CASSESE Charly

BEUF Antoine

P25

PARTIE PRÉPARATION : TP1 NE 424

P1. En quoi est-il avantageux d'avoir RWS=SWS (plutôt que RWS=1 par exemple) ? Quel peut être l'effet sur les performances d'augmenter encore RWS pour avoir RWS>SWS ? Argumentez.

- Cas 1 : si RWS < SWS :

Si il y a une perte lors de l'émission d'une trame, alors la fenêtre de réception ne peut pas stocker toutes les trames envoyées, l'émetteur envoie des trames pour rien. De plus, l'émetteur stocke des données dans sa fenêtre d'émission pour rien.

- Cas 2 : si RWS = SWS :

Si RWS=SWS, alors la fenêtre de réception peut stocker toutes les trames envoyées. L'émetteur et le récepteur ne stockeront pas de trames inutiles et n'envoient pas de trames inutiles. C'est donc le cas idéal car il est optimal.

- Cas 3 : si RWS > SWS

Si RWS > SWS, alors la taille de la fenêtre de réception sera capable de stocker des trames qui ne seront jamais envoyées. Donc la taille de la fenêtre de réception est inutilement grande. Par conséquent, cela ne va pas aller plus vite que si RWS=SWS.

- Conclusion : le cas 2 est le plus pertinent.

P2. Le code de swp.c est légèrement différent de celui montré en cours. Expliquez, décrivez dans quelle situation le comportement diffère.

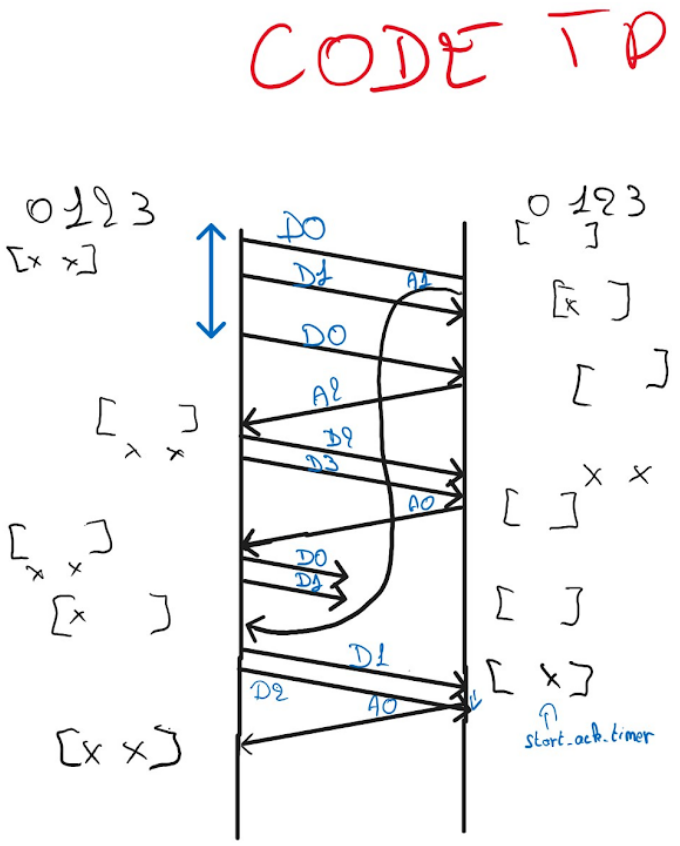
Nous avons remarqué que les timers d'acquittements ne se lancent pas au même moment. Cela change le fonctionnement du protocole:

Pour comprendre la différence entre le code du TP et le code du cours, nous avons élaboré un schéma qui détaille les différences.

CASSESE Charly

BEUF Antoine

P25



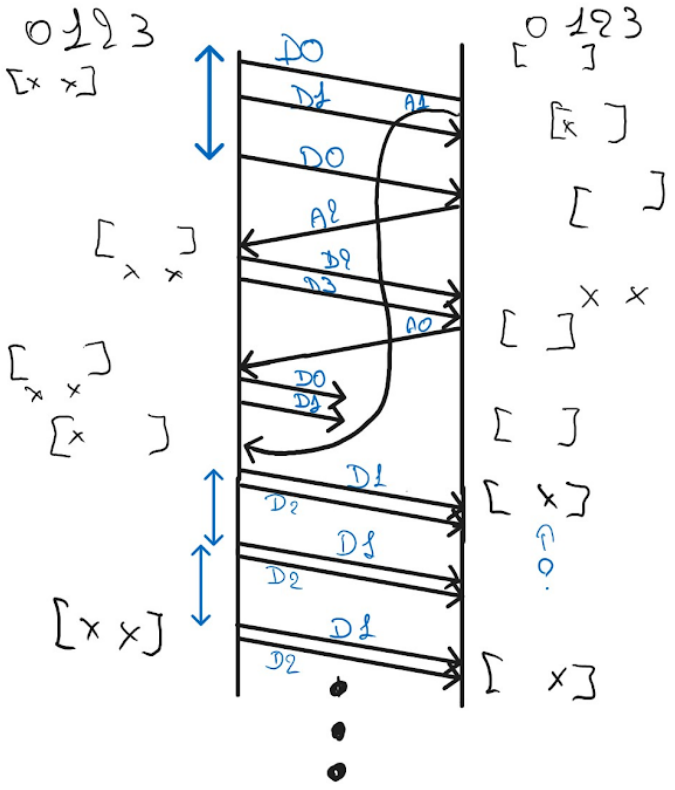
Le schéma a été simplifié et toutes les trames n'ont pas été représentées pour le simplifier. Dans le code du TP, on constate que lorsqu'une trame est reçue, un timer d'acquittement est lancé automatiquement. De ce fait, dans le schéma précédent, l'émetteur des données se rend compte qu'il y a un problème dans l'envoi des données.

CASSESE Charly

BEUF Antoine

P25

CODE CM



Dans le code du cours, l'émetteur envoie en boucle les données à la fin du temporisateur. Cependant, comme les timer d'acquittements ne se lancent que lorsque la première trame de la fenêtre de réception est reçue, alors aucun timer d'acquittement ne se lance. Par conséquent, l'émetteur ne se rend jamais compte que le récepteur ne peut pas recevoir ses données… la situation est bloquée.

Nous nous sommes rendu compte lors de la partie pratique qu'il y avait des cas de figure plus facile.. Voir partie 4.

CASSESE Charly

BEUF Antoine

P25

PARTIE PRATIQUE : TP1 NE 424

2.Fenêtres et tampons

1)

Nous avons représenté un schéma détaillant un échange en duplex de la communication entre A et B. Pour obtenir ce schéma, nous avons exécuté le code avec la commande suivante : ./swp 12 0 0 7 1

CASSESE Charly

BEUF Antoine

P25

CASSESE Charly

BEUF Antoine

P25

Les data sont représentées de la manière suivante D X Y Z où

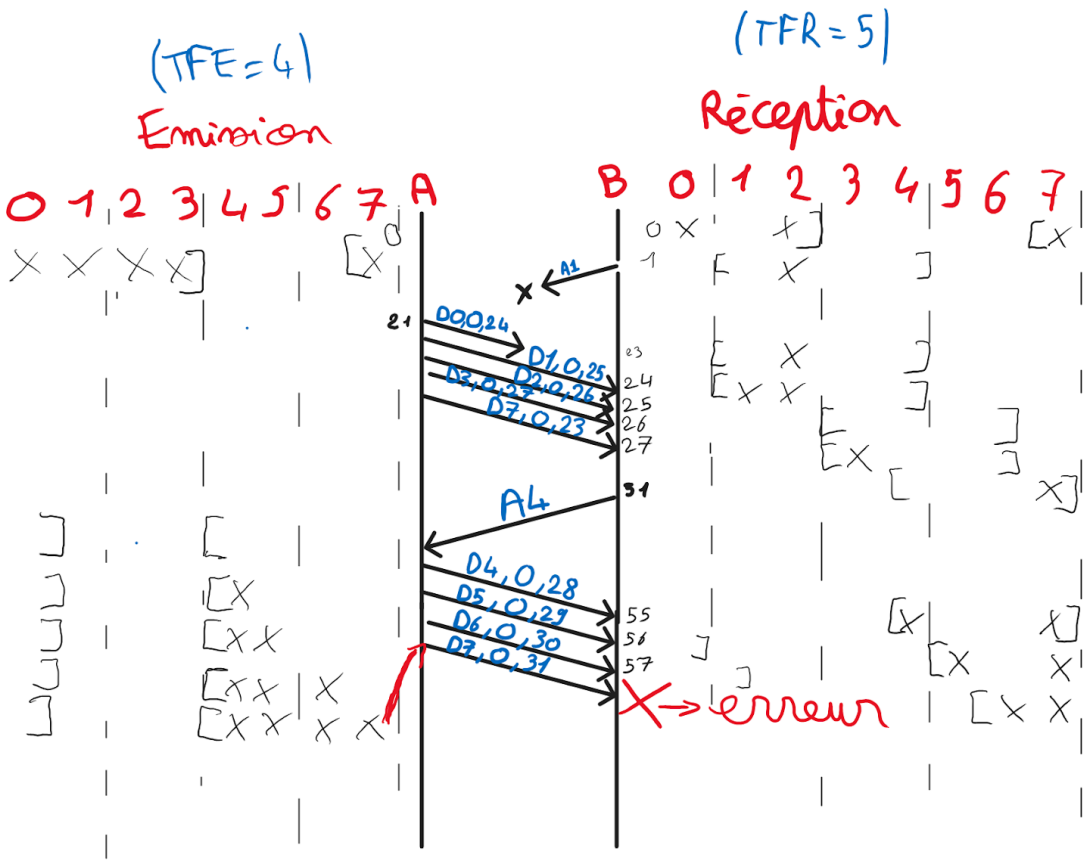
X = Le numéro de séquence correspondant à la donnée

Y = Le numéro d'acquittement "piggybacké"

Z = Compteur de trames envoyés

2) Nous avons modifié la taille de la fenêtre d'émission (passée de 4 à 5) dans le code swp.c et avons lancé une simulation de notre nouveau code swp avec un taux de perte de 25 %

avec la commande : ./swp 600 25 0 7 0



CASSESE Charly

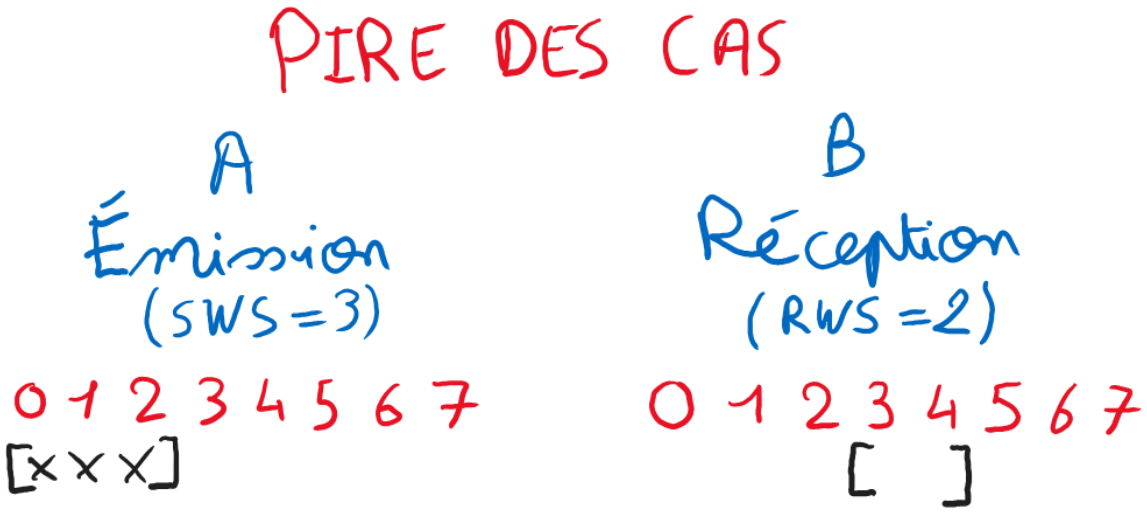
BEUF Antoine

P25

Nous pouvons constater que B stocke dans sa fenêtre de réception D,7,0,'23' alors qu'elle ne devrait pas. Lorsque la DATA D7,0,"31" est reçue par B, il constate qu'il a deux données D7 qui ne correspondent pas… Il y a une erreur et le protocole s'arrête.

3)

Ici, on assigne 3 à RWS et 2 à SWS. L'idéal dans un protocole est que les fenêtres restent synchronisées au maximum. Dans le pire des cas, les fenêtres peuvent être désynchronisées jusqu'à ce que le bord gauche de RWS soit collé au bord droit de SWS (voir schéma ci-dessous).



Dans ce schéma on a représenté le pire des cas : ici A a émis les trames 0, 1 et 2 (et n'a pas encore reçu ack3). B à lui émis le ack3 et a donc décalé sa fenêtre.

Dans ce cas la on observe qu'il n'y a pas de problème (pas de chevauchement anormal).

En effet, il n'y a pas de problème car SWS+RWS<NSEQ, en effet (SWS+RWS=3+2=5) <= (NSEQ = 8).

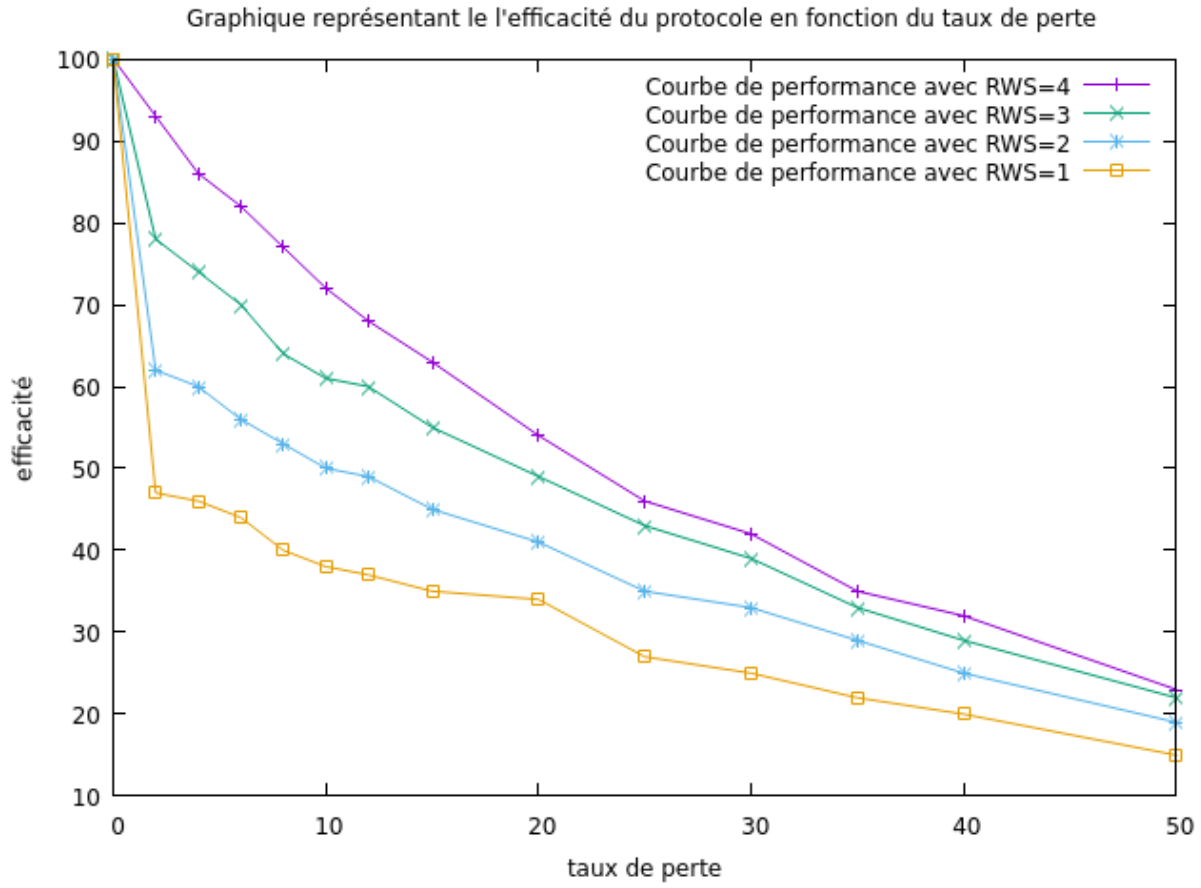
CASSESE Charly

BEUF Antoine

P25

3.Performances

2) Nous avons ci-dessous fait varier la taille de RWS et tracé différentes courbes pour comparer la performance du protocole en fonction de la taille de la fenêtre de réception à taille de fenêtre d'émission fixée.



On constate que lorsque la taille de la fenêtre de réception diminue, l'efficacité en fonction du taux de perte du protocole diminue. Cela correspond avec la partie théorique.

CASSESE Charly

BEUF Antoine

P25

3)

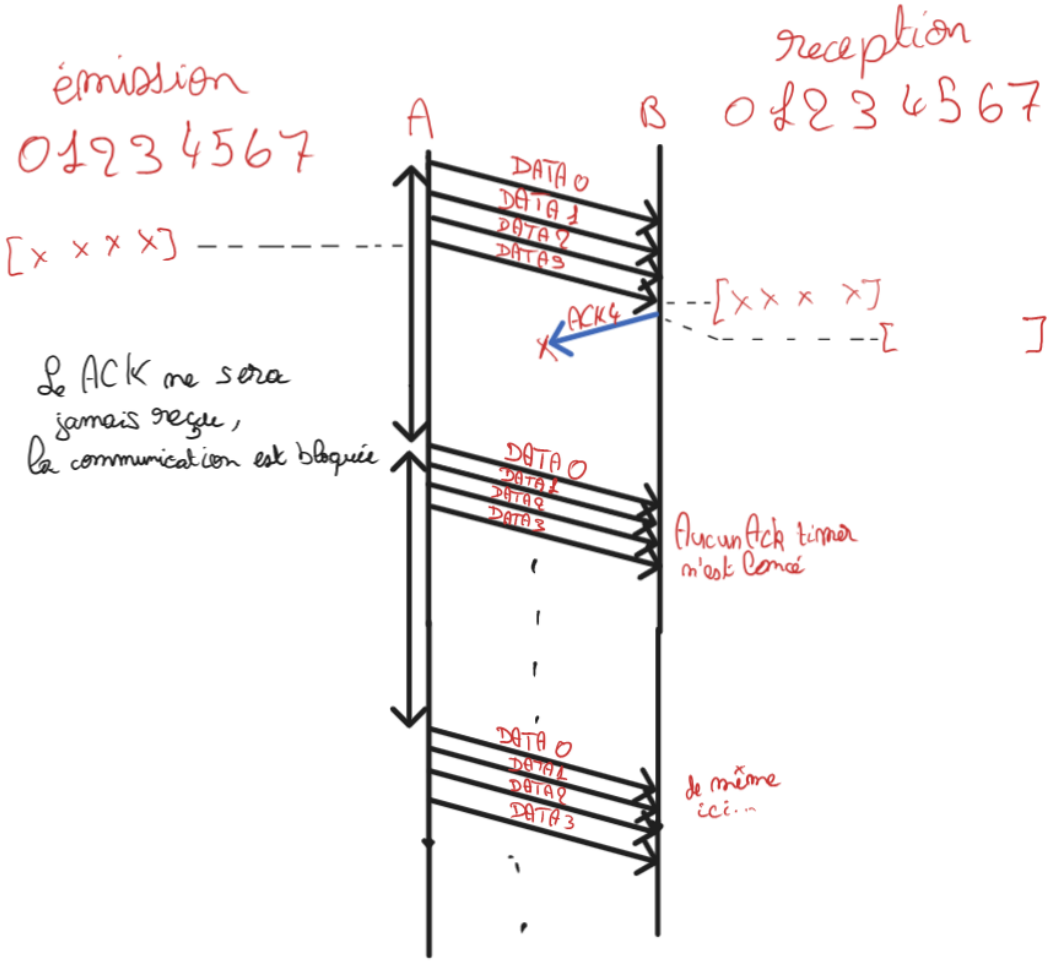
Avec un taux de perte de 50 % des paquets

- Pour RWS=4 -> 3341 retransmissions
- Pour RWS=2 -> 3351 retransmissions
- Pour RWS=1 -> 3601 retransmission

On observe que le nombre de retransmissions augmentent lorsque la taille de la fenêtre de réception diminue. Plus il y a de retransmission, plus le temps de communication augmente.

4.Bug…

En effet, le code du cours peut se retrouver bloqué.. En effet, dans la figure ci-dessous, on constate que la situation de la communication se répète:



Comme nous démarrons un timer d'acquittement uniquement lorsque nous décalons la fenêtre, si cet acquittement se perd alors que la fenêtre a glissé,

nous sommes bloqués. Dans le code du TP, cette situation ne peut pas arriver car nous démarrons un timer d'acquittement à chaque donnée reçue.

5.Modification

1)

Un acquittement négatif permet d'améliorer l'efficacité du protocole en gagnant du temps sur les retransmissions:

- Lorsque le récepteur se rend compte qu'il n'a pas reçu une trame, il en indique directement l'émetteur qui peut alors ré-émettre sans attendre la fin du timer de retransmission.

2)

- Envoie d'un NAK : Si on reçoit une trame qui n'est pas la première trame de la fenêtre de réception mais qu'elle appartient à la fenêtre de réception, alors on transmet un NAK avec le numéro de la trame qui s'est perdue pour indiquer à l'émetteur que l'on ne l'a pas reçu.
- Réception d'un NAK : Lorsque l'on reçoit un NAK, on réinitialise le timer de retransmission puis on renvoie la trame qui s'était perdue.

3)

Pour implémenter le NAK nous avons réalisé quelques modifications du code swp.c que vous trouverez ci-dessous.

- Nous avons d'abord ajouté la fonction send_nak_frame qui permet d'envoyer un NAK en utilisant un numéro de séquence.

CASSESE Charly

BEUF Antoine

P25

```
58  static void send_nak_frame(uint_t seq)
59  {
60      frame s;
61
62      s.kind = nak;
63      s.ack  = seq;
64
65      /* transmit the frame */
66      to_physical_layer(&s);
67      stop_ack_timer(); /* since we just sent one */
68  }
69
```

- Si la data reçue n’est pas la première de la fenêtre de réception mais qu'elle appartient à la fenêtre de réception alors on envoie un nak de la donnée perdue.

```
113     /*Verifier ici que l'on a bien la premiere data de TFR*/
114             if (r.seq != rw1 && in_rcv_window(r.seq, rw1, RWS)){
115               send_nak_frame(rw1);
116             }
```

- Ensuite nous gérons le cas de la réception : lorsqu'un NAK arrive. Nous vérifions d’abord le type de la data frame arrivée, si c’est un NAK, alors on arrête le ACK timer et on envoie la trame demandée(qui s’était perdue).

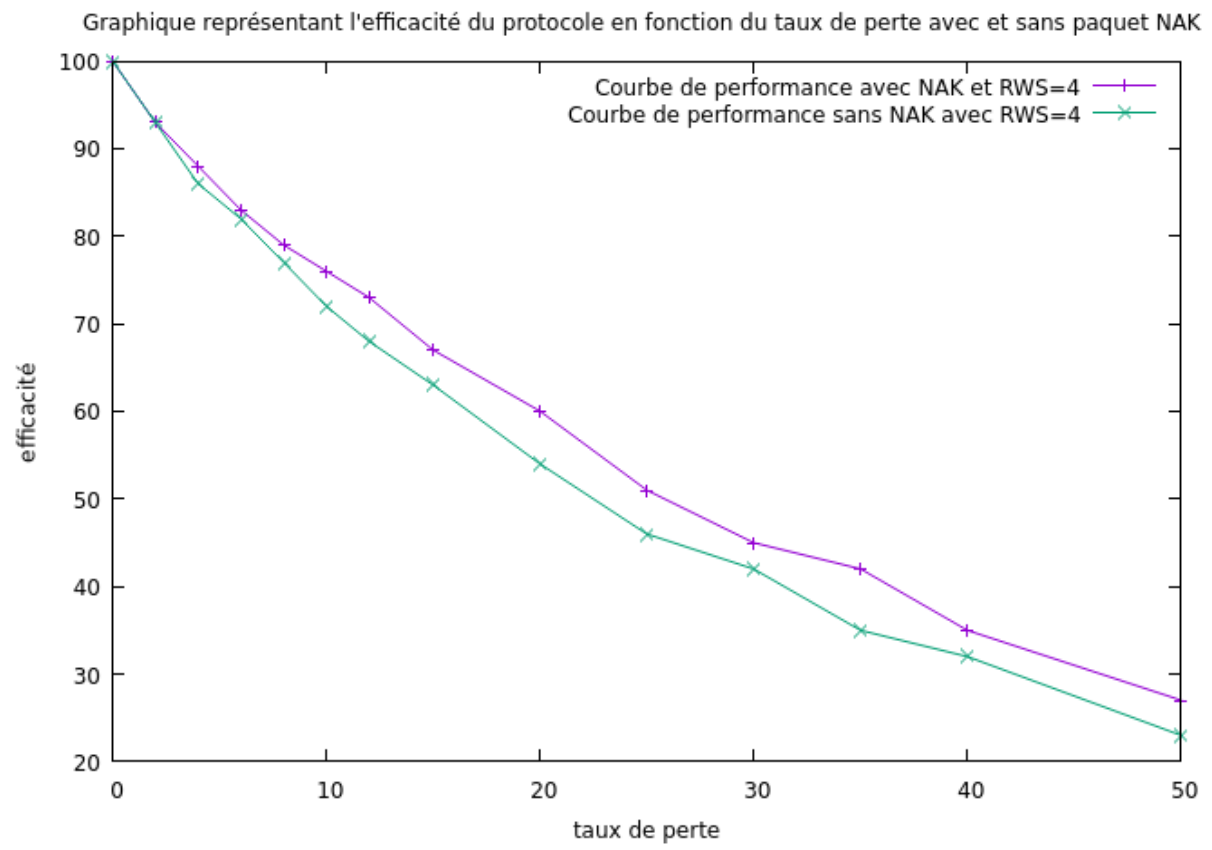
```
131       /*handle NAK*/
132       if(r.kind==nak){
133         stop_timer(r.ack);
134         send_data_frame(r.ack, rw1, snd_buf);
135       }
136
137       /* handle ACK */
138       while (is_new_ack(r.ack, sw1, sw2)) {
139           nbuffered = nbuffered - 1;
140           stop_timer(sw1);
141           inc_seq(sw1);     /* advance sender's window */
142       }
143         break;
144
```

CASSESE Charly

BEUF Antoine

P25

On peut donc observer les résultats suivant:



On peut remarquer que l'implémentation des NAK (courbe violette) permet un petit gain d'efficacité du protocole.

Pour 35 % de taux de perte, on constate un écart d'efficacité de (42-35=7%).

De plus, le principal avantage de l'implémentation du NAK est le temps de communication que nous gagnons. En effet, le NAK nous permet de ne pas attendre la fin du temporisateur et de renvoyer directement la donnée.