



# Projet Réseaux I

Modélisation d'une file d'attente



Faculté  
des Sciences

**UMONS**  
Université de Mons

Mohas Muke et Amara Houssine

## **Table des matières**

<b>Introduction.....</b>	<b>3</b>
<b>Exécution.....</b>	<b>3</b>
<b>Implémentation.....</b>	<b>4</b>
<b>Application du modèle.....</b>	<b>6</b>
1) Illustration de la gestion de la congestion.....	6
2) Illustration de la gestion de la congestion.....	8



# Introduction

**Groupe 17** : Mohas Arno Muke (240578) et Amara Houssine (221349)

**GitHub**: <https://github.com/adobbermann/projet-reseaux1-2024-gr17>

Dans le cadre du cours de réseau, nous avons dû implémenter un programme capable de simuler le comportement d'un réseau composé de systèmes terminaux et de routeurs. Le programme devrait permettre de déterminer les temps de départ et d'arrivée des paquets sur chaque entité du réseau en prenant en compte les différents types de délais.

Lors de ce projet nous avons dû mettre en œuvre les mécanismes de TCP ainsi que l'algorithme de contrôle de congestion "*AIMD*". Nous avons également adopté la stratégie "*tail drop*" afin de gérer la taille limitée de la file d'attente des routeurs.

Tout cela afin de pouvoir simuler différentes mise en situation du réseau, tel que les mécanismes de TCP, le goulot d'étranglement etc.

## Exécution

Avant d'exécuter le programme, l'utilisateur doit modéliser le réseau (les instructions sont expliquées dans la section "Implémentation"). Une fois le réseau modélisé, le code est prêt à être exécuté.

Ensuite, le simulateur crée une liste d'événements fonctionnant selon le concept LIFO (Last In, First Out). Chaque événement attend la fin de l'événement précédent avant de commencer.

Avant de commencer la transmission, le simulateur prépare déjà les colonnes pour notre tableau de résultats. Puis, le simulateur démarre la transmission des paquets, qui variera selon que les mécanismes TCP/AIMD sont implémentés ou non.

En démarrant la transmission, le simulateur commence à compter le temps et les délais (propagation, transmission) afin d'insérer les valeurs dans notre tableau de résultats et de vérifier s'il y a des pertes de paquets. Les principaux calculs et détections de pertes de paquets seront effectués dans notre classe `'Router.py'`.

Après la transmission d'un paquet, le simulateur crée une ligne contenant tous les temps de départ et d'arrivée, la position du paquet dans la file d'attente du routeur et si le paquet a été perdu.

Ce processus se répétera jusqu'à ce que la liste des événements soit vide après l'événement "end\_transmission".

À la fin, nous obtiendrons tous les journaux pour chaque paquet et notre tableau de résultats final.

## Implémentation

Nous avons implémenté un simulateur sous la forme d'un programme informatique écrit en Python.

De plus, nous avons utilisé la notion de classe pour modéliser les différents composants du réseau (hôtes, routeurs, liens, paquets, événements, tcp + aimd mécanismes).

### Modélisation d'un réseau

Pour modéliser un réseau dans ce projet de simulation d'envoi de paquets en Python, nous utilisons une classe principale appelée **Simulator**, qui sert de lanceur et intègre les méthodes des classes **Host**, **Router** et **Link**.

Les méthodes:

1. **Créer un hôte** : `add_host(host_id)`, où `host_id` est le nouvel identifiant de l'hôte.
2. **Créer un routeur** : `add_router(router_id, queue_size)`, où `router_id` est l'identifiant du routeur et `queue_size` est la taille de la file d'attente.
3. **Créer un lien** : `add_link(link_id, [start, end], distance)`, où `link_id` est l'identifiant du lien, `start` et `end` représentent respectivement le nœud de départ et le nœud de fin, et `distance` est la distance du lien.

Chaque nœud doit avoir la forme `(id, type, r/s)` :

- **id** correspond à l'identifiant de l'hôte ou du routeur créé.
- **type** peut être **host** ou **router**.
- **r/s** indique si le nœud est un récepteur (**r**) ou un émetteur (**s**).

### Envoi de paquets

Pour simuler l'envoi de paquets, à travers le réseau, entre deux hôtes en prenant en compte les caractéristiques des liens (distance, vitesse de propagation, débit de transmission) ainsi que les délais engendrés par les files d'attente des routeurs traversés, il suffit d'utiliser la method `send_packet` :

```
hosts[start_id].send_packet(end_id, [packets], is_tcp,  
                             is_aimd, interval_enabled)
```

De plus, un utilisateur peut envoyer des paquets en flux (sans activer les mécanismes TCP/AIMD) avec `interval_enabled`. ***Il est important d'activer cette fonctionnalité pour vérifier les intervalles entre les paquets.***

### **Fiabilité des transmissions**

Nous avons implémenté les mécanismes TCP assurant la fiabilité des transmissions de données (pipelining des paquets, acquittements et retransmissions).

*L'implémentation permet de désactiver ces mécanismes.*

### **Contrôle de congestion**

Nous gérons la congestion du réseau avec l'algorithme Additive Increase Multiplicative Decrease (AIMD) qui contrôle la taille de la fenêtre de congestion.

*L'implémentation permet de désactiver ces mécanismes.*

### **Gestion des files d'attente**

Étant donné la capacité limitée des files d'attente des routeurs, nous devons gérer le rejet des paquets en utilisant la stratégie de gestion de file d'attente "tail drop".

### **Choix d'implémentation et difficultés rencontrées**

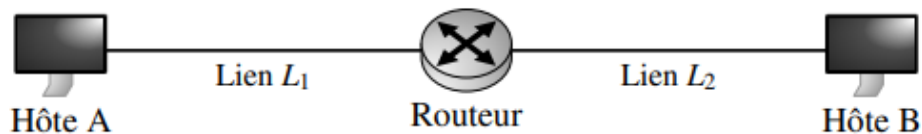
Nous avons décidé de faire de la classe `Simulator` la classe parente, afin de faciliter l'accès à toutes les méthodes nécessaires avec simplement `simulator.{méthode}`.

Les principales difficultés sont apparues lors de l'implémentation des mécanismes TCP et AIMD. Il était crucial de surveiller la perte de paquets, ce qui déclenchait la réaction de l'algorithme AIMD et lançait ensuite la retransmission pour garantir que tous les paquets étaient envoyés, une caractéristique essentielle du mécanisme TCP.

## **Application du modèle**

### **1) Illustration de la gestion de la congestion**

Pour l'application du modèle, nous avons dû prendre la topologie de réseau suivante et appliquer les différentes situations ci-dessous :



### Situation 1 : Illustration du goulot d'étranglement :

Pour cette situation, nous avons dû envoyer deux paquets consécutifs à partir d'un hôte A et montrer que l'écartement entre ces 2 paquets est influencé par le débit du goulot d'étranglement (lien avec le plus petit débit).

Nous avons donc choisi le lien 2 ( $L_2$ ) comme goulot d'étranglement. Cela signifie que tous les paquets vont être envoyés au débit de  $L_2$ . Pour montrer cette influence qu'à  $L_2$  nous avons comparé l'intervalle de temps d'arrivée sur l'hôte B qui séparait les paquets et avons dû montrer que cette intervalle ( $\Delta t$ ) correspondait bien à la formule suivante :

$t = \frac{L}{\min(R_i)}$  qui correspond au temps de transmission sur le lien 2 (avec  $L$ , taille d'un paquet, et  $R_i$ , débit des liens)

k	dép1	arR1	dépR1	pos	drp	ar2
0	0.00	0.40	0.41	0	No	0.42
1	0.40	1.20	1.21	1	No	1.22

Packets:  $k_0 = 40$  bits et  $k_1 = 40$  bits

Liens:  $L_1 = 100$  Mbps et  $L_2 = 50$  Mbps

Nous avons observé que  $\Delta t = 0.8$

Comme le goulot est sur le lien 2, la formule devient :  $t = \frac{L}{R_2} = \frac{40}{50} = 0.8$

Nous avons bien que  $\Delta t = \frac{L}{\min(R_i)}$  qui montre bien l'influence du goulot d'étranglement malgré le débit ( $R_1$ ) du lien 1 est plus rapide que celui du lien 2

### Situation 2 : Saturation de $L_2$ avec goulot d'étranglement :

Dans cette situation nous avons dû saturer le lien 2, afin de montrer que la file d'attente croissait et que par conséquent, à un moment donné, des paquets étaient jetés. Pour se faire, nous avons une colonne position qui montre la position du paquet dans la file d'attente. Si il n'y avait pas de saturation du lien 2. les paquets seraient tous à la position 0 étant donné que le paquet serait directement transmis du routeur à l'hôte B, mais le débit du lien 2 ( $R_2$ ) est tellement faible par rapport au lien 1 ( $R_1 > R_2$ ), que l'arrivée des paquets dans la file d'attente est plus rapide que leur transmission vers l'hôte B, ce qui provoque une saturation dans la file d'attente et de ce fait des paquets sont jetés.

Router: R1 (taille = 2)

Liens:  $L_1 = 100$  Mbps et  $L_2 = 50$  Mbps

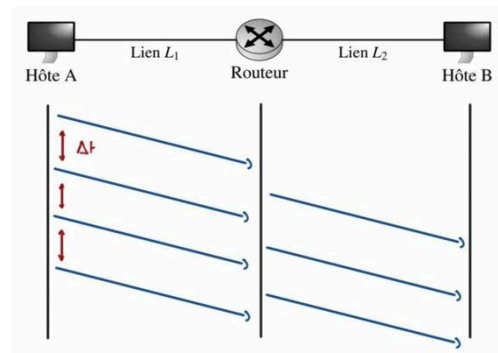
Packets : ['Hello', 'Hello', 'Hello', 'Hello']

k	dép1	arR1	dépR1	pos	drp	ar2
0	0.03	1.99	2.10	0	No	2.11
1	2.13	4.09	4.21	1	No	4.22
2	4.25	6.21	6.36	-	Yes	-.--
3	6.38	8.34	8.45	-	Yes	-.--

Dans l'image ci-dessus on a pu observer que la file croissait et que les paquets étaient jetés à partir du 3ème paquet.

**Situation 3** : Saturation de  $L_2$  sans congestion :

Dans cette situation-ci, nous devons considérer  $L_2$  comme goulot et le saturer en envoyant des paquets espacés du plus petit intervalle évitant la congestion. Si nous avions une congestion, cela était dû au fait que nous envoyions des paquets trop rapidement, il nous a fallu donc calculer l'intervalle de temps d'envoi des paquets afin d'éviter cette congestion. L'accumulation dans la file d'attente du routeur est dû au fait que le débit du lien 1 est plus grand que le débit du lien 2 ( $R_1 > R_2$ ). Donc pour éviter la congestion, il faut envoyer des paquets à intervalle  $\Delta t$  tel que  $\Delta t = \frac{L}{R_2}$  (avec  $L$ , taille d'un paquet, et  $R_2$ , débit du lien 2).



L'intervalle minimal entre les paquets doit être égal au temps de transmission d'un paquet pour éviter la congestion.

**La même topologie que dans “Situation 2”, mais R1 (taille = 5)**

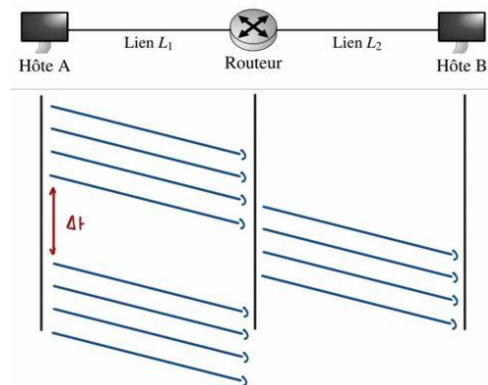
$$\Delta t = \frac{40 \text{ bits}}{50 * 10^6 \text{ bps}} = 0.0000008 \text{ ou } 8\text{e-}7 \text{ s.}$$

INTERVAL : 8e-07s

k	dép1	arR1	dépR1	pos	drp	ar2
0	0.00	0.20	0.21	0	No	0.22
1	0.19	0.39	0.40	1	No	0.41
2	0.39	0.59	0.60	0	No	0.61
3	0.60	0.80	0.81	1	No	0.82

**Situation 4** : Rafales de paquets avec  $L_2$  comme goulot d'étranglement :

Pour ce scénario, nous avons  $R_2$  comme goulot et nous devons le saturer en envoyant plus des paquets mais des rafales de paquets espacé du plus petit intervalle, étant donné que ce ne sont plus de simples paquets envoyé l'un à la suite l'autre, pour éviter la congestion il va falloir calculer l'intervalle, non plus entre les paquets mais entre les rafales de paquets. On a donc conclu que cette intervalle  $\Delta t$  était déterminée par formule suivante :  $\Delta t = \frac{N.L}{\min(R_i)}$  qui devient dans notre cas :  $\Delta t = \frac{N.L}{R_2}$  (où N correspond à la taille d'une rafale (nombre de paquet), L, la taille d'un paquet, et  $R_2$ , débit du lien 2)



**La même topologie que dans "Situation 2", mais  $R_1$  (taille = 5)**

$$\Delta t = \frac{4 * 40 \text{ bits}}{50 * 10^6 \text{ bps}} = 0.0000032$$



k	dép1	arR1	dépR1	pos	drp	ar2
0	0.00	1.20	0.04	1	No	1.26
1	1.24	2.40	0.08	2	No	3.74
2	3.72	3.60	0.12	3	No	7.46
3	7.45	4.80	0.15	4	No	12.42

## 2) Illustration de la gestion de la congestion

### 1. TCP sawtooth:

Hôtes: H1 et H2

Routers: R1(taille = 5), R2(taille = 2)

Liens: L1(distance = 5), L2(distance = 3), L2(distance = 5), L3(distance = 2)

Packets: **"Hello"** , **"World"**.

Schema:

Hôte1 ----- Lien1 ----- Routeur1 ----- Lien2 ----- Routeur2 ----- Lien3 ----- Hôte2

Résultats:

k	dép1	arR1	dépR1	pos	drp	arR2	dépR2	pos	drp	ack	ar2
0	0.00	0.06	0.21	0	No	0.34	0.46	0	No	Yes	0.72
1	0.52	0.64	0.76	1	No	0.94	1.07	1	No	Yes	1.35

Les logs:

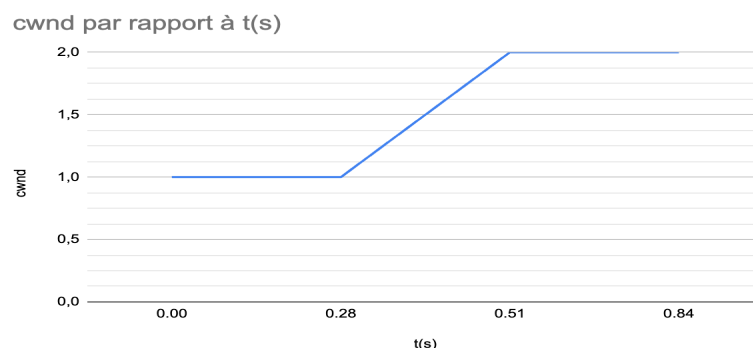
ROUTER: 1 Congestion window size: 1 at time: 0.00 seconds

ROUTER: 2 Congestion window size: 1 at time: 0.28 seconds

ROUTER: 1 Congestion window size: 2 at time: 0.52 seconds

ROUTER: 2 Congestion window size: 2 at time: 0.82 seconds

Le graphe:



### 2. Équité TCP

Hôtes: H1, H2 et H3

Routers: R1(taille = 5), R2(taille = 1)

Liens: L1(distance = 5), L2(distance = 3), L2(distance = 5), L3(distance = 2)

**Packets: "The", "university", "of", "Mons".**

Schema:

Hôte1 ----- Lien1 ----- Routeur1 ----- Lien2 ----- Routeur2 ----- Lien3 ----- Hôte2

|  
----- Lien4 ----- Hôte3

Résultats:

k	dép1	arR1	dépR1	pos	drp	arR2	dépR2	pos	drp	ack	ar2	ack	ar3
0	0.00	0.04	0.18	0	No	0.28	0.40	0	No	Yes	0.66	Yes	0.66
1	0.46	0.60	0.70	1	No	0.86	1.00	0	Yes	---			
									No	Yes	1.26	Yes	1.26
	1.06	1.10	1.21	2	No	1.32	1.45	0	No	Yes	1.85	Yes	1.85
2	1.50	1.55	1.67	3	No	1.78	1.91	0	No	Yes	2.19	Yes	2.19

● - l'envoi de "university"

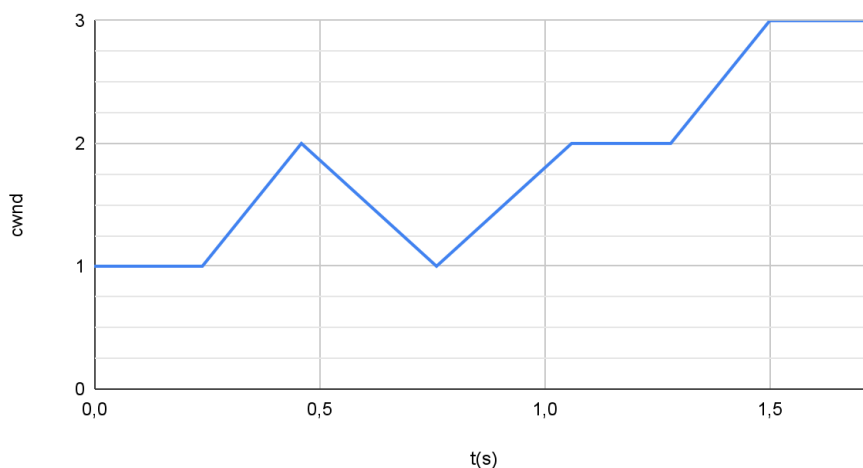
● - retransmission de "of"

Les logs:

ROUTER: 1 Congestion window size: 1 at time: 0.00 seconds  
ROUTER: 2 Congestion window size: 1 at time: 0.24 seconds  
ROUTER: 1 Congestion window size: 2 at time: 0.46 seconds  
ROUTER: 2 Congestion window size: 1 at time: 0.76 seconds (due to package loss)  
ROUTER: 1 Congestion window size: 2 at time: 1.06 seconds  
ROUTER: 2 Congestion window size: 2 at time: 1.28 seconds  
ROUTER: 1 Congestion window size: 3 at time: 1.50 seconds  
ROUTER: 2 Congestion window size: 3 at time: 1.73 seconds

Le graphe:

cwnd par rapport à t(s)



### 3. Bandwidth Delay Product

L'utilisation d'une file d'attente de taille égale au Bandwidth Delay Product dans un routeur permet d'optimiser l'efficacité du réseau en minimisant les pertes de paquets et en maximisant l'utilisation de la bande passante, ce qui améliore les performances globales du réseau.

L'article d'Appenzeller argue que cette règle traditionnelle ( $B = RTT \times C$ ) est obsolète pour les routeurs de “backbone” en raison du grand nombre de flux multiplexés ensemble sur un seul lien.

Ils montrent que pour un lien avec  $n$  flux, la taille nécessaire des tampons est en fait :

$$B = \frac{RTT \times C}{\sqrt{n}}$$

#### Conséquences et Avantages

##### 1. Réduction de la Taille des Tampons :

Un lien de 2,5 Gb/s transportant 10 000 flux pourrait réduire ses tampons de 99 % avec une différence négligeable en termes de débit.

##### 2. Implémentation plus Facile :

Un lien de 10 Gb/s transportant 50 000 flux ne nécessiterait que 10 Mbits de tampons, facilement réalisables avec des SRAM rapides et intégrées.

##### 3. Stabilité et Efficacité :

Des tampons plus petits réduisent les délais de mise en file d'attente et la variance de ces délais, stabilisant les algorithmes de contrôle de congestion.

##### 4. Efficacité Énergétique et Coût :

Utiliser des SRAM rapides permet une conception de routeur plus efficace en termes de coût et de consommation d'énergie.

La règle traditionnelle  $B = RTT \times C$  provient des principes fondamentaux du contrôle de congestion TCP et de la théorie des files d'attente, visant à maintenir les liens occupés et à minimiser les pertes de paquets. Cependant, l'évolution des réseaux et l'augmentation du nombre de flux multiplexés nécessitent une réévaluation de ces principes pour optimiser les performances dans les conditions modernes.