



SORBONNE UNIVERSITÉ  
Campus Sciences et Ingénierie, Jussieu

---

# COMPTE RENDU TME 2 ITQOS

---

KITOKO DAVID ET DIZDAREVIC ADRIAN

ITQOS

# Contrôle de congestion au niveau transport

## 1. Protocoles de contrôle de congestion

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/`

les protocoles de contrôle de congestion TCP disponibles sont :

`tcp_bic.ko` BIC (Binary Increase Congestion control)

`tcp_dctcp.ko` DCTCP (Data Center TCP)

`tcp_highspeed.ko` HighSpeed TCP

`tcp_htcp.ko` H-TCP (Hamilton TCP)

`tcp_hybla.ko` Hybla (pour les liaisons satellites)

`tcp_illinois.ko` Illinois

`tcp_lp.ko` LP (Low Priority)

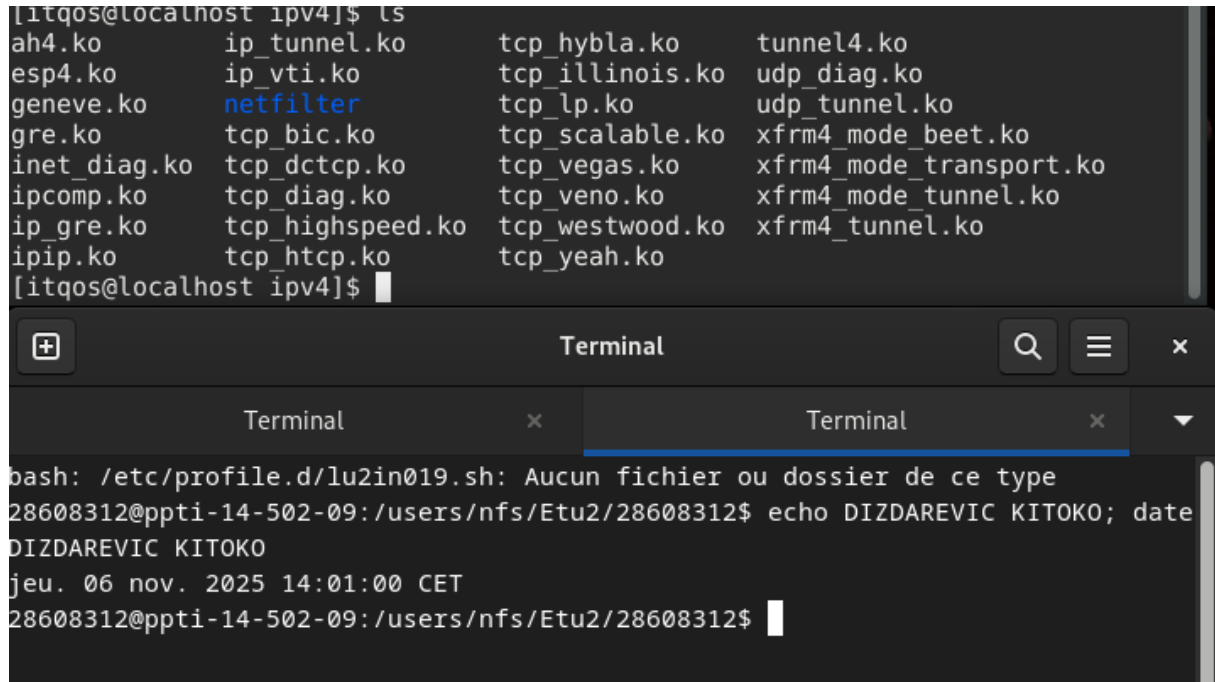
`tcp_scalable.ko` Scalable TCP

`tcp_vegas.ko` Vegas

`tcp_veno.ko` Veno

`tcp_westwood.ko` Westwood

`tcp_yeah.ko` YeAH (Yet Another Highspeed)



```
[itqos@localhost ipv4]$ ls
ah4.ko          ip_tunnel.ko      tcp_hybla.ko      tunnel4.ko
esp4.ko          ip_vti.ko         tcp_illinois.ko   udp_diag.ko
geneve.ko        netfilter          tcp_lp.ko         udp_tunnel.ko
gre.ko           tcp_bic.ko         tcp_scalable.ko   xfrm4_mode_beet.ko
inet_diag.ko     tcp_dctcp.ko       tcp_vegas.ko      xfrm4_mode_transport.ko
ipcomp.ko        tcp_diag.ko        tcp_veno.ko       xfrm4_mode_tunnel.ko
ip_gre.ko        tcp_highspeed.ko   tcp_westwood.ko   xfrm4_tunnel.ko
ipip.ko          tcp_htcp.ko        tcp_yeah.ko
[itqos@localhost ipv4]$
```

Terminal

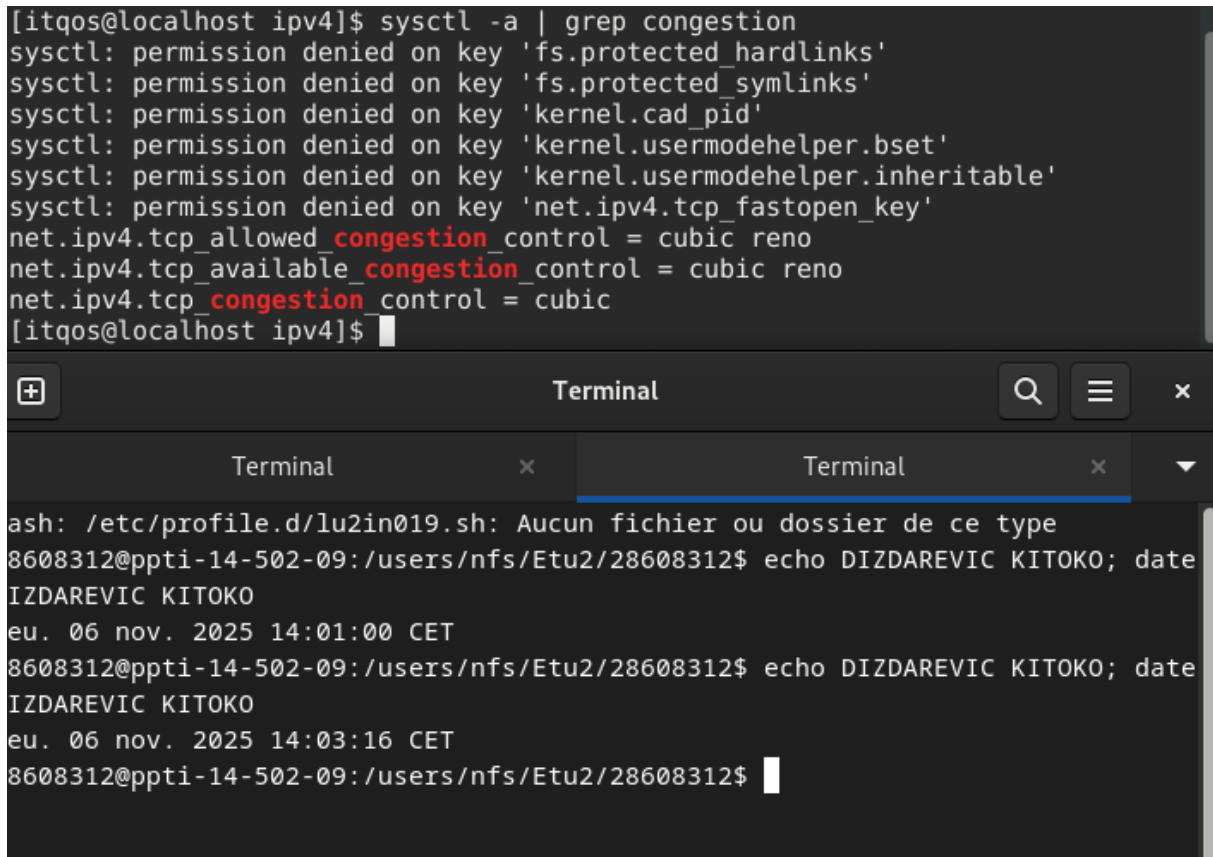
bash: /etc/profile.d/lu2in019.sh: Aucun fichier ou dossier de ce type  
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312\$ echo DIZDAREVIC KITOKO; date  
DIZDAREVIC KITOKO  
jeu. 06 nov. 2025 14:01:00 CET  
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312\$

FIGURE 1 – protocoles de contrôle de congestion TCP disponible

Parmi eux, quels sont ceux qui ont été chargés dans le noyau en cours d'exécution ?

**Commande :**

```
sysctl -a | grep congestion
```



```
[itqos@localhost ipv4]$ sysctl -a | grep congestion
sysctl: permission denied on key 'fs.protected_hardlinks'
sysctl: permission denied on key 'fs.protected_symlinks'
sysctl: permission denied on key 'kernel.cad_pid'
sysctl: permission denied on key 'kernel.usermodehelper.bset'
sysctl: permission denied on key 'kernel.usermodehelper.inheritable'
sysctl: permission denied on key 'net.ipv4.tcp_fastopen_key'
net.ipv4.tcp_allowed_congestion_control = cubic reno
net.ipv4.tcp_available_congestion_control = cubic reno
net.ipv4.tcp_congestion_control = cubic
[itqos@localhost ipv4]$
```

Terminal

```
ash: /etc/profile.d/lu2in019.sh: Aucun fichier ou dossier de ce type
8608312@ppti-14-502-09:/users/nfs/Etu2/28608312$ echo DIZDAREVIC KITOKO; date
DIZDAREVIC KITOKO
eu. 06 nov. 2025 14:01:00 CET
8608312@ppti-14-502-09:/users/nfs/Etu2/28608312$ echo DIZDAREVIC KITOKO; date
DIZDAREVIC KITOKO
eu. 06 nov. 2025 14:03:16 CET
8608312@ppti-14-502-09:/users/nfs/Etu2/28608312$
```

FIGURE 2 – protocoles de contrôle de congestion TCP disponible et chargés dans le noyau en cours d'exécution

---

## 2. Contrôle de congestion actif et chargement de nouveaux modules

- 2.1/ Quel est le contrôle de congestion en cours ?  
le contrôle de congestion en cours est cubic.
-

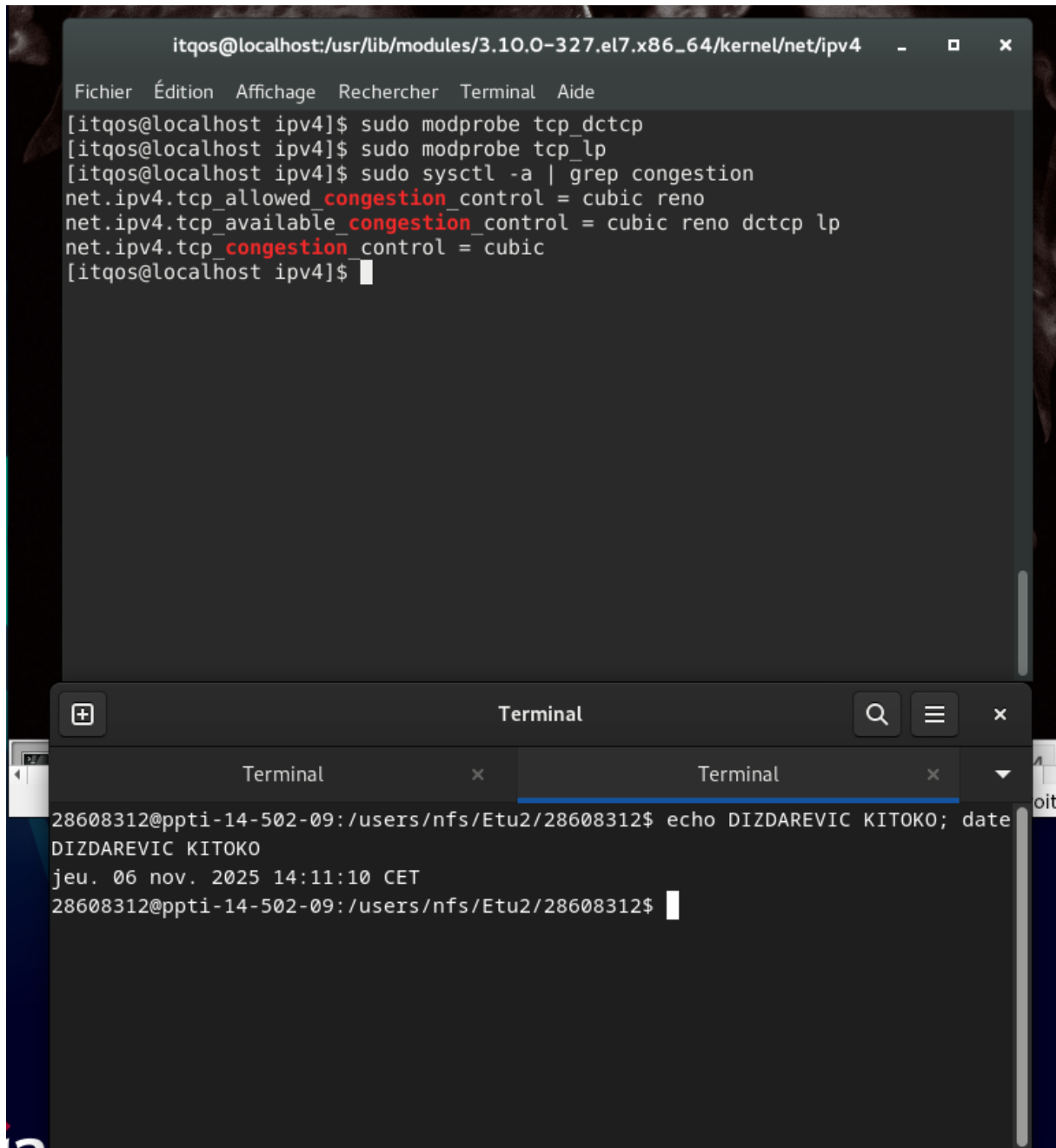
2.2/ Pour charger un contrôle de congestion dans le noyau en cours d'exécution, on utilise la commande `modprobe`.

Chargez le contrôle de congestion dédié conçu pour les centres de données **DCTCP**.

Chargez aussi celui de **TCP-LP** si ce n'est pas encore le cas.

Pour vérifier les contrôles de congestion chargés et/ou actifs, on utilise la commande `sysctl`.

Gardez toujours **cubic** comme contrôle courant (actif).



The image shows two terminal windows. The top window, titled 'itqos@localhost:/usr/lib/modules/3.10.0-327.el7.x86\_64/kernel/net/ipv4', displays the following commands and output:

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[itqos@localhost ipv4]$ sudo modprobe tcp_dctcp
[itqos@localhost ipv4]$ sudo modprobe tcp_lp
[itqos@localhost ipv4]$ sudo sysctl -a | grep congestion
net.ipv4.tcp_allowed_congestion_control = cubic reno
net.ipv4.tcp_available_congestion_control = cubic reno dctcp lp
net.ipv4.tcp_congestion_control = cubic
[itqos@localhost ipv4]$
```

The bottom window, titled 'Terminal', shows the output of the command `echo DIZDAREVIC KITOKO; date`:

```
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312$ echo DIZDAREVIC KITOKO; date
DIZDAREVIC KITOKO
jeu. 06 nov. 2025 14:11:10 CET
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312$
```

FIGURE 3 – contrôles de congestion chargés et/ou actives

---

### 3. Recherche dans le code source du noyau Linux

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 :

$$cwnd_{\text{prev}} + c \times (t - K)^3$$

**Indication :** cherchez la variable `bic_scale`.

Le nom de la fonction est `bictcp_update()`.

La valeur exacte de la constante  $c$  utilisée dans la formule est définie par `bic_scale` : `static int bic_scale __read_mostly = 41`; La constante est échelonnée par un facteur de 1024, ainsi :

$$c = \frac{41}{1024} \approx 0,04.$$

---

3.2/ Quelle est la valeur exacte du facteur multiplicatif  $\beta$ , utilisé pour réduire la taille de la fenêtre de congestion ?

**Indication :** cherchez la variable `beta`. Dans le module `tcp_cubic.c`, la variable `beta` est définie à la ligne 47 : `static int beta __read_mostly = 717`; La constante `BETA_SCALE` vaut 1024, ainsi le facteur de réduction est donné par  $\beta = \frac{717}{1024} \approx 0,7$ .

---

### 4. Expérimentations et observation des débits TCP

Un environnement de test a été créé sur votre machine virtuelle SL7 suivant le schéma ci-dessous.

## Création du fichier de test

Créez un fichier de grande taille avec la commande :

```
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.

## Connexion et transfert

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

```
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 Observation du débit moyen

Observez et notez le débit moyen affiché par **scp**, durant et à la fin du transfert. Afin de convertir le débit MB/s affiché par **scp** en Mbit/s, multipliez par :

$$1024 \times 1024 \times 8 / 1000000$$

Pour un débit de 11.8 MB/s affiché par **scp**, on obtient :

$$11.8 \times \frac{1024 \times 1024 \times 8}{1000000} \approx 98.7 \text{ Mbit/s.}$$



```
PC11
Fichier Édition Affichage Rechercher Terminal Onglets Aide
PC11 x PC12 x
[ltqos@localhost ipv4]$ telnet PC11
Trying PC11...
[PC11 /usr/lib/modules/3.10.0-327.el7.x86_64/kernel/net/ipv4]$ scp /tmp/gfighier pc12:/tmp/gfighier12
ltqos@pc12's password:
/tmp/gfighier: No such file or directory
[PC11 ~]$ scp /tmp/gfighier pc12:/tmp/gfighier12
ltqos@pc12's password:
/tmp/gfighier: No such file or directory
[PC11 ~]$ scp /tmp/gfighier pc12:/tmp/gfighier12
ltqos@pc12's password:
Permission denied, please try again.
ltqos@pc12's password:
/tmp/gfighier: No such file or directory
[PC11 ~]$ scp /tmp/gfighier pc12:/tmp/gfighier12
ltqos@pc12's password:
gfighier 100% 1192MB 11.8MB/s 01:41
PC11 ~$
```

FIGURE 5 –

— Quelle est le débit du deuxième transfert juste après son lancement ?

```
PC11
Fichier Édition Affichage Rechercher Terminal Onglets Aide
PC11 x PC12 x PC22 x PC21 x
[ltqos@localhost ipv4]$ telnet PC21
Trying PC21...
[PC21 /usr/lib/modules/3.10.0-327.el7.x86_64/kernel/net/ipv4]$ cd -
[PC21 ~]$ scp /tmp/gfighier pc22:/tmp/gfighier22
The authenticity of host 'pc22 (10.14.22.5)' can't be established.
ECDSA key fingerprint is 8a:9f:01:03:28:c3:2e:79:45:45:b0:00:70:30:dd:1c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pc22,10.14.22.5' (ECDSA) to the list of known hosts.
ltqos@pc22's password:
gfighier 30% 364MB 6.3MB/s 02:11 ETA
```

FIGURE 6 –

Au tout début nous avons un débit d'environ 1 MB/s mais augmente très vite vers la valeur affichée sur le screen.



- Décrivez ce qui se passe ensuite.

Au début lorsqu'on lance le premier transfert on a le même débit qu'à la question 4.1. et Lorsque nous lançons le deuxième on a débit très faible puis petit à petit que ce soit le premier et le deuxième transfert converge vers un débit similaire.

- 
- 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 ?

Pour le premier nous sommes à 5.8 MB/s. Pour le deuxième nous sommes à 6.3 MB/s. Donc nous avons bien une convergence vers l'équité.

---

### 4.3 Trois transferts en parallèle

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.

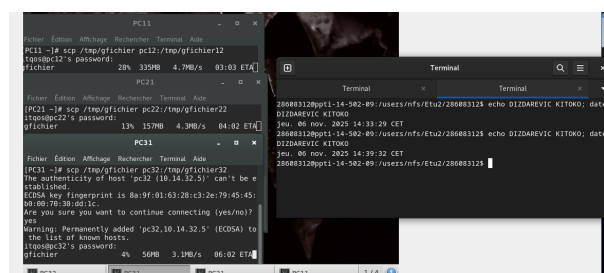


FIGURE 7 —

Nous pouvons observer ici comment le débit des trois transferts au fur et à mesure converge vers des valeurs quasi-similaires.

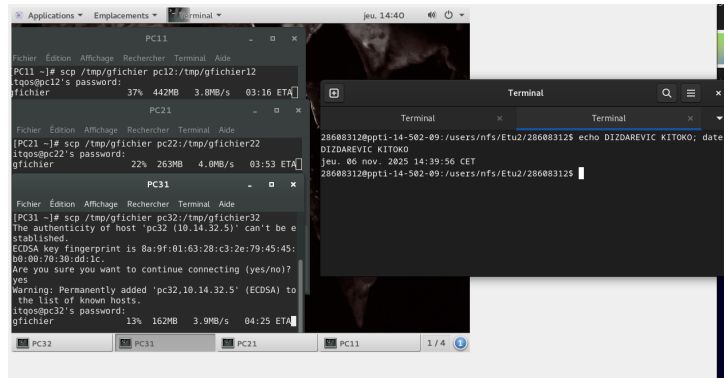


FIGURE 8 –

## 4.4 Équité 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 Expérimentation à quatre transferts

Lancez un serveur sshd sur chaque PC*i*,  $i = 1$  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).

- Débit du transfert 1 : 2.7 MB/s  $\Rightarrow$  22.65 Mbit/s \_\_\_\_\_
- Débit du transfert 2 : 2.3 MB/s  $\Rightarrow$  19.29 Mbit/s \_\_\_\_\_
- Débit du transfert 3 : 3.0 MB/s  $\Rightarrow$  25.17 Mbit/s \_\_\_\_\_
- Débit du transfert 4 : 3.8 MB/s  $\Rightarrow$  31.88 Mbit/s \_\_\_\_\_

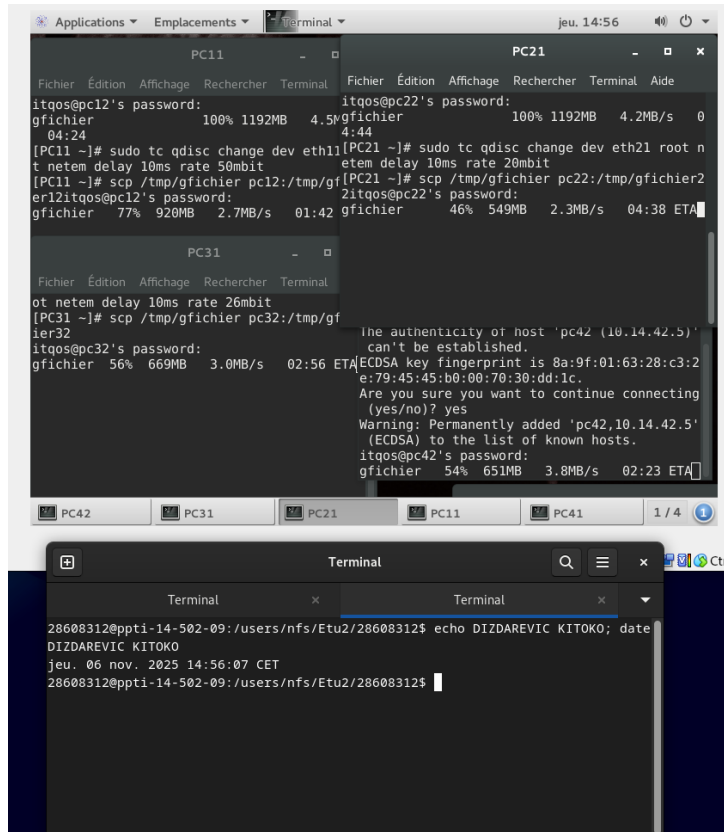


FIGURE 9 –

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

**Déroulement de l'algorithme selon l'équité max-min**

**Notations :**

- $C$  : capacité totale (100 Mbit/s)
- $pn$  : part normale
- $a$  : allocation
- $D$  : demandeur

**Itération 1 :**

D	50	20	26	40
pn	$\frac{100}{4}$	$\frac{100}{4}$	$\frac{100}{4}$	$\frac{100}{4}$
a	—	20	—	—

$$C = 100 - 20 = 80$$

**Itération 2 :**

D	50	26	40
pn	$\frac{80}{3}$	$\frac{80}{3}$	$\frac{80}{3}$
a	—	26	—

$$C = 80 - 26 = 54$$

**Itération 3 :**

<b>D</b>	50	40
<b>pn</b>	$\frac{54}{2}$	$\frac{54}{2}$
<b>a</b>	27	27

*L'algorithmes'arrête.*

#### 4.4.2 Indice d'équité de Jain

Calculez l'indice de l'équité de Jain (*Jain's Fairness Index*) permettant de mesurer l'« éloignement » de l'allocation réalisée par TCP CUBIC par rapport à l'équité max-min. Si la valeur de l'indice est très proche de 1, alors on peut conclure que TCP a atteint l'équité max-min.

La formule de l'indice de Jain est :

$$J(X) = \frac{(\sum_{i=1}^n X_i)^2}{n \cdot \sum_{i=1}^n X_i^2}$$

où  $X_i$  est le débit du flux  $i$  et  $n$  le nombre de flux.

#### Données

$$X_1 = 22.65, \quad X_2 = 19.29, \quad X_3 = 25.17, \quad X_4 = 31.88$$

$$n = 4$$

#### Étape 1 : somme des débits

$$\sum_{i=1}^4 X_i = 22.65 + 19.29 + 25.17 + 31.88 = 98.99$$

#### Étape 2 : somme des carrés des débits

$$\sum_{i=1}^4 X_i^2 = 22.65^2 + 19.29^2 + 25.17^2 + 31.88^2$$

Calcul détaillé :

$$22.65^2 \approx 513.22, \quad 19.29^2 \approx 371.06, \quad 25.17^2 \approx 633.53, \quad 31.88^2 \approx 1016.33$$

$$\sum X_i^2 = 513.22 + 371.06 + 633.53 + 1016.33 \approx 2534.14$$

### Étape 3 : calcul de l'indice de Jain

$$J(X) = \frac{(\sum X_i)^2}{n \cdot \sum X_i^2} = \frac{(98.99)^2}{4 \cdot 2534.14}$$

$$(98.99)^2 \approx 9798.02, \quad 4 \cdot 2534.14 = 10136.56$$

$$J(X) \approx \frac{9798.02}{10136.56} \approx 0.967$$

La valeur de l'index est très proche de 1, alors on peut conclure que TCP a atteint l'équité max-min.

---

## 4.5 Test avec TCP-LP

Dans la VM SL7 (et à l'extérieur des PC du réseau de test), changez le contrôle de congestion actif à TCP-LP. Faites un seul transfert de PC11 à PC12.

**Question :** Que constatez-vous? Est-ce normal? (inutile de terminer le transfert jusqu'au bout)

```
[PC11 ~]# scp /tmp/gfichier pc12:/tmp/gfichier12
itqos@pc12's password:
gfichier                                97% 1166MB   5.5MB/s   00:04 ETA^
CKilled by signal 2.
[PC11 ~]#
```

```
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312$ echo DIZDAREVIC KITOKO; date
DIZDAREVIC KITOKO
eu. 06 nov. 2025 15:12:48 CET
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312$
```

FIGURE 10 –

On observe un débit d'environ 5.5 MB/s (soit 45 Mbit/s), ce qui est un débit très proche de la limite de 50 Mbit/s. Cela est normal, puisque le comportement *Low Priority* n'intervient que lorsqu'il existe une situation de congestion. Dans ce cas, le transfert PC11 → PC12 étant le seul en cours, il peut se permettre d'atteindre le débit maximal.

---

Remettez TCP CUBIC après ce dernier test.

## 6. Random Early Detection (RED)

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 d'activer et de configurer RED 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 Seuils minimum et maximum

Comment doit-on fixer le seuil **min** par rapport au seuil **max** ? Pourquoi ?

Le seuil **min** fixé doit au moins être de moitié par rapport au seuil max pour prévenir les retransmissions synchrones.

---

### 6.2 Seuil maximal du buffer

Comment doit-on fixer le seuil max par rapport à la limite 'physique' du buffer ? Pourquoi ?

Le **seuil max** du buffer ne doit pas être fixé exactement à la limite physique du buffer mais légèrement en dessous.

- La probabilité de rejet n'est jamais nulle, même avant que le buffer atteigne sa capacité maximale.
  - Fixer le seuil exactement à la limite physique peut provoquer des rejets fréquents de paquets à cause de petites variations dans l'arrivée des données.
- 

### 6.3 Probabilité maximale de rejet

Comment doit-on fixer la probabilité maximale de rejet ? Pourquoi ?

La **probabilité maximale de rejet** ne doit pas être fixée à 1.

- Une probabilité de rejet de 1 signifierait que **tous les paquets sont rejetés**, ce qui n'est pas acceptable.
  - On choisit une valeur inférieure à 1 (typiquement faible, comme 1% ou 5%) pour garantir un niveau de qualité de service acceptable.
- 

### 6.4 Test de configuration de RED

Testez la commande suivante :

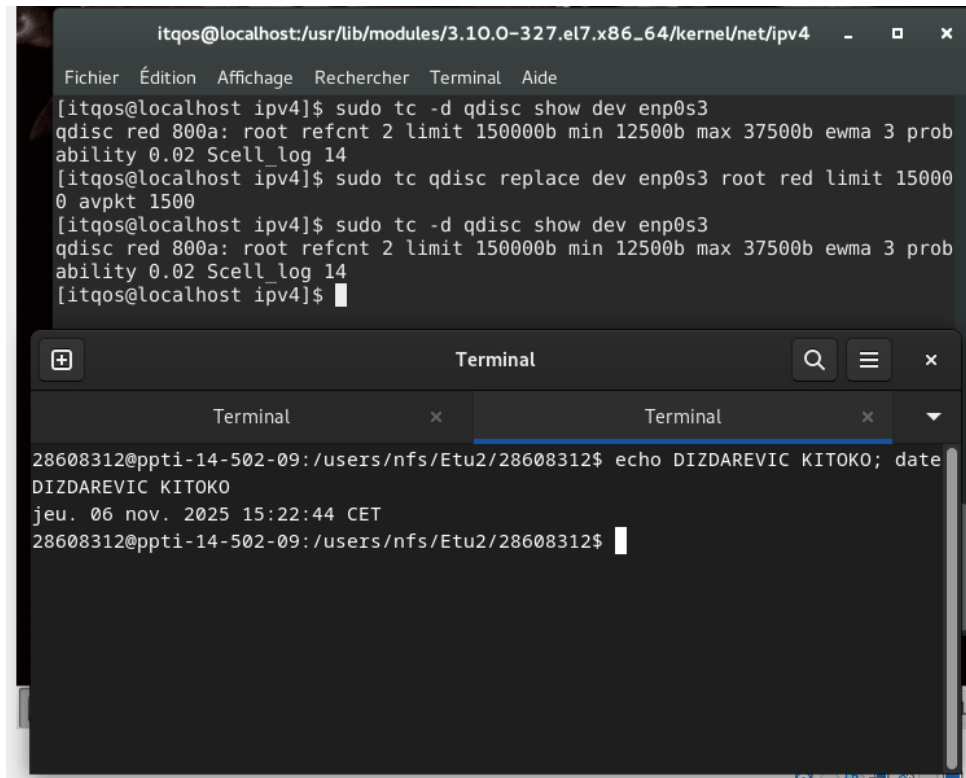
```
sudo tc qdisc replace dev enp0s3 root red limit 150000 avpkt 1500
```

Ensuite :

```
sudo tc -d qdisc show dev enp0s3
```

*Remarque* : Si l'interface de sortie de la VM SL7 ne se nomme pas **enp0s3**, utilisez le 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 ?



```
itqos@localhost:usr/lib/modules/3.10.0-327.el7.x86_64/kernel/net/ipv4
Fichier Édition Affichage Rechercher Terminal Aide
[itqos@localhost ipv4]$ sudo tc -d qdisc show dev enp0s3
qdisc red 800a: root refcnt 2 limit 150000b min 12500b max 37500b ewma 3 probability 0.02 Scell_log 14
[itqos@localhost ipv4]$ sudo tc qdisc replace dev enp0s3 root red limit 150000 avpkt 1500
[itqos@localhost ipv4]$ sudo tc -d qdisc show dev enp0s3
qdisc red 800a: root refcnt 2 limit 150000b min 12500b max 37500b ewma 3 probability 0.02 Scell_log 14
[itqos@localhost ipv4]$

Terminal
Terminal
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312$ echo DIZDAREVIC KITOKO; date
DIZDAREVIC KITOKO
jeu. 06 nov. 2025 15:22:44 CET
28608312@ppti-14-502-09:/users/nfs/Etu2/28608312$
```

FIGURE 11 –

Pour le 6.1 : Le seuil max (37500b) est le triple du seuil min (12500b), donc la recommandation est respectée.

Pour le 6.2 : Le seuil max (37500b) est plus petit que la limite physique du buffer (150000b), ce qui respecte la recommandation.

## 6.5 Paramètre `ewma` et taille de rafale (`burst`)

Si l'on note par  $w$  le poids utilisé dans le calcul de la moyenne EWMA, alors on peut exprimer  $w$  comme une puissance de deux :

$$w = 2^{-wlog}$$

La valeur de `wlog` est affichée par `tc qdisc` comme paramètre `ewma`. `wlog` est aussi appelé *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 sans impacter significativement le calcul de la moyenne, et donc le taux de rejet.

Le paramètre **burst** de la commande permet de contrôler directement la taille de cette rafale, et donc indirectement  $w_{\log}$  et  $w$ . *Remarque* : **burst** doit être supérieur à **min**.

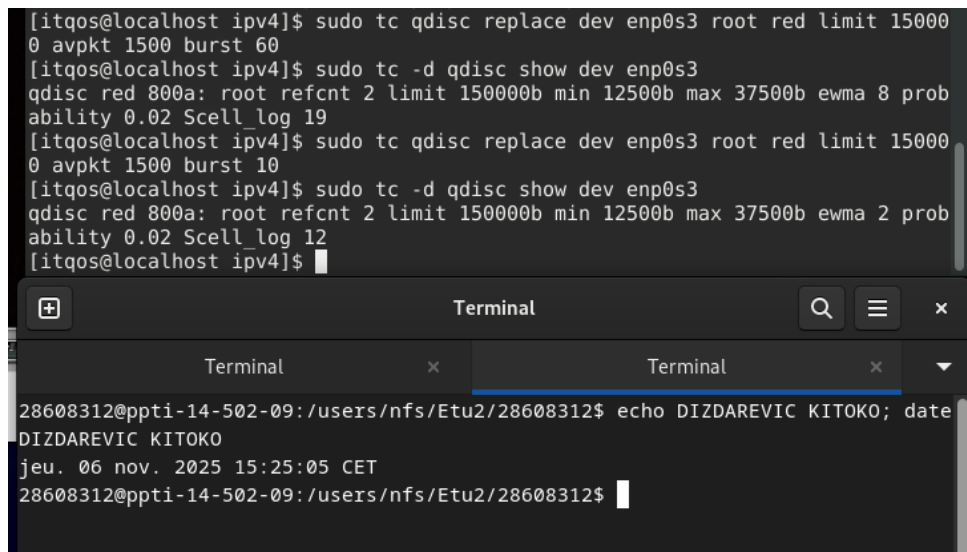
Testez :

```
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 ?



```
[itqos@localhost ipv4]$ sudo tc qdisc replace dev enp0s3 root red limit 15000
0 avpkt 1500 burst 60
[itqos@localhost ipv4]$ sudo tc -d qdisc show dev enp0s3
qdisc red 800a: root refcnt 2 limit 150000b min 12500b max 37500b ewma 8 prob
ability 0.02 Scell_log 19
[itqos@localhost ipv4]$ sudo tc qdisc replace dev enp0s3 root red limit 15000
0 avpkt 1500 burst 10
[itqos@localhost ipv4]$ sudo tc -d qdisc show dev enp0s3
qdisc red 800a: root refcnt 2 limit 150000b min 12500b max 37500b ewma 2 prob
ability 0.02 Scell_log 12
[itqos@localhost ipv4]$
```

The screenshot shows a terminal window with two tabs. The top tab shows the execution of the `tc qdisc` command to replace the root qdisc on `enp0s3` with a red qdisc, first with a burst of 60 and then with a burst of 10. The `tc -d qdisc show` command is used to display the configuration, showing the `ewma` parameter (8 for burst 60 and 2 for burst 10) and the `Scell_log` parameter (19 for burst 60 and 12 for burst 10). The bottom tab shows a simple `echo` and `date` command being executed.

FIGURE 12 –

Pour  $\text{burst} = 10$ ,  $w_{\log} = 2 \implies w = 2^{-2} = 0,25$ ,

Pour  $\text{burst} = 60$ ,  $w_{\log} = 8 \implies w = 2^{-8} = 0,00390625$ .

## 7. Notification Explicite de Congestion (ECN)

### 7. Observation des drapeaux ECN dans Wireshark

#### Captures d'écran Wireshark

Insérez ci-dessous les captures d'écran montrant :

- Le paquet marqué ECN=0x3 (CE=1).
- Le ou les paquets avec ECN-Echo (ECE=1).
- Le paquet avec CWR=1.
- Le retour à un état sans drapeau ECE.

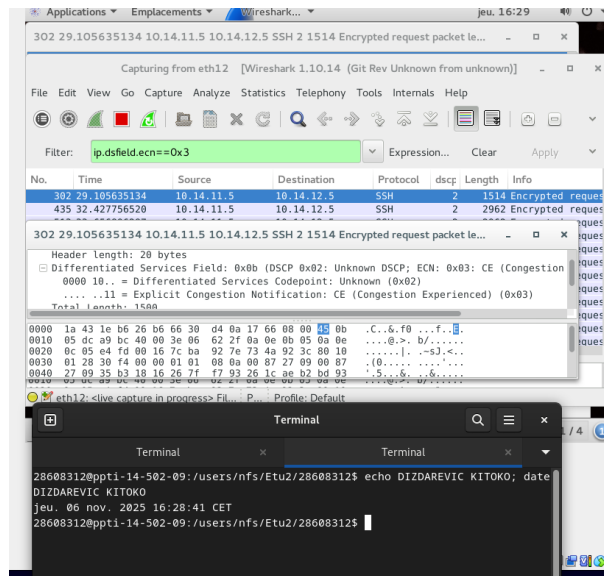


FIGURE 13 –

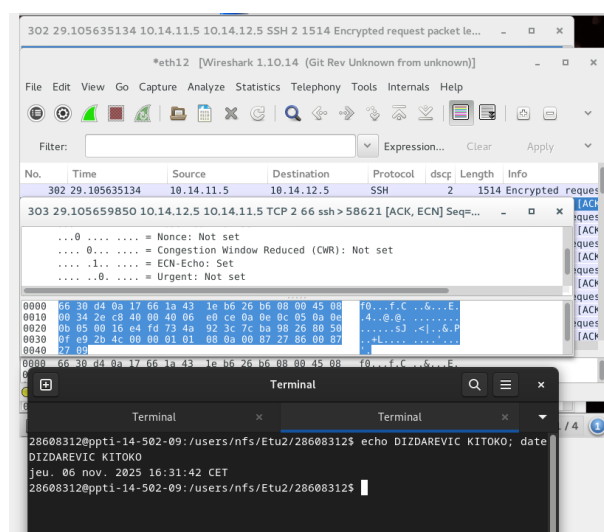


FIGURE 14 –

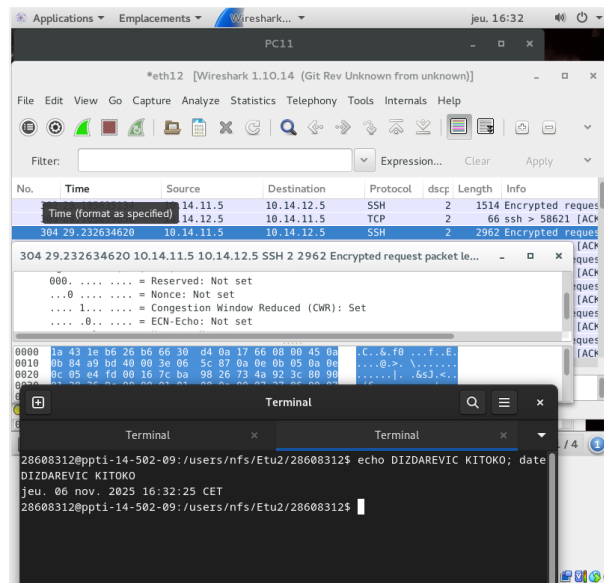


FIGURE 15 –

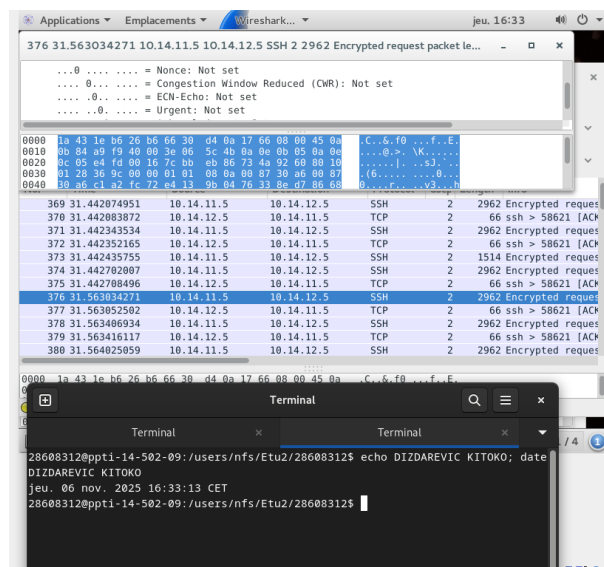


FIGURE 16 –