ils sont largement utilisés dans de nombreux protocoles de communication, notamment dans les réseaux informatiques et les transmissions sans fil.

Le principe des codes polynomiaux CRC repose sur la division polynomiale des données transmises par un polynôme générateur prédéfini. Cette opération génère un code de vérification qui est transmis avec les données. Le récepteur effectue ensuite la même division polynomiale et vérifie si le reste obtenu correspond au code de vérification reçu. Si les deux codes correspondent, cela indique que les données ont été transmises sans erreur. En revanche, si les codes diffèrent, cela indique qu'une ou plusieurs erreurs se sont

produites lors de la transmission.

3.1 Implémentation des codes polynomiaux CRC

Dans notre projet, nous avons choisi d'utiliser le *CRC-8* comme algorithme de détection d'erreurs pour nos transmissions de données par paquets de 8 bits. Nous avons sélectionné le polynôme générateur `0xD5` pour notre implémentation.

$$P(x) = x^8 + x^7 + x^6 + x^4 + 1 \tag{1}$$

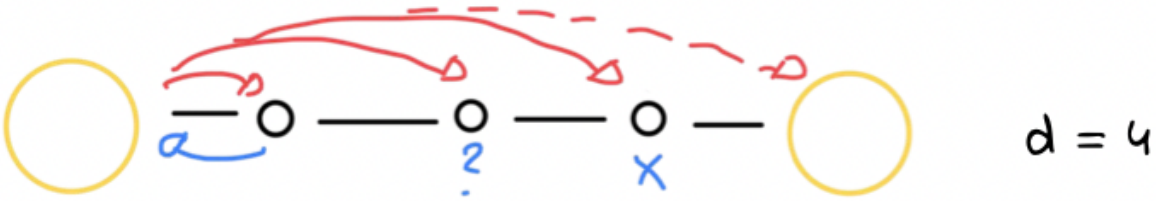
Ainsi, notre paquet de données fait 16 bits, avec les 8 bits de poids fort qui représentent le message et les 8 bits de poids faible qui correspondent au *CRC-8*.

En choisissant le *CRC-8* avec le polynôme `0xD5`, nous avons opté pour un algorithme de détection d'erreurs efficace pour les transmissions par paquets de 8 bits. Ce polynôme a été spécifiquement choisi pour ses bonnes propriétés de détection d'erreurs.

3.2 Performance de la méthode

Pour évaluer les performances de détection et de correction, nous avons utilisé une évaluation de la distance de Hamming minimale. Cette méthode permet de quantifier les performances d'un code linéaire en utilisant les distances de Hamming. La distance de Hamming mesure la différence entre deux mots de code en comptant le nombre de positions où les bits diffèrent. Dans notre contexte, la distance de Hamming minimale représente la quantité d'erreurs que notre code est capable de détecter ou de corriger.

Plus précisément, la distance de Hamming minimale d'un code linéaire est définie comme la plus petite distance de Hamming entre tous les paires de mots de code différents du code. Elle représente le nombre minimum d'erreurs qu'un code peut détecter. Dans notre cas, la distance de Hamming minimale est de 4, ce qui signifie que notre code peut détecter jusqu'à 3 erreurs et corriger une erreur unique.



3.3 Limites de la méthode

Bien que les codes polynomiaux offrent une méthode pratique et efficace pour la détection d'erreurs, ils ont certaines limites en termes de capacité de détection, de types d'erreurs couverts et de dépendance du choix du polynôme. Il est important de comprendre ces limites et de les prendre en compte lors de la conception et de l'utilisation de codes polynomiaux pour la détection d'erreurs.

4 Conclusion

En conclusion, notre projet de conception d'un échange de texte avec correction d'erreurs de transmission à l'aide d'un reverse proxy a été une réussite. Nous avons réussi à

mettre en place un système de communication fiable entre un émetteur et un récepteur TCP en utilisant un proxy pour injecter des erreurs. Notre code linéaire basé sur un modèle convenu a démontré sa capacité à détecter et corriger un simple bit par tranche de 8 bits.