UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

RAPPORT DE L'APP 3

Réseaux et protocoles de communication
GIF332

Présenté à
Domingo Palao Munoz

Présenté par
Pascal-Emmanuel Lachance – LACP3102
Yan Ha Routhier-Chevrier - ROUY2404

Site web S3 – Sherbrooke – 19 octobre 2021

Table des matières

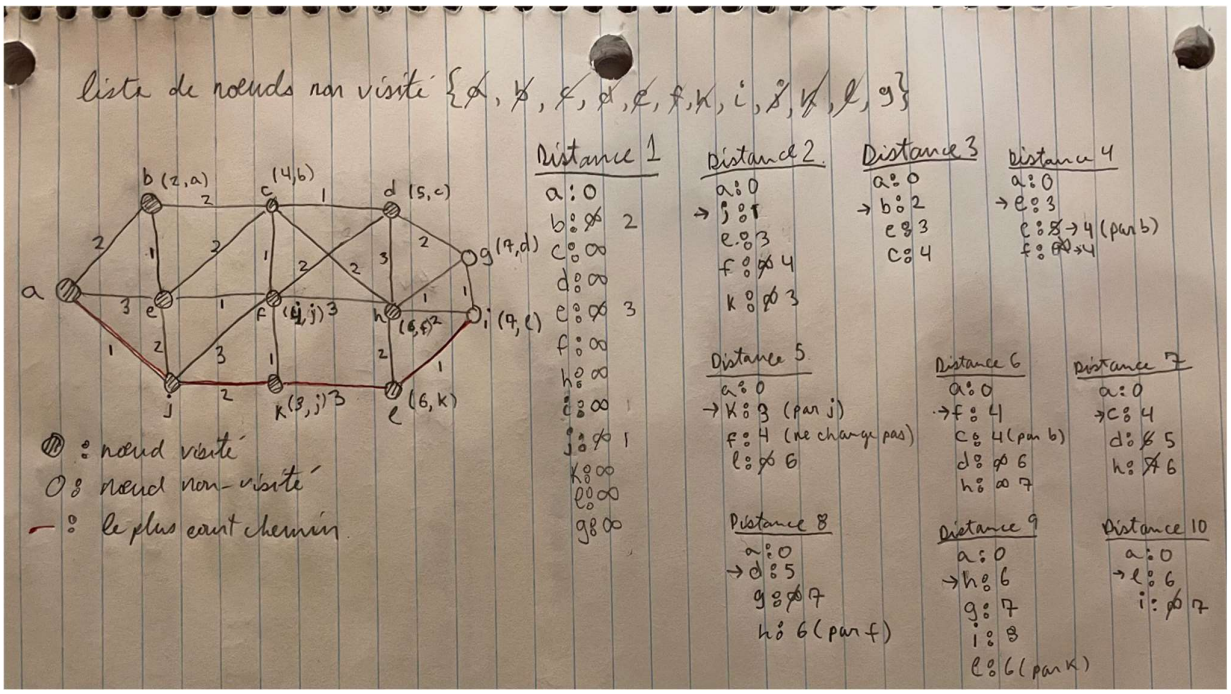
LISTE DES FIGURES

1 ROUTAGE

Le chemin pris pour le routeur de l'entreprise devrait être un coût de *7*, lequel est le plus petit coût que nous avons obtenu durant l'analyse. En effet, le chemin pris selon l'algorithme de *Dijkstra* est par les nœuds **(J)**, suivi de **(K)**, suivi de **(L)** et suivi de **(i)**. Selon nous, il serait mieux d'avoir un routage par vecteur de distance. Lequel pourrait s'ajuster au fil du temps; si nous ajoutons un routeur, si le coût de certains chemins change. Ayant un petit réseau le routage par vecteurs de distances ne prendra pas beaucoup de temps pour effectuer l'analyse des chemins.

1.1 ÉTAPE DE L'ALGORITHME DE DIJKSTRA

1. Mettre les chemins vers tous les autres nœuds à un coût ∞, à part le nœud de départ.
2. À partir du nœud de départ, il faut regarder les nœuds voisins et changer le coût de chemin par le coût le plus petit.
3. Par la suite nous recommençons l'étape 2 en partant par les nœuds voisins, débutant par le nœud le plus proche du nœud du départ, sans revisiter les nœuds déjà visités.



1.2 COMPARAISON DES SOLUTIONS DE ROUTAGE

1.2.1 Routage RIP

a) Extensibilité de la solution si l'on ajoute des routeurs

Il est possible d'ajouter des routeurs sur un réseau impliquant le protocole RIP. Cependant, il y a une limite réglée pour respecter le saut maximum, lequel est de 15. Pour un saut de 16, celui-ci indique que le routeur est inaccessible. Alors il fonctionne bien dans un réseau à petite taille, mais ne supporte pas les réseaux plus grands. Il peut venir utile d'utiliser le protocole RIP sur des réseaux avec routage statiques, par son ancienneté il est beaucoup documenté, donc plus facile à implémenter. De plus, les routeurs convergent périodiquement, faisant en sorte qu'en ajoutant des routeurs nous n'avons pas besoin de changer les routes à main.

b) Tolérance aux pannes

Le protocole a une convergence lente, faisant en sorte que lors d'une panne il sera plus lent pour se reconfigurer pour retrouver recalculer les chemins les plus courts. Il serait toujours mieux d'utiliser RIP, pour être plus tolérant aux pannes si jamais nous sommes sur un réseau à routage statique.

c) Facilité de configuration des routeurs pour privilégier une route plutôt qu'une autre.

La configuration est faite simplement par l'échange périodique des informations des routages, permettant la mise à jour des routes. Dans ce protocole, tous les routeurs font confiance à ses routeurs voisins. Utiliser le protocole RIP, est plus simple à utiliser pour la configuration des routeurs que le faire à la main chaque fois que la structure du réseau change.

1.2.2 Routage OSPF (Open Shortest-Path First)

a) Extensibilité de la solution si l'on ajoute des routeurs

Lorsque nous ajoutons des routeurs dans réseaux utilisant le OSPF, les routeurs sur le réseau connaîtra son existence en peu de temps. Il est utilisé par beaucoup, car il peut être utilisé dans de larges réseaux locaux, ainsi que complexes. Donc, le protocole ne limite pas le nombre de routeurs dans le réseau comme le ferait le protocole RIP.

b) Tolérance aux pannes

Le protocole à comme caractéristique d'enregistrer tous les coûts liés pour toutes les destinations différentes sur le réseau. Ainsi, chaque routeur maintient leur enregistrement à jour. En effet, les routeurs savent en tout temps l'état de ses voisins et le partage à leur voisin et inversement. Soulignant une caractéristique importante du protocole, celui de la détection des changements à la topologie du réseau. Permettant l'adaptation rapide lors des pannes, ayant la convergence efficiente sa tolérance aux pannes est beaucoup plus grande que le routage statique et même le protocole RIP, lequel est aussi à routage dynamique.

c) Facilité de configuration des routeurs pour privilégier une route plutôt qu'une autre.

En 1990 le protocole OSPF eu été normalisé, mis sur pied après le protocole RIP, il eut été conçu pour les problèmes plus modernes. Soit afin de soutenir un environnement plus dynamique et tendance à agrandir. De plus, les paramètres que prend le protocole pour calculer le coût sont une caractéristique importante. Car pour ce calcule peut prendre en charge la distance physique, le délai de transmission, le trafic, etc. Et donc, il est dynamique et peut facilement changer la configuration afin de privilégier les routes les plus courtes. Ce qui n'est pas le cas pour un réseau à routage statique et qui est mieux que le RIP lequel prend qu'un seul paramètre pour trouver le chemin pour le routage.

1.3 RECOMMANDATION

L'entreprise à présentement un total de 12 routeurs, soit un routeur d'entreprise, un routeur qui est la passerelle par défaut et 10 autres. Si nous prenons le routage statique, il faudra tenir à jour le routage chaque fois que la topologie du réseau changera, ce qui n'est plus réellement recommandé dans nos jours modernes. En revanche, en implémentant le protocole RIP, nous gagnons sur le plan dynamique de la topologie, ainsi nous pouvons ajouter, ou enlever un routeur avec plus grande aise. Cependant, le protocole RIP présente une limite soit un maximum de 15 sauts sur le réseau, ce qui est problématique, surtout dans notre cas, car nous sommes déjà à un total de 12 routeurs sur le réseau. Nous devons garder la possibilité que l'entreprise grandisse et que nous ayons besoin d'ajouter des routeurs à ce dernier et que le nombre puisse facilement dépasser la limite de saut. Ce sont les raisons pour lesquelles nous supportons l'usage du protocole OSPF, lequel pourrait subvenir aux besoins dans le long terme de l'entreprise. Ne permettant aucune limite connue et, de plus, il a une très grande tolérance aux pannes compte tenu à ça convergence rapide et ça topologie adaptative.

2 DÉTOURNEMENT DE LA SESSION

2.1 ANALYSE DES PAQUETS

2.1.1 PD1

Le paquet 1, nous avons une requête ARP pour demander qui aurait l'adresse IP 192.168.1.1, cette demande a été fait par la MAC adresse F8 B1 56 A4 64 50, que nous avons identifié Dell1, lequel a aussi l'adresse IPv4 192.168.1.2. Le paquet est un paquet normal pour une requête ARP.

```
0000   FF FF FF FF FF FF F8 B1 56 A3 64 50 08 06 00 01
0010   08 00 06 04 00 01 F8 B1 56 A3 64 50 C0 A8 01 02
0020   00 00 00 00 00 00 C0 A8 01 01

Ethernet ---------------
Addresse Destination: FF FF FF FF FF FF -> Broadcast
Addresse Source: F8 B1 56 A4 64 50 -> Dell1

Type: 08 06 -> ARP

ARP ----------------------
Hardware Type: 00 01
Protocol Type: 08 00
Hardware Address Length: 06
Protocol Address Length: 04 -> IPv4
Operation Code: 00 01 -> Request

Source Hardware Address: F8 B1 56 A3 64 50 -> Dell
Source IPv4 Address: C0 A8 01 02 (192.168.1.2)
Target Hardware Address: 00 00 00 00 00 00 -> Normal pour du ARP
Target IPv4 Address: C0 A8 01 01 (192.168.1.1)
```

2.1.2 PD2

Le paquet 2 est la réponse ARP de la part du périphérique situé au 192.168.1.1, que nous avons identifié Dell2. Le destinataire est le Dell1 pour la réponse située au 192.168.1.2.

```
0000   F8 B1 56 A3 64 50 F8 B1 56 A3 59 E0 08 06 00 01
0010   08 00 06 04 00 02 F8 B1 56 A3 59 E0 C0 A8 01 01
0020   F8 B1 56 A3 64 50 C0 A8 01 02
```

```
Ethernet ------------
Address Destination: F8 B1 56 A4 64 50 -> Dell1
Address Source: F8 B1 56 A3 59 E0 -> Dell2

Type: 08 06 -> ARP

ARP ---------------
Hardware Type: 00 01 -> Internet
Protocol Type: 08 00 -> IPv4
Hardware Address Length: 06
Protocol Address Length: 04 -> IPv4
Operation Code: 00 02 -> Reply

Source Hardware Address: F8 B1 65 A3 59 E0 -> Dell2
Source IPv4 Address: C0 A8 01 01 (192.168.1.1)

Target Hardware: F8 B1 56 A4 64 50 -> Dell1
Target IPv4 Address: C0 A8 01 02 (192.168.1.2)
```

2.1.3 PDU 3

Le paquet 3 est la communication TCP entre le Dell2 et un 3 IIème acteur ayant comme adresse MAC F8 B1 56 A5 90, lequel nous avons assigné le nom Dell3.

```
0000   F8 B1 56 A5 90 E1 F8 B1 56 A3 59 E0 08 00 45 00
0010   00 28 01 A1 40 00 80 06 77 DA C0 A8 01 01 C0 A8
0020   01 03 0B DF 04 1E 68 FA A3 6F 4A 5B EF 58 50 10
0030   44 6B 93 F9 00 00

Ethernet -----------
Address Destination: F8 B1 56 A5 90 E1 -> Dell3
Address Source: F8 B1 56 A3 59 E0 -> Dell2

Type: 08 00 -> IPv4

IPv4 -----------------
Version: 4 -> IPv4
LET: 5 -> (20 bytes)
Services Différenciés: 00
Longueur Totale: 00 28

Identification: 01 A1
DF : set -> Dont fragment
Position de fragment: 00 00
Durée de vie : 80
Protocole : 06 -> TCP
Total contrôle ET : 77 DA

Address Source : C0 A8 01 01 (192.168.1.1) Dell2
Address Destination : C0 A8 01 03 (192.168.1.3) Dell3

TCP ----------------
Port Source: 0B DF -> :3039
Port Destination : 041E -> :1054
# sequence : 68 FA A4 6F
# ack: 4A 5B EF 58
Longueur header: 5
010 -> 0000 0001 0000
ACK: 1

Taille de Fenêtre: 44 6B -> 17515
Total Contrôle: 93 F9
Pointeur Urgent : 00 00
```

2.1.4 PDU 4

On peut voir qu'il a y une communication avec un adresse IP externe soit 132.210.74.162, transmet un message « How are you? » dans le data. On peut croire que le Dell2 est la passerelle par défaut, car c'est lui qui transmet au Dell3 le message de l'IP distant.

```
0000   F8 B1 56 A5 90 E1 F8 B1 56 A3 59 E0 08 00 45 00
0010   00 34 01 DC 40 00 80 06 68 C2 84 D2 4A A2 C0 A8
0020   01 03 0B E1 04 20 6F F5 0F 95 51 56 5F 36 50 18
0030   44 6B 06 46 00 00 68 6F 77 20 61 72 65 20 79 6F
0040   75 3F

Ethernet ---------------

Destination: F8:B1:56:A5:90:E1      -> Dell3
Source: F8:B1:56:A3:59:E0           -> Dell2

Type: 08 00 -> IPv4


IPv4 ----------------
Version: 4
LET: 5 -> (20 bytes)
Services différenciés: 00
Longueur totale: 00 34

Identification: 01 DC
MF: set -> More fragments
Position de fragment: 00 00
Durée de vie: 80
Protocole: 06 -> TCP
Total Contrôle: 68 C2

Address Source: 84 D2 4A A2 -> 132.210.74.162
Address Destination: C0 A8 01 03 -> 192.168.1.03

TCP --------------------
Port Source: 0B E1 -> 3041
Port Destination: 04 20 → 1056

# seq: 6F F5 0F 95
# ack: 51 56 5F 36
Longueur header: 5
0101 0000 0001 1000
ACK: 1
PSH: 1      -> Send reply immediately on reception

Window size: 44 6B
Total Contrôle: 06 46
Pointeur Urgent: 00 00

Data ---------------
68 6F 77 20 61 72 65 20 79 6F 75 3F
ASCII: how are you?
```

2.2 ANALYSE DU FICHIER DE CAPTURE

Nous pensons que l'attaquant a usurpé l'adresse IP du client. Dans ce cas-ci, un client (00 :00 :c0 :29 :36 :e8) communique avec un serveur (00 :06 :5b :d5 :1e:e7) via protocole TELNET, et un attaquant (00 :01 :03 :87 :a8 :eb) prend l'adresse IP du client (192.168.1.103), et envoie un message au serveur (00 :06 :5b :d5 :1e:e7), en s'insérant exactement à la bonne séquence #233 [#461] (si le # de séquence était trop bas, l'attaquant (00 :01 :03 :87 :a8 :eb) recevrait un acknowledge du serveur (00 :06 :5b :d5 :1e:e7), indiquant que ce dernier a déjà reçu le message, si le # de séquence était trop haut, le serveur (00 :06 :5b :d5 :1e:e7) ne recevrait pas les messages dans le bon ordre et recevrait une requête du serveur (00 :06 :5b :d5 :1e:e7) pour les paquets manquants).

Lorsque le client (00 :00 :c0 :29 :36 :e8) envoie le vrai paquet #233 au serveur [#463] (00 :06 :5b :d5 :1e:e7), ce dernier indique au client (00 :00 :c0 :29 :36 :e8) qu'il a déjà reçu le paquet #233 [#464] et qu'il attend le paquet #243. Le client (00 :00 :c0 :29 :36 :e8) ne s'attend pas à cette réponse et renvoie le paquet #233 à nouveau [#465], ce qui répète la séquence indéfiniment. L'attaquant (00 :01 :03 :87 :a8 :eb) a maintenant le contrôle de la communication vers le serveur (00 :06 :5b :d5 :1e:e7).

En prenant le contrôle du serveur (00 :06 :5b :d5 :1e:e7), l'attaquant (00 :01 :03 :87 :a8 :eb) commence par effacer le contenu de la ligne de commande actuelle avec des *\b* [#461], puis insère d'un coup *echo HACKED* dans le profil du client (00 :00 :c0 :29 :36 :e8) dans le serveur [#656] (00 :06 :5b :d5 :1e:e7).

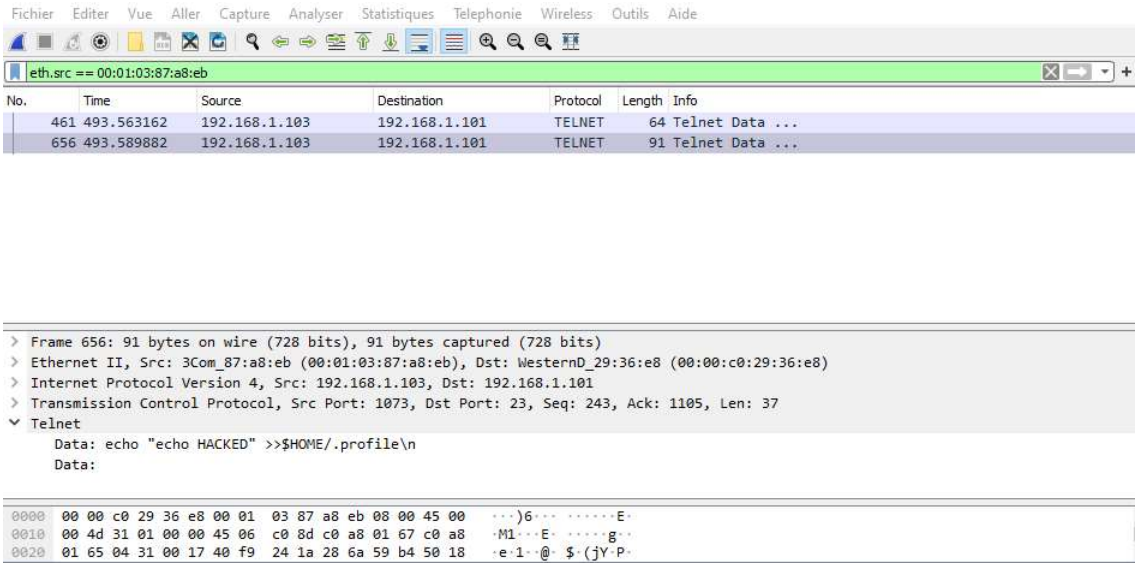


Figure 1 - Vue des messages de l'attaquant

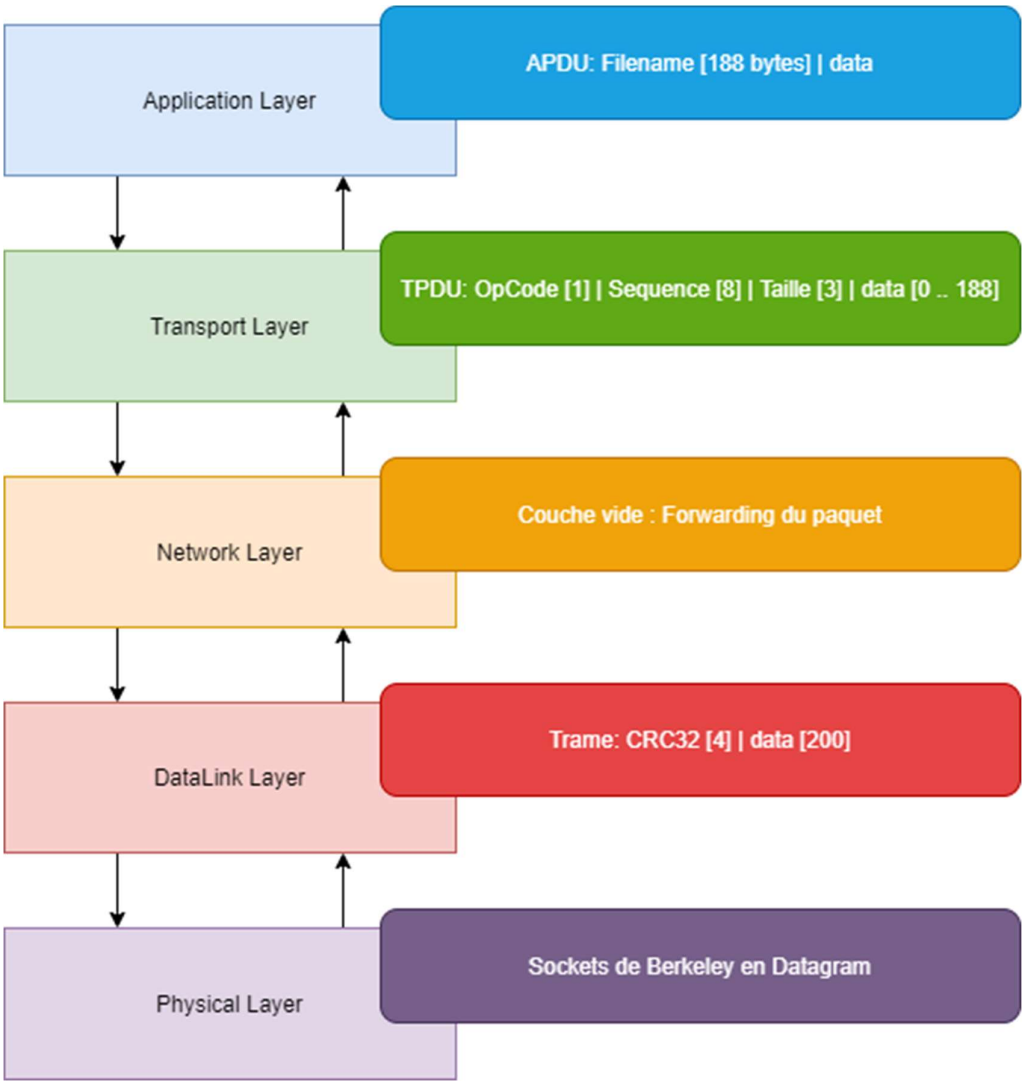
```
YPBINDPROC_DOMAIN: Domain not bound
YPBINDPROC_DOMAIN: Domain not bound
..... ..#..'........ ..#..'..........P.............ANSI............!...........!......
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.14-5.0 on an i686
...login: ...cclluueelleessss

Password: madeULook

YPBINDPROC_DOMAIN: Domain not bound
Last login: Sat Apr 13 16:23:56 from 192.168.1.103
This computer is reserved for use by the Spinglass group.
If you are not a Spinglass group member, please use one of the department's
public servers instead.

[clueless@heineken clueless]$ ccaatt  >>  ttooppSSeeccrreett..[D .[D..[D .[Deett

BBooyy  II''mm  gglladd  nnoo  oonnee  iiss  wwaattcchhiinngg  mmee  ttyyppee

tthhiiss  ttoopp  sseeccrreett  ffiillee

.[clueless@heineken clueless]$

[clueless@heineken clueless]$

[clueless@heineken clueless]$

[clueless@heineken clueless]$

[clueless@heineken clueless]$

[clueless@heineken clueless]$ II  aamm  ttyyppeeiinngg..[D .[D..[D .[D..[D .[D..[D .[Diinn..[D .[D..[D .[D..[D .[D..[D .[D..[D .[D..[D .[D..[D .[D..[D .
[D..[D .[D..[D .[D..hh..[D .[Dccaa..[D .[D..[D .[Dllss

.[0m.[0mtopSecret.[0m
.[m[clueless@heineken clueless]$ ccaatt..[D .[Dtt  ttooppSSeecctt..[D .[Drreett

Boy I'm glad no one is watching me type
this top secret file
[clueless@heineken clueless]$ ll..[D .[Dcc........

echo "echo HACKED" >>$HOME/.profile
..[D .[D.......
[clueless@heineken clueless]$
[clueless@heineken clueless]$
```

Figure 2 - Vue des transmissions client/attaquant (rouge) - serveur (bleu)

Avec le mot de passe en clair.

3 PROTOCOLE DE TRANSFERT DE FICHIERS

3.1 DESCRIPTION DES COUCHES



Les couches de notre réseau sont implémentées sous un modèle de Chain of Responsibility de Singletons. La couche Physique est un wrapper autour des sockets de Berkeley, recevant dans un thread d'écoute et envoyant de façon bloquante. La couche de DataLink utilise un CRC32 utilisant le polynôme de la norme 802.3 autant dans la transmission que la réception; à la réception, si le CRC32 est erroné, la trame est simplement droppée. La couche de Network n'est qu'un forwarding vide entre la couche DataLink et Transport.

La couche de transport est la plus étoffée, c'est dans celle-ci qu'en transmission, le fichier est séparé en morceaux de maximum 188 octets, auxquels on rajoute 12 octets d'en-tête humainement lisible, contenant un code d'opération (par exemple 'd' pour DÉBUT, 'a' pour ACKNOWLEDGE ou 'r' pour

RETRANSMIT), un numéro de séquence encodé en ASCII ainsi que la quantité de données (généralement 188 octets) également encodées en ASCII. À la réception, les paquets sont emmagasinés dans un dictionnaire selon leur numéro de séquence et un ACK est envoyé pour le numéro de séquence spécifique. S'il manque un numéro de séquence précédent à la réception d'un paquet, une demande de retransmission du paquet précédent est envoyée, et un compteur d'erreur est incrémenté, qui throw une TransmissionErrorException après 3 erreurs. Après la réception de tous les paquets et un recompte, ces données sont recombinées pour êtres envoyées à la couche d'Application. Cette dernière en réception extrait des données le nom du fichier, et réécrit le fichier dans un dossier de destination spécifié par les paramètres de l'application, en transmission, l'inverse est fait; le nom du fichier est lu et mis dans les 188 premiers octets du paquet (coupé si dépasse), et le reste des données sont streamées dans le paquet. Nous pouvons par la suite vérifier si nos fichiers sont les mêmes avec un hash MD5 qui démontre que même de gros fichiers tels que le guide étudiant peuvent être transmis aisément sans erreurs.

3.2 PLAN DE TEST

Test	Résultat attendu
Démarrer le programme avec en paramètre un petit ficher et le paramètre ajoutant une erreur à *false.*	Fichier transmit, pas des erreurs détectées. Un hash MD5 permet de confirmer l'intégrité du transfert.
Démarrer le programme avec en paramètre un petit fichier le paramètre ajoutant une erreur à *true.*	Fichier transmit, une seule erreur détectée, mais corrigée. Un hash MD5 permet de confirmer l'intégrité du transfert.
Démarrer le programme avec en paramètre un gros ficher et le paramètre ajoutant une erreur à *false.*	Fichier transmit, pas des erreurs détectées. Un hash MD5 permet de confirmer l'intégrité du transfert.
Démarrer le programme avec en paramètre un gros fichier le paramètre ajoutant une erreur à *true.*	Fichier transmit, une seule erreur détectée, mais corrigée. Un hash MD5 permet de confirmer l'intégrité du transfert.
Démarrer le programme avec en paramètre un fichier manquant.	Une erreur est affichée à l'utilisateur et le programme s'éteint.

Démarrer le programme client dans un des tests précédents, dans l'interface serveur, pendant que la transmission est en cours, presser la touche *q* pour arrêter l'application.	Arrêt de la l'application serveur, le fichier n'est pas transmis.

3.3 DIAGRAMME DE CLASSE

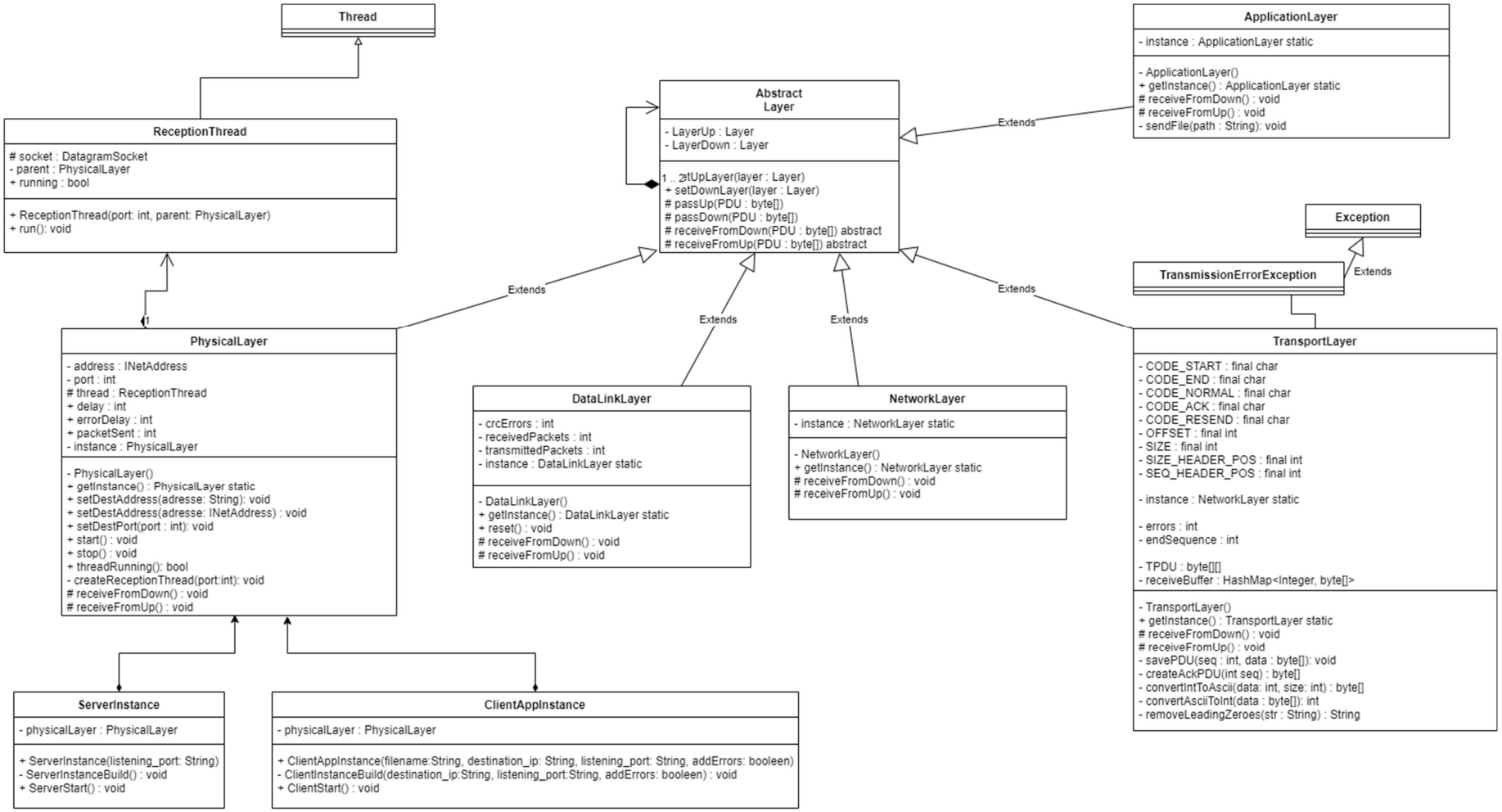


Figure 3 - Diagramme UML de classe