TPs Évaluation de performances réseaux

Édouard Lumet, [redacted] - IR 2020

TP1 : Modèles simples de réseaux à commutation de paquets

Simulation de files M/M/1, M/D/1

2) Le temps de simulation choisi est de 800s. Dans ce cas-là, l'intervalle de confiance concernant la taille de la file est de 9,2%, soit inférieur à 10%, de même pour le temps de réponse dont l'intervalle de confiance est de 5,3%. On obtient les résultats suivants :

ρ	λ	E[R]	E[L]	E[R] théo	E[L] théo
0,3	9,9	0,045811	0,480834	0,043	0,43
0,6	19,8	0,080371	1,716164	0,075	1,5
0,9	29,7	0,273970	8,710278	0,303	9

À noter que les temps de réponses et nombres de clients moyens théoriques sont calculer par : $E[L] = \frac{\rho}{1-\rho}$ et $E[L] = \frac{E[L]}{\lambda}$.

Pour une durée supérieure, T=1500s par exemple, les intervalles de confiance restent inférieurs à 10% et on obtient les résultats :

ρ	λ	E[R]	E[L]	E[R] théo	E[L] théo
0,3	9,9	0,046406	0,493661	0,043	0,43
0,6	19,8	0,078358	1,661344	0,075	1,5
0,9	29,7	0,289408	9,222031	0,303	9

On peut observer que la précision augmente quand la durée de la simulation augmente. Par ailleurs, à mi-charge, l'intervalle de confiance est meilleur qu'à basse ou forte charge.

3) Dans le cas d'une file M/D/1, les temps de réponses et nombres de clients moyens sont calculés par : $E[L] = \frac{\rho(2-\rho)}{2(1-\rho)}$ et $E[R] = \frac{E[L]}{\lambda}$. On obtient les résultats suivants :

ρ	λ	E[R]	E[L]	E[R] théo	E[L] théo
0,3	9,9	0,036696	0,363108	0,0368	0,3643
0,6	19,8	0,053713	1,070455	0,053	1,05
0,9	29,7	0,193979	5,774169	0,167	4,95

Ici nous avions une durée de simulation de 1500s car l'intervalle de confiance était satisfaisant. Cependant, on observe une différence entre valeurs simulées et valeurs théoriques relativement importante en cas de forte charge. En effet, la durée de simulation est trop faible face à la durée de convergence du système, pour des paquets de taille constante.

TP2 : Flots TCP

NB: Suite à une perte de l'ensemble des copies des graphes produits lors de la séance, l'ensemble des réponses est issue des notes prises lors du TP. Les réponses sont donc fondées sur nos graphes et observations de ceux-ci même s'ils sont absents ici.

1. Contrôle de congestion TCP

Question 1

- Le mécanisme de *slow-start* sert à éviter les congestions réseaux. Il consiste au démarrage progressif de l'envoi des segments, en commençant avec une fenêtre initiale (*Initial Window* ou *IW*) très petite (1 ou 3 par exemple) puis en l'augmentant au fur-et-à-mesure jusqu'à atteindre la capacité du réseau.
- Le passage du *slow-start* au *congestion avoidance* est visible par un changement de courbe d'évolution de la taille de la fenêtre de congestion (*cwnd*). En effet, lors du *slow-start* la courbe suit une croissance exponentielle tandis qu'elle suit une croissance linéaire lors du *congestion avoidance*.

Question 2

La variation de la taille de la fenêtre est cyclique car à chaque fois que l'on atteint la capacité du réseau à l'issue du *congestion avoidance*, on se trouve en congestion. On repart alors avec *cwnd* = *IW* puis on repart en *slow-start*, en *congestion avoidance* et ainsi de suite.

Question 3

Lorsque le délai de propagation (RTT) augmente sur le premier lien par exemple (donc avant le goulot d'étranglement), et ce de même lorsque le débit de ce premier lien est inférieur au débit du lien du goulot, on observe après une certaine durée que TCP reste en *congestion avoidance* car la file d'attente présente au niveau du goulot ne peut plus se remplir. En l'absence de congestion, TCP ne revient pas à la valeur initiale de la fenêtre et continue d'augmenter la taille de la *cwnd* linéairement.

Question 4

On repère la retransmission suivante en visualisant la taille de la *cwnd* :

- lorsqu'un RTO se déclenche, suite à un certain temps lorsqu'un ACK n'est pas reçu, la *cwnd* est alors réinitialisée à 1, soit la valeur de la fenêtre initiale.

Un inconvénient important est le fait que lors du *slow-start*, le débit est mal utilisé par TCP, la convergence vers la capacité du réseau est plutôt lente.

2. Versions de TCP

Question 1

L'inconvénient de TCP Tahoe est le temps perdu lors de retransmissions, à la détection d'une congestion donc. En effet, comme expliqué dans la partie précédente, TCP repart systématiquement à une valeur de fenêtre de 1. S'ajoute à cela le temps que TCP Tahoe détecte une perte. Ce temps est égal au RTO, instant à partir duquel la *cwnd* est réinitialisée.

Question 2

Avec TCP Reno, les pertes sont détectées à la fois sur déclenchement d'un RTO et sur réception de 3 ACKs dupliqués :

- lorsque 3 *ndups* (ACK dupliqués) sont reçus, TCP réémet immédiatement le segment perdu (c'est le *fast retransmit*), la valeur du *sshthresh* est réinitialisée à la moitié de la *cwnd* et TCP reprend en linéairement jusqu'à cette nouvelle valeur, en omettant la phase de *slow-start* : c'est la phase de *fast recovery*.

Question 3

La différence entre TCP Reno et TCP New Reno n'est observable que si une autre perte au moins intervient dans la même fenêtre d'émission. En effet, si une seconde perte survient, TCP Reno ne sera pas en mesure d'effectuer un second *Fast Retransmit* et le RTO sera armé. Lors de son expiration, on repassera, comme pour TCP Tahoe, à *cwnd* = 1 puis reprise en *slow-start*, etc. TCP New Reno, lui, réémet l'intégralité de la fenêtre lorsqu'une perte survient, permettant ainsi de prévenir d'éventuelles autres pertes dans la fenêtre et donc ne jamais avoir besoin de prendre en compte le RTO. Ainsi, on observe que la *cwnd* a toujours une évolution cyclique mais uniquement entre la valeur de *sshtresh* et la capacité maximale.

3. Performances de TCP

Question 1

On peut noter que deux facteurs ont un impact sur le débit de TCP, selon la version : le RTT et le taux de perte. Dans tous les cas, un RTT trop faible ne laisse pas le temps aux différentes files d'attente de se vider, dans le cas où on est en présence d'un goulot d'étranglement comme ici. Concernant le taux de perte, c'est TCP New Reno qui se démarque, avec une meilleure performance dans le cas de pertes multiples.

Question 2

Pour une taille de la file d'attente de 42, on observe 1 perte seulement pour TCP Reno comme pour TCP New Reno. Comme indiqué précédemment, one ne voit là aucune différence dans le comportement de ces deux versions TCP, ils passent tous les deux en *Fast Retransmit* dès les réception de 3 *ndups* : *sshthresh* = *cwnd* / 2. TCP Tahoe lui attend le RTO puis repart en *slow-start* à partir de *cwnd* = 1.

Pour une taille de file d'attente inférieure, on observe l'occurence de 2 pertes dans la même fenêtre d'émission. Apparaît alors la différence telle que décrite dans la question 3 de la partie précédente. Alors que TCP Reno enclenche le mécanisme de *Fast Retransmit* puis la phase de *Fast Recovery*, il repart subitement à une valeur de *cwnd* = 1, lorsque le RTO lié à la seconde perte se déclenche. TCP New Reno a quant à lui réémis l'ensemble de la fenêtre d'émission à partir de la perte, empêchant tout RTO de se déclencher et recouvrant ainsi la seconde parte par l'occasion de la première.

Question 3

Cette étude met effectivement en exergue l'importance de la taille des files d'attente des routeurs, pouvant mener à différents comportement selon la version de TCP.

4. Équité TCP

Question 1

Dans le cas où la seconde source démarre avec un décalage, dès qu'elle commence à émettre, elle monopolise le réseau, la *cwnd* de la première source s'effondre. Cependant, très rapidement les deux *cwnd* vont converger vers une même valeur médiane et vont ainsi évoluer de la même façon.

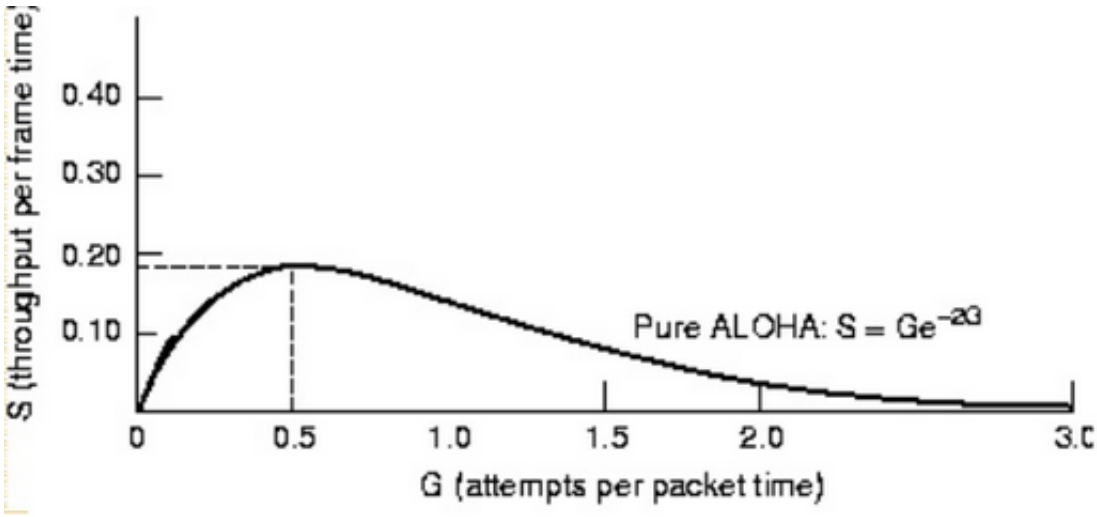
Lorsque les deux sources commencent à émettre au même instant, de façon périodique on observe que l'une des deux sources va monopoliser le réseau plus que l'autre puis inversement. En d'autres termes, on observe que la *cwnd* de l'une prend le dessus sur l'autre source, puis les deux reviennent à une valeur médiane avec que la *cwnd* de l'autre prenne le dessus à son tour.

TCP n'est donc par forcément équitable : il l'est dans le premier cas, en revanche dans le cas où les sources commencent à émettre au même instant, il ne l'est plus.

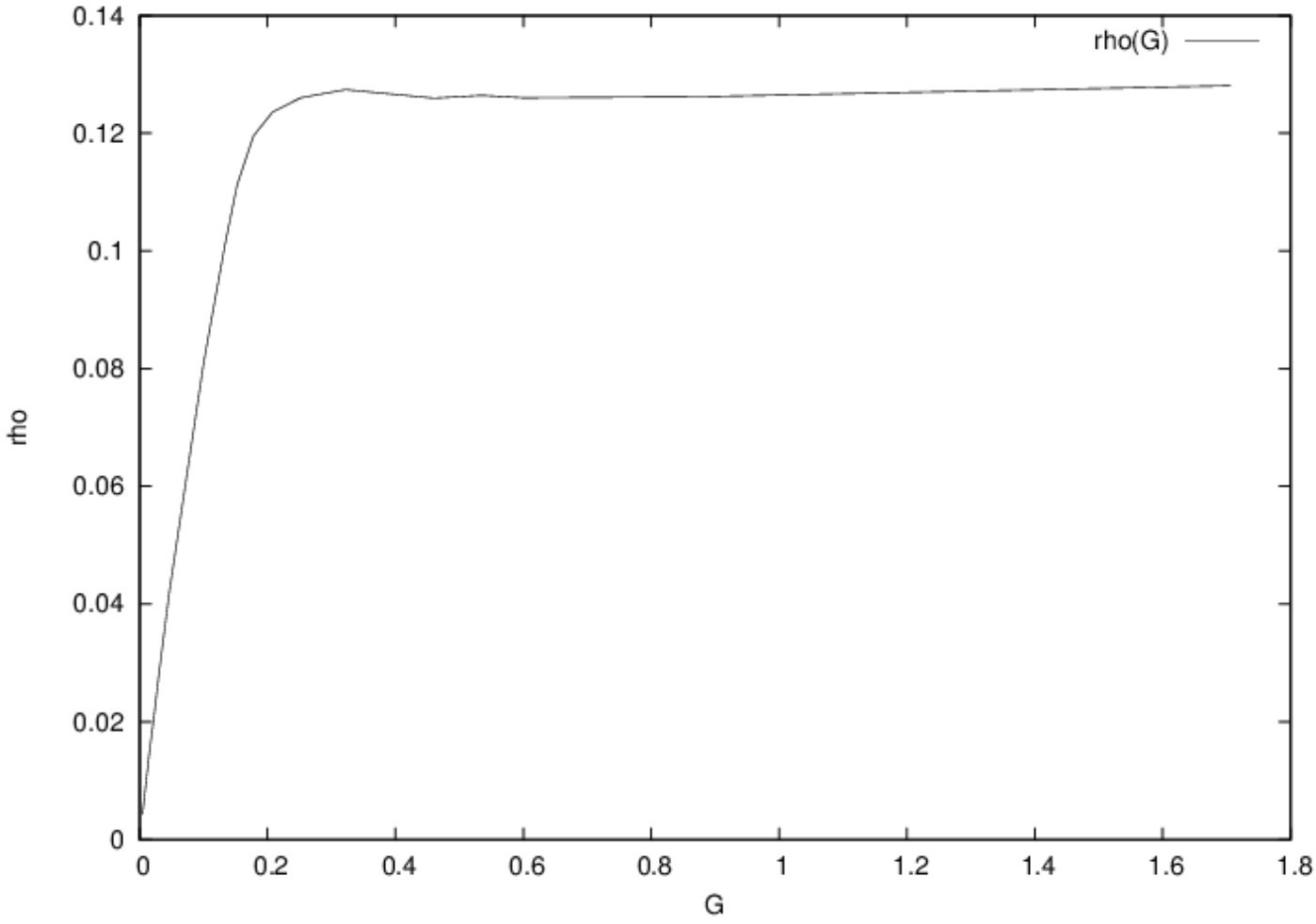
TP3 : Méthode d'accès ALOHA

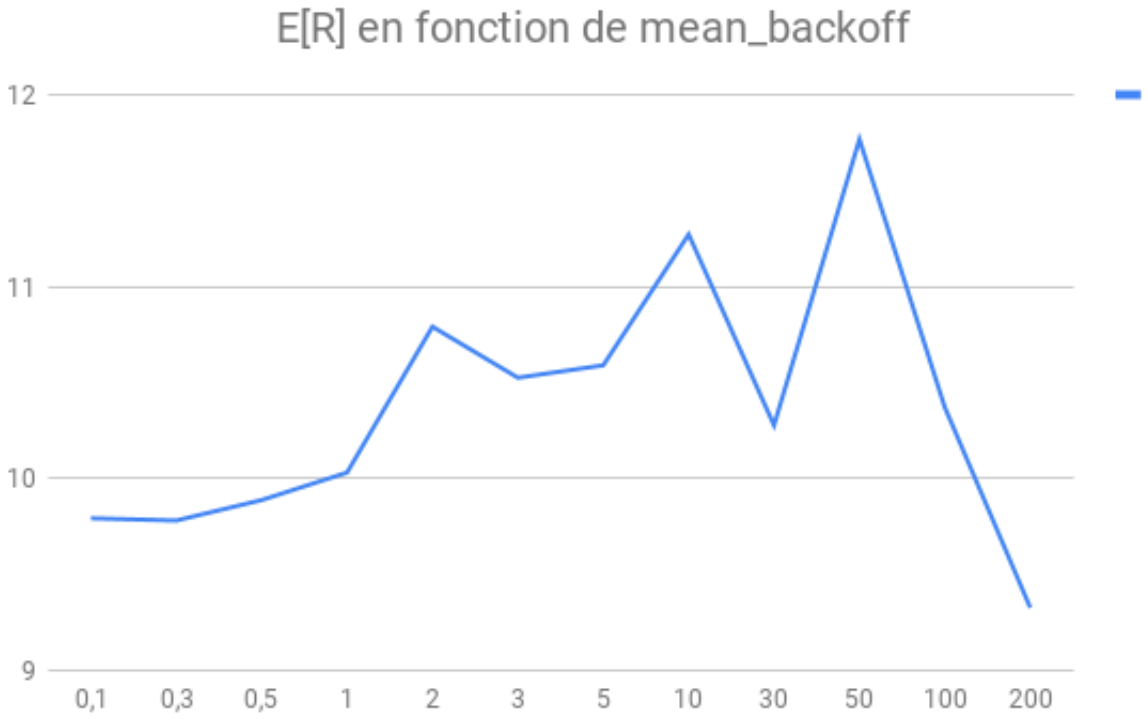
Dans ce TP, l'objectif est de comparer les performances théoriques d'Aloha à des performances simulées, à l'aide de *ns*.

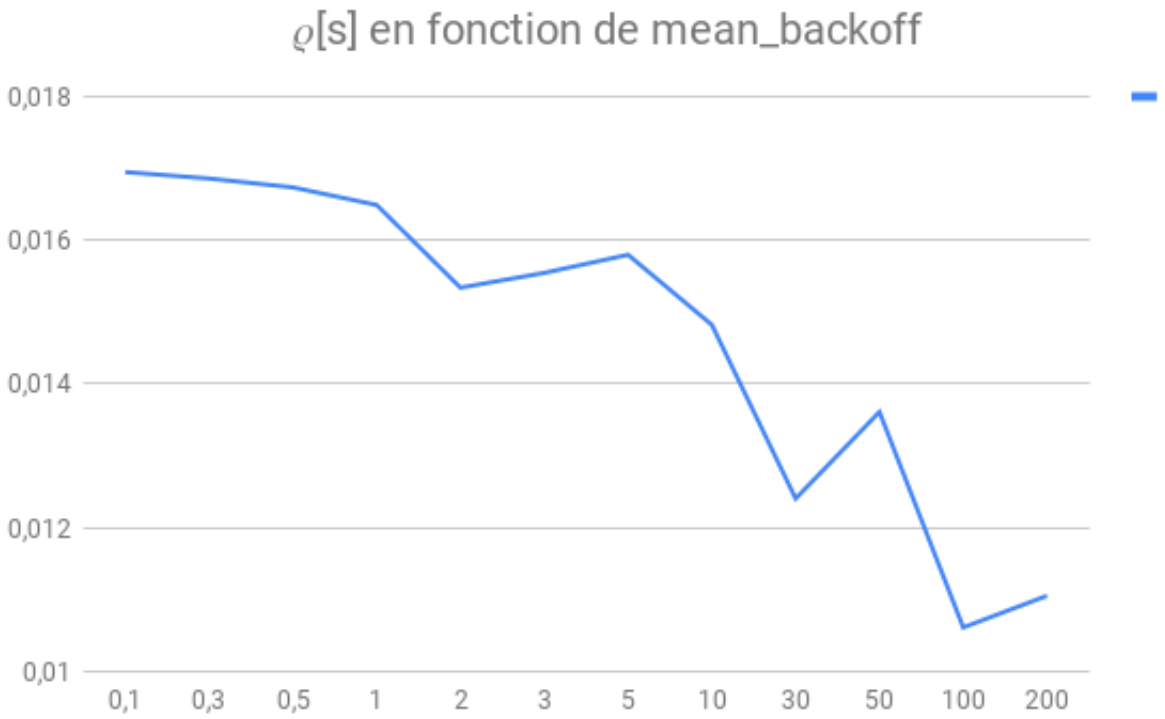
Le schéma suivant présente les performances théoriques d'Aloha. On observe que, théoriquement, la capacité maximale est de 0,18 pour une charge offerte de 0,5. Après cette valeur, les performances réduisent jusqu'à une limite de 0.



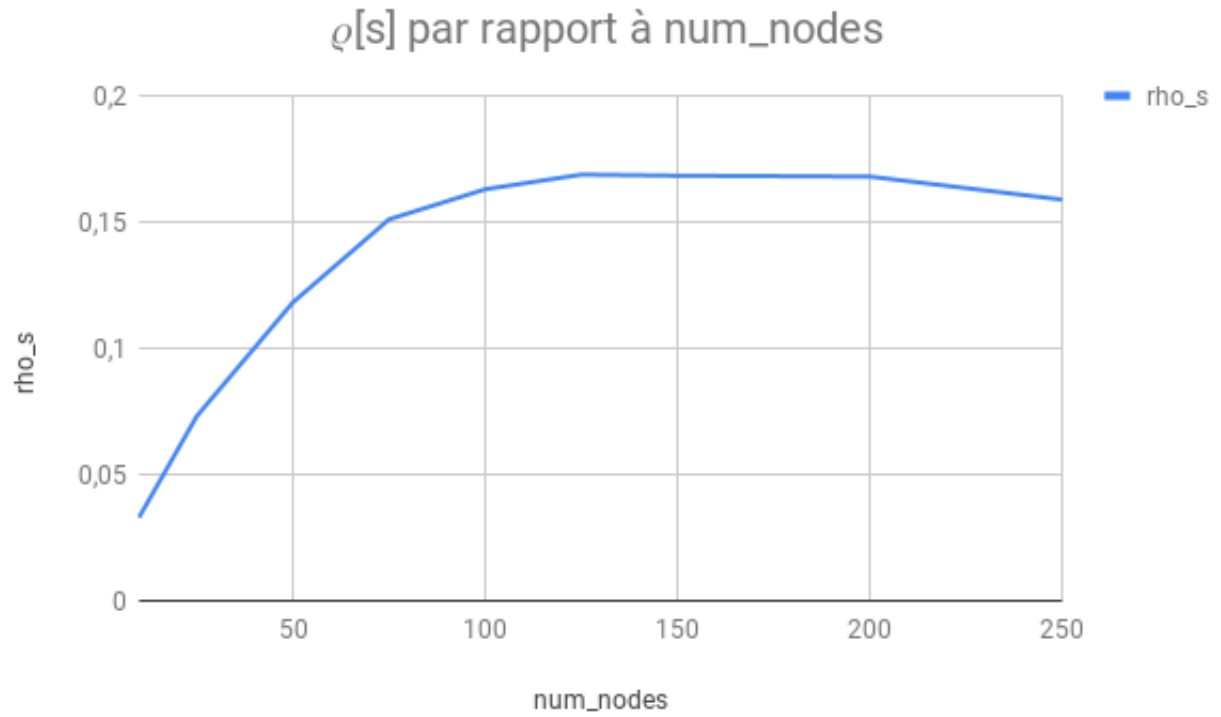
Lors d'une simulation, on observe que la capacité maximale est située aux alentours de 0,13, atteinte pour une charge offerte d'environ 0,4 et que les performances restent stables ensuite. La différence entre les performances théoriques et simulées s'expliquent par le fait que nous avons fait varier le *idle_time*, soit la durée entre les paquets. Par conséquent, plus cette durée est longue, moins le taux d'arrivée diminue, ce qui charge moins le réseau et améliore donc l'accès.


rho(G)
rho
G
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8


E[R] en fonction de mean_backoff
12
11
10
9
0,1
0,3
0,5
1
2
3
5
10
30
50
100
200

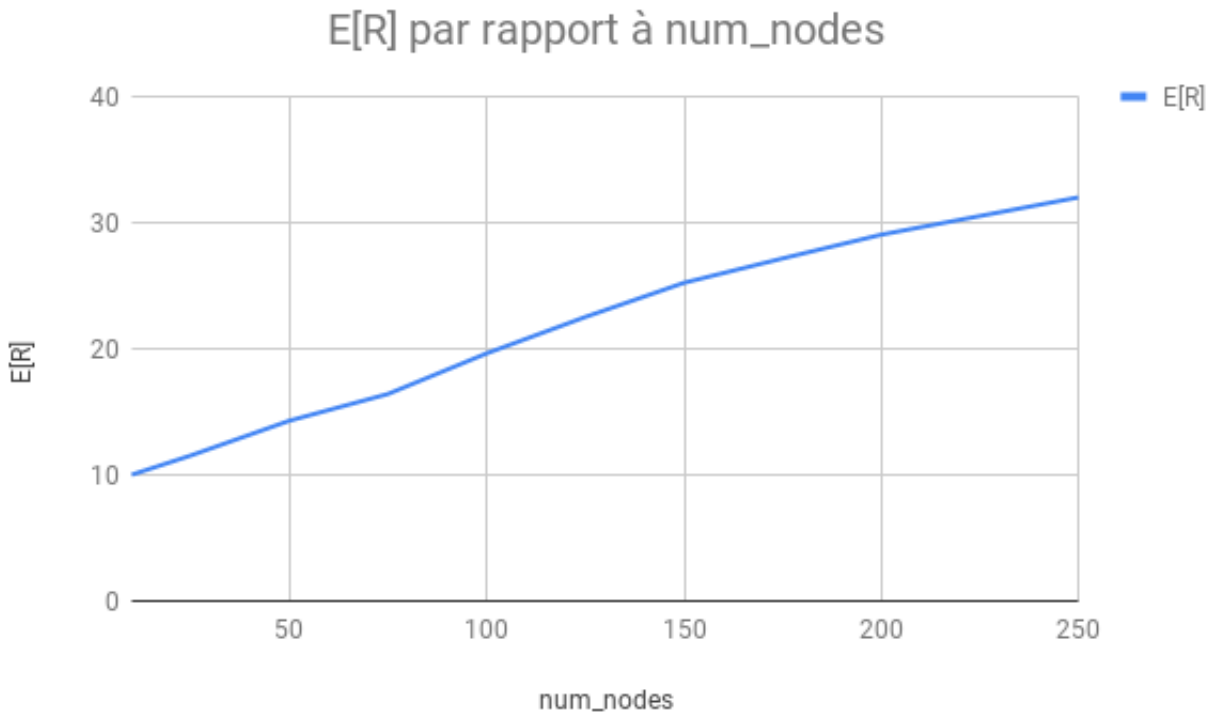


Le graphique suivant est le résultat d'une simulation présentant la charge en sortie en faisant varier le nombre de nœuds, avec un *mean_backoff* optimal (= 0,1). On observe qu'à partir d'un grand nombre d'utilisateurs (ici 200 environ), les performances commencent à chuter, ce qui correspond à une augmentation de la charge.



Ici, c'est le temps de réponse moyen en fonction du nombre de nœuds qui est observé. Trivialement, on

note qu'il augmente linéairement et significativement à mesure que le nombre de nœuds augmente.



En conclusion, on peut dire qu'Aloha présente une resistance toute relative à la charge quant à l'accès à un medium de transmission. Cependant, il est très simple à implanter, au vu de son algorithmique.