2021 – 2022

Sorbonne Université, Master 2e informatique, spécialité réseaux

Travaux sur Machines Encadrés 2 UE ITQoS

Equité, contrôle de congestion (TCP et TFRC) et gestion de buffer (RED et ECN)

À Rendre : Compte rendu électronique (pdf)

Dans ce TME, vous étudierez le contrôle de congestion implanté dans les protocoles de transport et le mécanisme de gestion active de file d'attente RED (« Random Early Detection). Le compte rendu doit contenir les réponses aux questions du TME et/ou les résultats des tests demandés. Il est à envoyer par courriel au plus tard avant le début du prochain TME. (Voir instructions détaillés d'envoi à la fin de cette page)

Préparation

A partir d'un terminal, démarrez la machine virtuelle du TP qui est une machine virtuelle Scientific Linux 7 avec la commande :

`Vbox ITQoS20_2016` Cette commande vous demandera le mot de passe de votre compte.

Le démarrage prendra plusieurs minutes. Si vous n'êtes pas automatiquement en mode plein écran, vous pouvez l'activer si vous souhaitez en tapant simultanément [Ctrl droite] [F].

Accédez à la machine virtuelle SL7 avec le compte **itqos**.

Remarque : Si vous travaillez sur votre ordinateur personnel, alors téléchargez le vmdk de la vm SL7 (RHEL7) ici. Le vmdk est à rattacher au contrôleur SATA de la vm à créer avec VMWare ou Virtual Box ou tout autre logiciel de virtualisation supportant le format vmdk pour le disque dur virtuel.

Répondez aux questions suivantes :

Contrôle de congestion au niveau transport

1/ Quels sont les protocoles de contrôle de congestion implantés pour TCP dans la machine SL7 ? (Indication : Regardez dans le répertoire `/usr/lib/modules/3.10.0-327.el7.x86_64/kernel/net/ipv4/`). Parmi eux, quels sont ceux qui ont été chargés dans le noyau en cours d'exécution ? (`sysctl -a | grep congestion`)

2/ Quel est le contrôle de congestion en cours ? (`sysctl -a | grep congestion`)

Remarque : Pour charger un contrôle de congestion dans le noyau en cours d'exécution, on utilise la commande modprobe. Pour vérifier les contrôles de congestion chargés et/ou actives, on utilise la commande **`sysctl`**.

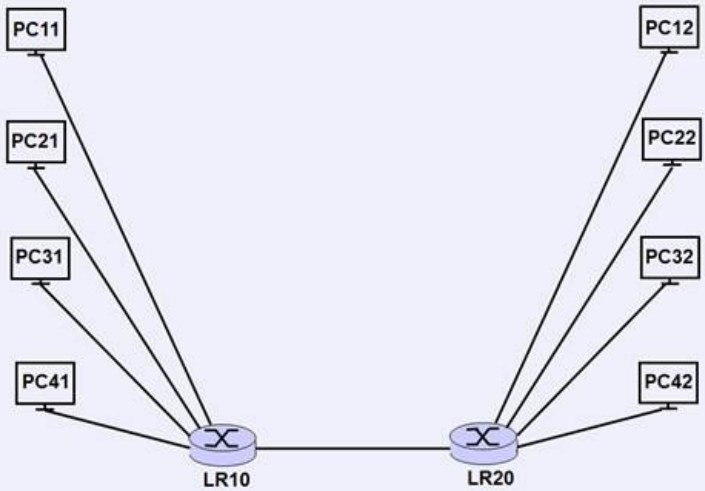
3/ Sur le site https://www.kernel.org/ (ou un autre site donnant accès aux codes sources du noyau Linux), cherchez le fichier `tcp_cubic.c`.

3.1/ Quelle est le nom de la fonction qui calcule la fenêtre de congestion selon une augmentation cubique ? Quelle est la valeur exacte de la constante c utilisée dans la formule $cwnd\_prev + c \times (t - K)^3$? Indication : cherchez la variable `bic_scale`.

3.2/ Quelle est la valeur exacte du facteur multiplicatif, β, utilisé pour réduire la taille de la fenêtre de congestion ? Indication : cherchez la variable `beta`.

Remarque : Le code est optimisé afin d'utiliser le plus possible les opérations de décalage binaire qui sont plus rapides.

4/ Un environnement de test a été créé sur votre machine virtuelle SL7 suivant ce schéma ci-dessous :



Créez un fichier de taille très grande avec la commande `dd` :

```
dd  if=/dev/zero  of=/tmp/gfichier  bs=1000  count=1250000
```

Ce fichier est accessible à partir de toutes les machines du réseau mentionné ci-dessus car leur système de fichiers est partagé avec celui de la machine SL7.

Afin de se connecter à une machine du réseau, utilisez la commande telnet. Exemple :

```
telnet PC12
```

Connectez-vous à PC12, et lancez le serveur ssh :

```
[PC12 ~]# sudo /usr/sbin/sshd
```

Ensuite, connectez-vous à PC11, et lancez le transfert du fichier gfichier vers PC12 :

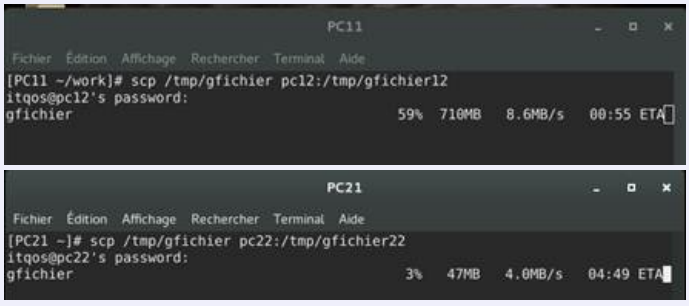
```
[PC11 ~]# scp /tmp/gfichier pc12:/tmp/gfichier12
```

4.1/ Observez et notez le débit moyen affiché par scp, durant et à la fin du transfert. Afin de convertir le débit en MB/s affiché par scp en Mbit/s, multipliez par (1024*1024*8/1000000).

Recommencez la même opération et avant que le transfert se termine, lancez un deuxième transfert entre pc21 et pc22 :

```
[PC22 ~]# sudo /usr/sbin/sshd

[PC21 ~]# scp /tmp/gfichier pc22:/tmp/gfichier22
```

4.2/ Observez l'évolution des débits de transfert sur PC11 et PC21 :
- Quelle est le débit du premier transfert juste avant de lancer le deuxième ?
- Quelle est le débit du deuxième transfert juste après son lancement ?
- Décrivez ce qui se passe ensuite.
- Enfin, quelle est le débit du premier et du deuxième transfert juste avant la fin d'un des deux transferts ? Y a-t-il convergence vers l'équité ?



L'état des deux transferts juste après le lancement du deuxième de pc21 vers pc22.

4.3/ Refaites le test avec trois transferts en parallèle et vérifiez seulement la convergence vers l'équité en mentionnant les trois valeurs de débits juste avant la fin d'un des trois transferts.

4.4/ Equité max-min :

Changez les bandes passantes des liens entre les PC et la machine LR10 comme suit :

```
[PC11 ~]# sudo tc qdisc change dev eth11 root netem delay 10ms rate 50mbit
[PC21 ~]# sudo tc qdisc change dev eth21 root netem delay 10ms rate 20mbit
[PC31 ~]# sudo tc qdisc change dev eth31 root netem delay 10ms rate 26mbit
[PC41 ~]# sudo tc qdisc change dev eth41 root netem delay 10ms rate 40mbit
```

4.4.1/ Maintenant, lancez un serveur sshd sur chaque PCi2, i = 1 à 4. Ensuite, lancez quatre transferts en suivant le même principe des expériences précédentes.

Notez les débits affichés de chaque transfert juste avant la fin d'un des transferts ou après un temps suffisamment long, afin de laisser le temps aux connexions TCP CUBIC de converger.

Est-ce que TCP CUBIC partage la bande passante (lien LR10 — LR20 de 100 Mbit/s) selon l'équité max-min ?

Afin de répondre à cette question. Il faut trouver l'allocation théorique selon l'équité max-min. Plusieurs algorithmes et/ou méthodes permettent de déterminer l'allocation max-min. L'objectif est toujours le même. Il s'agit de trouver l'allocation qui maximise le minimum des valeurs allouées. Ainsi, l'augmentation de ceux qui demandent plus, ne se fait pas au détriment de ceux qui demandent moins. Un algorithme possible est le suivant : On commence par calculer la part normale de chaque demandeur, c'est-à-dire normalement dédiée selon un ou plusieurs critères. Par exemple la part normale peut correspondre à celle obtenue avec un partage égalitaire. Ensuite, on donne à tous ceux qui demandent moins que leur part normale ce qu'ils demandent. On répète ce même procédé avec la quantité restante de la ressource à partager et les demandeurs encore sans allocations. On arrête lorsque toutes les demandes restantes soient supérieures ou égales à leur part normale. On termine alors l'algorithme par donner à chacun de ces demandeurs ces dernières parts normales. En plus des demandes, on peut associer à chaque demandeur un *poids*. Dans ce cas, on peut calculer l'allocation max-min *pondérée*. Les poids sont alors pris en compte dans le calcul des parts normales.

(**Attention :** Ne confondez pas poids et priorité. Un demandeur avec un poids plus fort ne signifie pas qu'il est prioritaire. Les deux notions sont indépendantes)

Dans notre réseau, les BPs des liens d'accès représentent les demandes. On suppose qu'il n'y a pas de poids associés aux connexions.

4.4.2/ Indice de l'équité de Jain (Jain's Fairness Index) :

Calculez l'index de l'équité permettant de mesurer "l'éloignement" de l'allocation réalisée par les connexions TCP CUBIC de l'équité max-min. Si la valeur de l'index est très proche de 1, alors on peut conclure que TCP a atteint l'équité max-min.

5/ Equation-based congestion control

Est-ce que le protocole de transport DCCP est implanté dans le noyau du système SL7 ? Indication : Tapez la commande `locate dccp`.

Sur le site https://www.kernel.org/, ou un autre site du noyau Linux, cherchez les fichiers qui implantent le contrôle de congestion TFRC (`net/dccp/ccids` et `lib`). Remarque : TFRC correspond au CCID 3.

Rappel : TFRC (RFC 5348) est un protocole de contrôle de congestion qui a pour but de contrôler le débit d'un transfert de données afin d'éviter les oscillations de débit et partager la bande passante disponible équitablement avec les connexions TCP. Il est utilisé notamment par le protocole de transport DCCP (« Datagram Congestion Control Protocol »). Il est basé sur la formule d'estimation de débit d'une connexion TCP en présence de pertes. Cette formule, appelée couramment PFTK (pour Padhye, Firoiu, Towsley et Kurose, les noms des quatre auteurs de la formule en question), définit le débit maximal à ne pas dépasser en cours de transmission.

$$T = \frac{MSS}{R\sqrt{\frac{2bp}{3}} + t_{RTO}\left(3\sqrt{\frac{3bp}{8}}\right)p\left(1+32p^2\right)}$$

Voici une brève description du protocole TFRC :

- L'émetteur commence la transmission par une phase de Slow Start assez similaire à celle de TCP.
- Le récepteur mesure la probabilité de pertes p, puis l'envoie à l'émetteur.
- L'émetteur utilise des paquets envoyés par le récepteur pour mesurer un temps d'aller-retour moyen R. Il estime aussi le temporisateur de retransmission t_{RTO}.
- L'émetteur utilise la formule PFTK pour ajuster son débit de transmission en conséquence.

L'estimation de la probabilité de pertes est un élément essentiel dans la conception du protocole. Si cette probabilité est mesurée sur une *longue* période de temps, elle peut résulter en un contrôle trop lent, pas suffisamment réactif. Si la probabilité de perte est mesurée sur une *courte* période de temps, elle peut donner lieu à un contrôle instable surréagissant à chaque changement d'état du réseau même minime. L'algorithme calculant cette probabilité est donc primordial pour réaliser un bon compromis entre stabilité et convergence rapide.

La probabilité de pertes est calculée de la façon suivante :

- $p = 1/S$
- $$S = \frac{s_1 + s_2 + s_3 + s_4 + \frac{4}{5}s_5 + \frac{3}{5}s_6 + \frac{2}{5}s_7 + \frac{1}{5}s_8}{6},$$
- Le nombre 8 représente le nombre d'intervalles de pertes. Un intervalle de perte est défini comme étant l'intervalle entre deux événements de pertes successives.
- s_i = le nombre de paquets reçus pendant le $i^{ème}$ intervalle de perte (donc entre deux événements de pertes). s_1 étant le plus récent et s_8 étant le plus ancien (voir figure ci-dessous. Par conséquent, les intervalles qui contribuent le plus au calcul de la probabilité de pertes sont les intervalles s_1 à s_4 représentant l'état récent actuel du réseau. Les poids, et donc

la contribution, des autres intervalles sont réduits en fonction de leur ancienneté : Plus l'intervalle est ancien, moins il reflète l'état actuel du réseau, plus son poids est faible, moins il contribue au calcul de la moyenne (la somme pondérée S).

- Lorsque le récepteur commence à recevoir les nouveaux paquets dans l'intervalle courant (s_0) après la dernière perte, il faut inclure ces paquets dans le calcul de S à partir du moment où le taux de perte commence à diminuer (c-à-d S commence à augmenter). Cela permet d'autoriser une augmentation du débit justifiée par le fait que la période s_0 est sans pertes. L'algorithme exact de calcul de S est donc le suivant:

$$S = \max\left(\frac{s_1 + s_2 + s_3 + s_4 + \frac{4}{5}s_5 + \frac{3}{5}s_6 + \frac{2}{5}s_7 + \frac{1}{5}s_8}{6}, \frac{s_0 + s_1 + s_2 + s_3 + \frac{4}{5}s_4 + \frac{3}{5}s_5 + \frac{2}{5}s_6 + \frac{1}{5}s_7}{6}\right)$$

5.1/ Que fait la fonction `tfrc_lh_calc_i_mean()` du fichier loss_interval.c ?

5.2/ Quels sont les poids utilisés dans l'estimation de la probabilité de perte ? Le calcul de la moyenne S, est-il identique à celui présenté ci-dessus ? Bien justifiez votre réponse.

Random Early Detection

6/ RED est implanté dans le fichier `sch_red.c` dans le répertoire `sched/` dans lequel on trouve les mécanismes liés à la planification de la transmission de paquets. Une description de cette implémentation se trouve dans le fichier `include/net/red.h`. ARED est implanté dans le même fichier `sch_red.c`. WRED est implanté dans le fichier `sch_gred.c` (`g` pour générique).

Afin de l'activer et de le configurer sous Linux, on utilise la commande **`tc`** (traffic control). La syntaxe et les paramètres d'entrée sont expliqués dans les pages du manuel :

`man tc-red`.

6.1/ Comment doit-on fixer le seuil min par rapport au seuil max ? Pourquoi ?
6.2/ Comment doit-on fixer le seuil max par rapport à la limite 'physique' du buffer ? Pourquoi ?
6.3/ Comment doit-on fixer la probabilité maximale de rejet ? Pourquoi ?

6.4/ Testez :

```
sudo tc qdisc replace dev enp0s3 root red limit 150000 avpkt 1500
```

Ensuite : **`sudo tc -d qdisc show dev enp0s3`**

Remarque : Si l'interface de sortie de la vm SL7 ne se nomme pas `enp0s3` alors utilisez le bon nom d'interface correspondant à votre vm

Donnez les valeurs des seuils min et max calculées par `tc qdisc`. Ces valeurs suivent-elles les recommandations données en 6.1/ et 6.2/ ?

6.5/ Si on note par w le poids utilisé dans le calcul de la moyenne EWMA, alors on peut exprimer w comme une puissance de deux : $w=2^{-wlog}$. C'est la valeur de wlog qui est affichée par `tc qdisc` comme paramètre **`ewma`**. wlog est appelé aussi "ewma exponent" ou "weight exponent". Si on veut augmenter la stabilité de l'algorithme, il faut diminuer w, autrement dit augmenter wlog. Cela permet d'augmenter la taille de la rafale (« burst ») de paquets qui peuvent arriver en même temps sans impacter de beaucoup le calcul de la moyenne, et donc le taux de rejet (Voir cours 3, diapo 49 – L'algorithme). Le paramètre burst de la commande permet de contrôler directement la taille de cette rafale, et donc indirectement wlog et w. Remarque : `burst` doit être supérieur à `min`.

Testez :

```
sudo tc qdisc replace dev enp0s3 root red limit 150000 avpkt 1500 burst 10
```

et

```
sudo tc qdisc replace dev enp0s3 root red limit 150000 avpkt 1500 burst 60
```

Quels sont les valeurs obtenues des deux paramètres `ewma` et les poids (w) associés ?

Remarque : Nous testerons les performances de RED ultérieurement.

Explicit Congestion Notification

Pour activer la notification explicite de congestion avec RED, il suffit de rajouter le mot **`ecn`** dans la même ligne de la commande `tc qdisc`. Néanmoins, afin d'observer les échanges liés à l'ECN de manière plus simple, sûre et contrôlable, nous n'allons pas passer par RED. Nous allons plutôt *émuler* l'envoi de message explicite de congestion vers le récepteur. À partir de LR10 :

```
[LR10 ~]# sudo tc qdisc replace dev eth100 root netem delay 100ms loss 1% ecn
```

Activez la prise en charge de la notification explicite de congestion par TCP dans le noyau du système linux avec la commande :

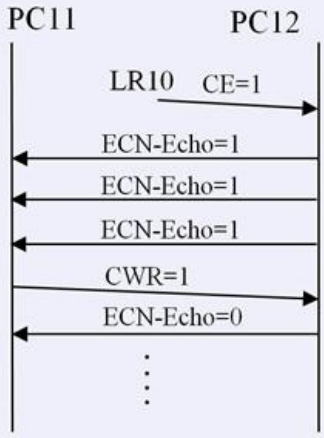
```
[PC11 ~]# sudo sysctl net.ipv4.tcp_ecn=1

[PC12 ~]# sudo sysctl net.ipv4.tcp_ecn=1
```

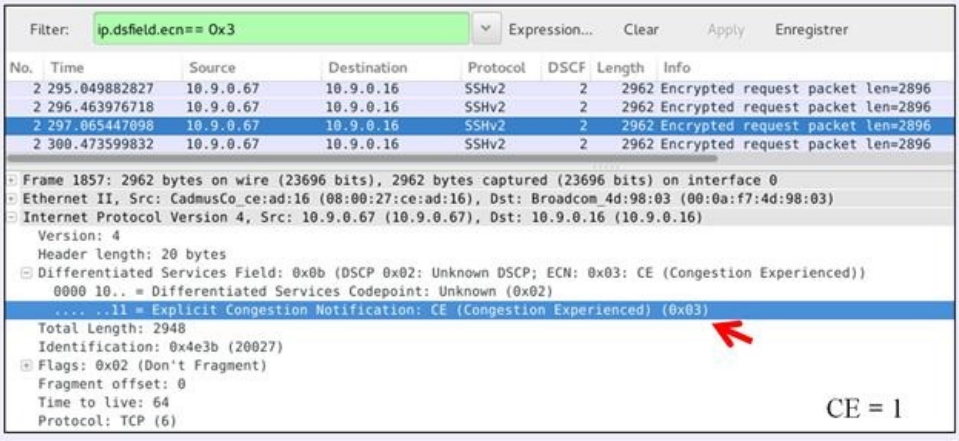
Ensuite, à partir de PC11, envoyez avec `scp` le fichier `gfichier` vers PC12.

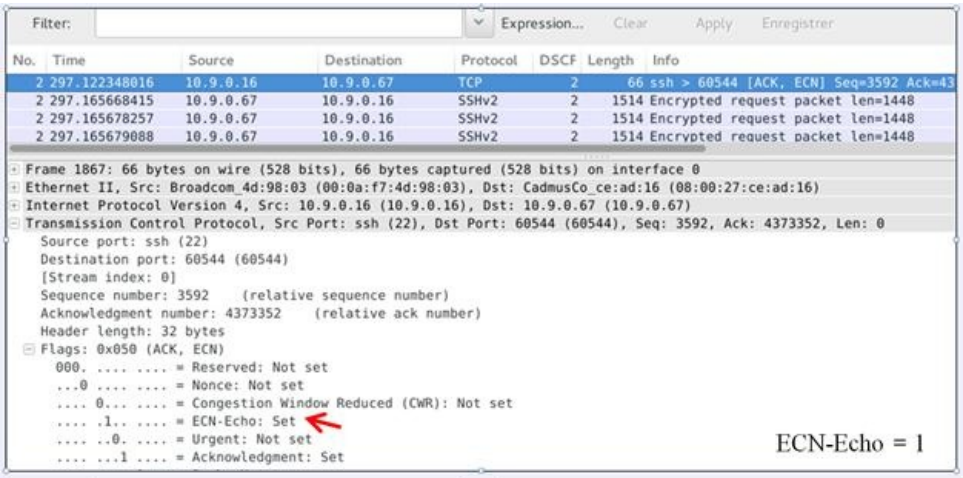
Avec wireshark, vérifiez que les paquets TCP envoyés par l'émetteur (PC11) sont ECN-capable Transport (champs 0x2→ECT=1, ou ECN=0x1). Filter : **`ip.dsfield.ecn == 0x2`**

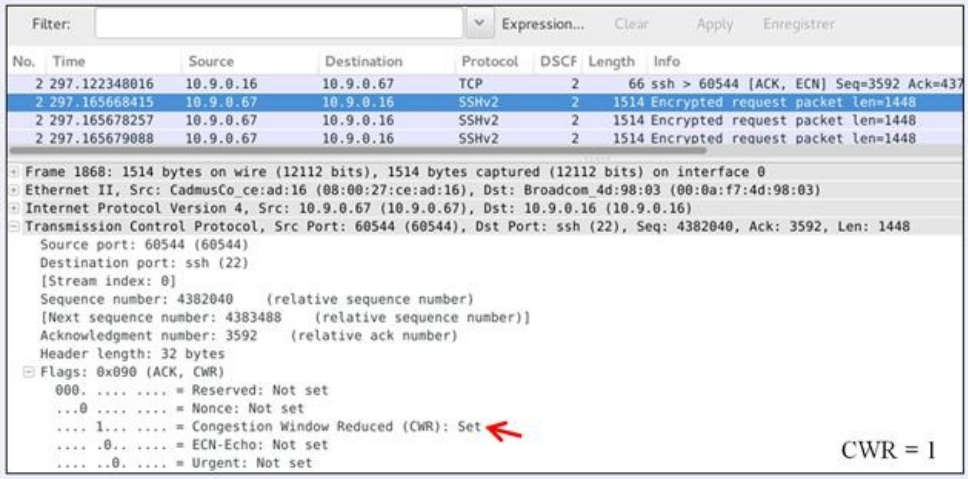
Ensuite, vérifiez toujours avec wireshark que l'interface marque le champ ECN de certains paquets (ECN=0x3→CE=1=Congestion Experienced, et ECT=1).

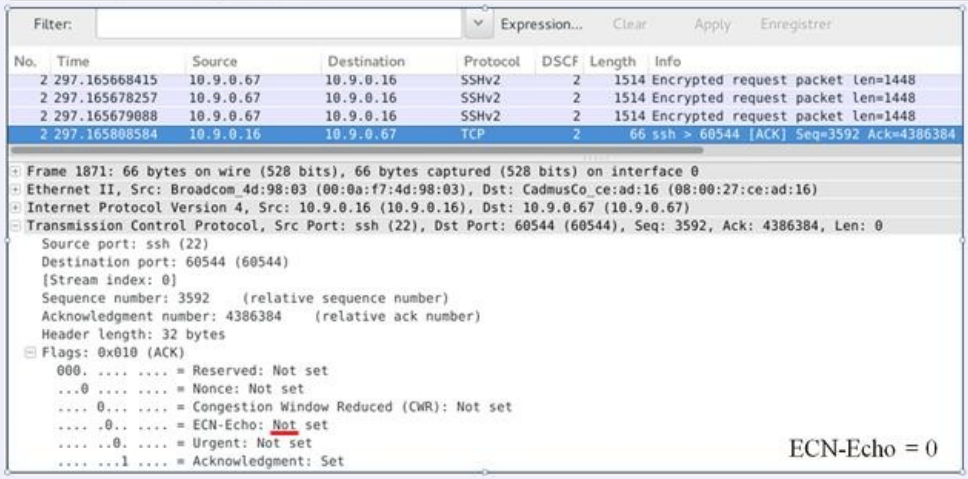
7/ A partir, du paquet marqué ECN=0x3, trouvez dans les paquets suivants, les paquets TCP avec le drapeau (« flag ») ECN-echo, envoyés par PC12 à PC11. Ensuite, le paquet TCP avec le drapeau CWR, envoyé par PC11 à PC12, indiquant que l'émetteur PC11 a réagi à la congestion. Vérifiez que lorsque ce dernier est reçu par le récepteur PC12 alors il ne marque plus le bit ECN-echo (autrement dit il ne lève plus le drapeau ECN-echo). Faites des captures d'ecran qui montrent ces différents paquets dans wireshark. Voir images et schéma ci-dessous.











Remarque : Nous testerons les performances de RED-ECN ultérieurement.

Procédure d'envoi de votre comptre rendu de TME :

Nommez le fichier de votre compte rendu TME2*_NOM1_NOM2*.pdf, avec *NOM1_NOM2* les noms des deux binômes.

Envoyez le fichier pdf à Naceur.Malouch@lip6.fr

Le sujet de l'email doit être : [ITQoS] Compte rendu TME 2 de NOM1 et NOM2

Fin !