

# MPTCP

## Performances et optimisation de la sécurité avec un ordonnancement réparti dans les topologies virtualisées OpenFlow

Encadrants : S. Secci,  
Y. Bensaïb, M. Coudron,

Etudiants : R. Ly, K. Lam, Q. Dubois, S. Ravier

## Table des matières

<b>1</b>	<b>Cahier des charges</b>	<b>2</b>
1.1	Objectifs . . . . .	2
1.2	Contexte . . . . .	2
1.3	Méthodes . . . . .	2
1.4	Répartition . . . . .	3
<b>2</b>	<b>Plan de développement</b>	<b>4</b>

# 1 Cahier des charges

## 1.1 Objectifs

Les objectifs du projet sont de

- mesurer les performances de MPTCP sur différentes topologies de réseaux virtuels.
- modifier l'ordonnanceur de MPTCP pour privilégier une répartition équilibrée sur les différents sous-flots.

## 1.2 Contexte

MPTCP est une extension de TCP qui permet pour une connexion TCP donnée d'utiliser plusieurs chemins pour l'échange de données. La multiplicité des sous-flots a pour but d'améliorer le débit et d'augmenter la résilience de la connexion [1–3].

L'utilisation de MPTCP ne doit en aucun cas diminuer les performances de l'utilisateur en dessous des performances d'une simple connexion TCP ou diminuer le débit des autres utilisateurs sur le même réseau. Les performances de MPTCP dépendent en partie de l'algorithme utilisé pour la répartition des données entre les différents sous-flots ouverts [4]. Pour caractériser les performances de l'ordonnanceur de MPTCP, nous allons le tester dans différents réseaux virtualisés en utilisant dans un premier temps l'algorithme par défaut implémenté dans MPTCP dans le kernel de linux.

L'utilisation exclusive d'un chemin dans une connexion TCP permet la possibilité à un attaquant d'écouter la totalité des données échangées, d'utiliser des attaques de type rejeu ou *man in the middle*. L'emploi de MPTCP améliorerait la sécurité si les données transitaient de manière équilibrée entre les différents sous-flots. Le débit global de la connexion serait affecté car les chemins les plus lents vont ralentir le débit des chemins les plus rapides, ce qui, en contre partie, peut s'avérer moins performant qu'une simple connexion TCP. Nous allons, dans un premier temps, modifier l'ordonnanceur afin de garantir la répartition équitable des charges au détriment du débit puis analyser l'influence de cette modification sur les performances de MPTCP dans les topologies réseaux utilisés précédemment. Si nous avons le temps, nous réfléchirons s'il est possible d'obtenir un algorithme pouvant garantir une sécurité et un débit supérieur à celle d'une simple connexion TCP pour que cela soit conforme aux objectifs premiers de MPTCP.

## 1.3 Méthodes

La simulation des réseaux sera effectué via mininet qui va nous permettre de simuler un réseau OpenFlow. Pour caractériser l'influence de l'ordonnanceur sur les performances, nous utiliserons des réseaux simples où les différents sous-flots sont asymétriques et différent par une propriété à la fois : latence, débit, pertes ...

Nous testerons différents algorithmes de répartition de charge entre les sous-flots : l'algorithme implémenté par défaut (LIA), celui qui satisfait l'optimum de pareto par

rapport aux objectifs de MPTCP (OLIA, [4]) ou celui qu'on va développer qui permet de répartir de manière équilibrée les charges dans les différents sous-flots.

Nous utiliserons des scripts python pour automatiser la création des sous-flots ainsi que l'analyse des tests.

Pour implémenter l'algorithme de balancement équilibré (favorisant la sécurité), nous modifierons le code source de MPTCP : [https://github.com/multipath-tcp/mptcp/tree/mptcp\\_v0.88/net/mptcp](https://github.com/multipath-tcp/mptcp/tree/mptcp_v0.88/net/mptcp). Dans un premier temps,

- nous privilégierons une répartition équitable entre les différents sous-flots.
- il sera peut-être aussi nécessaire de gérer les problèmes liés à la retransmission lors d'une congestion par exemple,
- et ensuite mesurer les performances de notre algorithme.

Nous utiliserons *git* comme outil collaboratif pour maintenir les versions du projet.

## 1.4 Répartition

Nous allons répartir la charge de travail en deux groupes de deux personnes. Le premier groupe va édifier les réseaux virtuels ainsi que les outils d'analyses. Le deuxième groupe s'occupera du code de l'ordonnanceur, sans exclure l'aide ou la concertation entre les deux groupes.

## 2 Plan de développement

La première partie est de consulter les topologies virtualisées et de tester les performances de MPTCP en faisant varier les paramètres des sous-flots. La seconde partie est de construire un algorithme d'ordonnancement répondant à des critères de sécurité.

Les étapes du développement suivront les points suivants :

- Préparation d'une machine mininet contenant MPTCP pour l'ensemble de l'équipe.
- Lecture et compréhension du code de MPTCP et écriture de commentaires.
- Préparation de plusieurs topologies : *fat tree* pour simuler un *data center* et d'une topologie permettant de tester la concurrence entre MPTCP et TCP.
- Préparation d'une bibliothèque de tests et de mesures via l'API python
- Préparation et écriture des algorithmes d'ordonnancement
- Mesures de performances sur les différents algorithmes

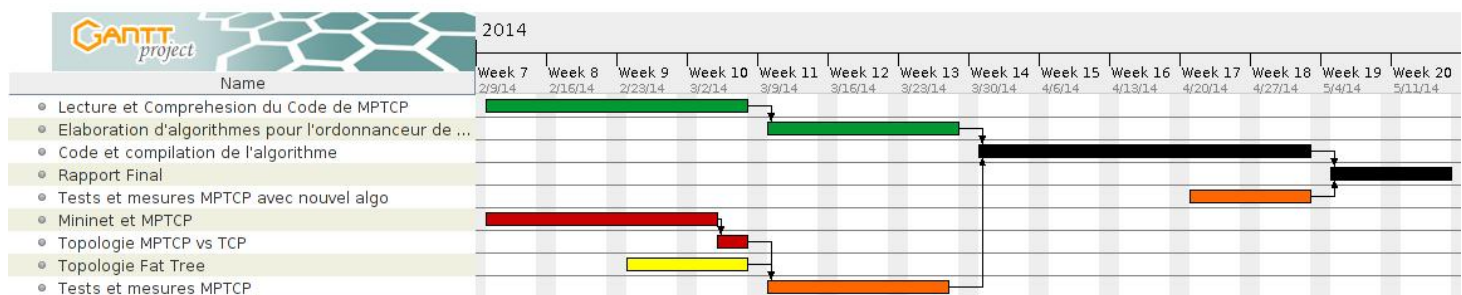


FIGURE 1 – **Diagramme de Gantt**. Les couleurs correspondent à la répartition grossière entre les membres de l'équipe : en rouge M. Ly, en jaune M. Ravier et en vert M. Dubois et M. Lam.

## Références

- [1] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, “Architectural guidelines for multipath tcp development,” *RFC 6182*, March 2011.
- [2] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “Tcp extensions for multipath operation with multiple addresses,” *RFC 6824*, January 2013.
- [3] M. Coudron, S. Secci, G. Pujolle, P. Raad, and P. Gallard, “Cross-layer cooperation to boost multipath tcp performance in cloud networks,” in *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on*, pp. 58–66, IEEE, 2013.
- [4] R. Khalili, N. Gast, M. Popovic, U. Upadhy, and J.-Y. Le Boudec, “Mptcp is not pareto-optimal : Performance issues and a possible solution,” *Networking, IEEE/ACM Transactions on*, vol. 21, no. 5, pp. 1651–1665, 2013.