Notification de congestion

1 Description

On désire concevoir une application "source" qui génère du trafic *UDP* (potentiellement plusieurs flux). Cette application maintient une fenêtre d'émission par flux avec les éléments envoyés et non acquittés. Cette fenêtre permettra de gérer les retransmissions des messages perdus. La taille de la fenêtre d'émission de la source est dynamique. Cette taille est le minimum de la taille de la fenêtre de congestion et de la fenêtre de réception, annoncée par le serveur. La fenêtre de congestion évolue en fonction du temps et du nombre d'acquittements reçus (ou non). Une application "destination" reçoit et acquitte le trafic.

On cherche à coupler ces applications avec le mécanisme *ECN*. Le récent mécanisme ECN (Explicit Congestion Notification) permet aux éléments du réseau de signaler qu'ils sont trop chargés. Lors de la réception de ce signal, l'application "source" réduit sa fenêtre de congestion. Cette réduction réactive permet d'éviter la surcharge de l'élément ayant utilisé le mécanisme ECN, afin que ce-dernier ne soit pas contraint de jeter des paquets. Vous étudierez ensuite l'évolution du débit, des pertes et autres caractéristiques avec et sans ce mécanisme. L'implémentation des divers composants (expliqués en détail par la suite) *et* l'étude de l'impact du mécanisme ECN seront évalués.

Pour ce projet, nous vous fournissons un programme "perturbateur" en python, simulant un élément intermédiaire du réseau. Ce programme relaye les messages qu'il reçoit, mais peut jeter des messages et/ou mettre la valeur ECN > 1 lorsque le trafic se fait plus dense afin de signaler le besoin de réduire le débit de transmission de l'application source.

Ce programme est disponible sur un git public :

`https://git.unistra.fr/alfroy/projet_algo_reseau2021`.

En cas d'erreur, ouvrez une *issue* sur le git. N'envoyez pas de mail.

Ce programme sera appelé par la commande

```
python3 medium.py
```

Lorsque le medium pertubateur reçoit un message de la source (p.ex., sur le port 4444, cf Fig. 1), il peut le transmettre à la destination (p.ex., vers le port 6666). Le medium pertubateur peut ou non simuler sa surcharge, en taggant des paquets via le bit ECN, ou en jetant des paquets. Lorsque le medium reçoit un message de la destination (p.ex., sur le port 5555), il le transmet à la source sans modification des données ou du champs ECN (p.ex., vers le port 3333). Le programme supporte les arguments suivants :

```
Usage : python3 medium.py [option]

This program is used to simulate loss or congestion (with the ECN mode).
In normal mode, the program generates 30 percent loss when receiving
more than 100 packets within a second.
In ECN mode, it sets the ECN bit for the first packet of a 1 second
interval when receiving more than 100 packets within a second.

    -v,--verbose        Used for debug, display the header
                        for each packet received

    -s,--second         Display the number of received message each second

    -e,--ecn            Activate the ECN mode

    -l,--limit [val]    Set the packet limit rate to val.
                        When more then val packet are received within one
                        second, 30% of the packets are dropped.
```

Le schéma de communication entre l'application cliente, le serveur et le perturbateur est illustré sur la figure 1.

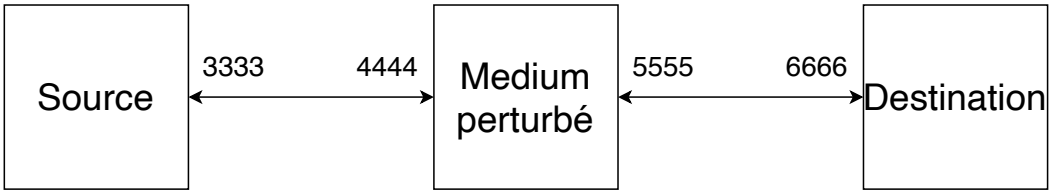


FIGURE 1 – Connectivité entre la source, le perturbateur et la destination. Les ports affichés sont les ports d'écoute de chaque programme. Par exemple, le perturbateur écoute sur le port 4444 du côté de la source. La source doit donc envoyer son trafic sur le port 4444, le perturbateur se chargera de l'acheminer à la destination, sur le port 6666.

2 Implémentation

Vous devez donc implémenter une source, émettant du trafic et conservant **une fenêtre d'émission par flux**, ainsi qu'une destination, recevant le trafic et acquittant les messages reçus. La source ajustera sa fenêtre d'émission en fonction des acquittements reçus (ou non) ainsi que d'une éventuelle notification de congestion (ECN). Les détails concernant l'évolution de la fenêtre sont indiqués plus bas dans l'énoncé.

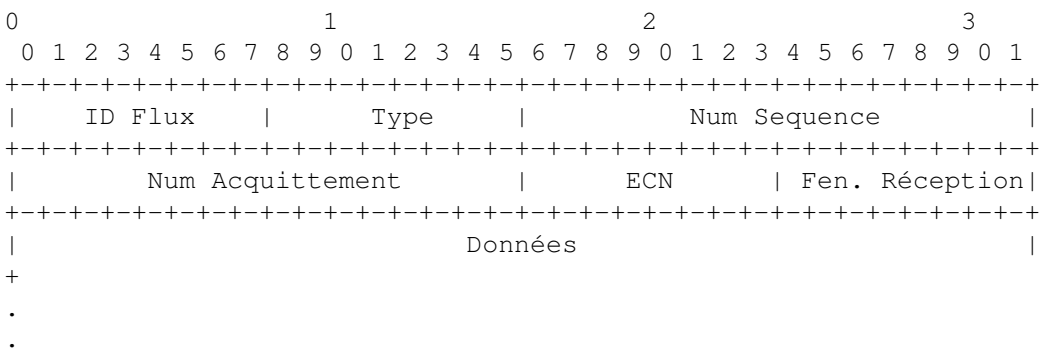
Le programme source envoie des messages dont la taille totale est fixée à 52 octets.

Le format des messages envoyés par l'application "source" est :

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    ID Flux    |      Type     |           Num Sequence          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Num Acquittement       |      ECN      | Fen. emission |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Données                             |
+
.
.
```

Le champ *ID Flux* identifie le flux du paquet (la source pouvant envoyer plusieurs flux à la fois). Le champ *type*, sur un octet, prend les valeurs 16 (ACK), 4 (RST), 2 (FIN), 1 (SYN). Un paquet peut posséder plusieurs types à la fois. Par exemple, un paquet acquittant des données peut aussi signaler l'ouverture de la communication avec le drapeau SYN. Il suffit de tester la valeur des bits correspondants aux drapeaux pour déterminer les différentes données pertinentes dans un message. Le champ *Fen. Emission* indique, sur un octet, la taille courante de la fenêtre d'émission.

La destination enverra des messages au format suivant :



Ces messages ont également une taille de 52 octets. La destination recopie la valeur du champ ECN reçu dans l'acquittement.

Les mécanismes que la source et la destination doivent supporter sont référencés ci-dessous :

- le 3-way handshake en début de connexion pour la négociation de l'ouverture de la communication,
- les mécanismes d'envois suivants :
  1. **Stop and wait** : Pour envoyer un message, la source doit avoir reçu l'acquittement du message précédent. Le mécanisme du numéro de séquence $0 - 1$ alterné sera également implémenté.
  2. **Go-Back-n** : La source peut envoyer un nombre de messages équivalent à la taille de la fenêtre de congestion avant de recevoir un acquittement. Chaque acquittement libère de la place dans la fenêtre d'émission et permet l'envoi d'un nouveau message.
- les mécanismes de contrôle de congestion suivants (dans le cas de go-back-n) :
  1. une initialisation de la taille de la fenêtre de congestion à 52 octets (1 message),
  2. un accroissement de la taille de la fenêtre de congestion de 52 octets à chaque réception d'un acquittement en séquence,
  3. une division de la fenêtre de congestion par deux lorsqu'une perte est détectée par l'expiration d'un timer,
  4. un retour de la fenêtre de congestion à 52 octets lors de la détection d'une perte par la réception de 3 acquis dupliqués.
  5. une réduction de la fenêtre de congestion de 10% lors de la réception d'un message avec le champ ECN > 0.
- la fermeture de connexion bidirectionnelle comme dans TCP à l'aide des drapeaux FIN et ACK.
- la source doit pouvoir envoyer plusieurs flux en simultané à la destination. Cette caractéristique doit permettre d'évaluer l'efficacité des différents mécanismes d'envoi dans des conditions pseudo-réalistes. Vous étudierez le partage de la bande passante entre les différents flux.

L'application source sera lancée via la commande :

```
./source <mode> <IP_distante> <port_local> <port_ecoute_src_pertubateur>
```

`mode` est soit égal à stop-wait ou go-back-n. Si l'on prend l'exemple de la figure 1 <port_local> doit valoir 3333 et <port_envoi> doit valoir 4444.

L'application "destination" sera lancée via la commande :

```
./destination <IP_distante> <port_local> <port_ecoute_dst_pertubateur>
```

L'IP distante passée en argument de l'application "destination" indique l'adresse IP utilisée en écoute par la source pour obtenir les réponses, les acquittements de la destination. Si l'on prend l'exemple de la figure 1, <port_local> doit valoir 6666 et <port_envoi> doit valoir 5555.

3 Travail demandé

Il vous est demandé :

- de programmer les composants de l'application en langage C sous Unix, avec les sockets, selon les caractéristiques évoquées ci-dessus (les programmes ne compilant pas ou ne suivant pas les spécifications demandées ne seront pas évalués) ;
- de **décrire votre implémentation et de justifier vos choix dans un rapport**. Le fond, tout comme la forme de ce rapport, seront évalués. Vous veillerez à rendre un pdf soigné (division en sections, références éventuelles, page de garde avec logo de l'université...) ;
- d'illustrer l'impact du mécanisme ECN via diverses figures **commentées**, **justifiées** et **interprétées**. Le choix des figures (évolution du débit, débit utile, taux de pertes...) est laissé libre. Vous veillerez à fournir les scripts, paramètres et données utilisés pour extraire vos figures. **Les expériences sous-jacentes aux figures doivent être reproductibles aisément**. ;
- d'**indenter** et de **commenter** votre code et de fournir un **makefile**.

4 Modalités de remise

Ce projet est à réaliser en **binôme ou individuellement**. Votre binôme est à choisir dans **votre groupe de TP**.

Vous déposerez sur Moodle, dans l'espace de devoirs prévu à cet effet pour votre groupe de TP, votre projet (documentation, sources, Makefile) sous forme d'une archive au format « *login*.tar.gz » où *login* est votre nom de login sur Turing. Vous prendrez soin de supprimer tous les fichiers binaires (exécutables, objets).

Date limite : le mercredi 17 novembre 2021 à 23 heures.

Sauf contre-ordre, une soutenance aura lieu la semaine du 29 novembre, l'heure et le jour diffèrent en fonction des groupes.