# Routage haut-débit et filtrage de DDOS avec DPDK et Packet Journey

Nicolas Gonzalez, Laurent Guerby, Matthieu Herrb, Medhi Abaakouk

# Routage haut-débit et filtrage de DDOS avec DPDK et Packet Journey

**Nicolas Gonzalez**
Tetaneutral.net
Toulouse

**Laurent Guerby**
Tetaneutral.net
Toulouse

**Matthieu Herrb**
Laboratoire d'Analyse et d'Archirecture des Systèmes, Université de Toulouse, CNRS
Toulouse

**Medhi Abaakouk**
Tetaneutral.net
Toulouse

## Résumé

*La technologie DPDK (Data Plane Development Kit) a été développée initialement par Intel et est maintenant gérée collaborativement par la Linux Foundation. Elle permet de remonter le traitement des paquets en mode utilisateur plutôt que dans la pile réseau du noyau. On atteint ainsi des performances considérables en nombre de paquets par seconde sur des processeurs standards.*

*L'application Packet Journey basée sur DPDK, est un routeur logiciel libre développé entre autres par Gandi. Packet Journey permet de faire tourner un démon de routage traditionnel, par exemple le logiciel libre bird pour BGP et OSPF, et d'utiliser la table de routage produite pour décider de l'acheminement des paquets. Packet Journey fournit aussi une gestion de filtres et de limitation de débit.*

*Il est donc possible d'envisager d'utiliser du matériel serveur standard et du logiciel libre pour les fonctions routages et pare-feu haute performance plutôt que du matériel réseau spécialisé.*

*L'association Tetaneutral.net, fournisseur d'accès internet et hébergeur de données, dispose à Toulouse de plus de 20 Gb/s d'interconnexions avec l'internet. Certains sites et services hébergés étant soumis à des attaques de déni de service distribué, nous évaluons une solution de protection basée sur la technologie DPDK et Packet Journey pour assurer le routage.*

*Cet article présente les technologies utilisées et la plate-forme de routage en cours de mise en place sur le réseau de l'association, ainsi que les tests qui ont été réalisés pour valider cette approche.*

## Mots clefs

*Réseau, Routage, Déni de service distribué, Sécurité*

# 1 Introduction

## 1.1 Le réseau de Tetaneutral.net

Tetaneutral.net est une association loi 1901 qui milite pour la neutralité du réseau et pour cela a décidé de devenir opérateur enregistré auprès de l'ARCEP. L'association a été créée en 2011 et s'est développée jusqu'à arriver à près de 600 membres au moment de la rédaction de cet article. Tetaneutral.net propose à ses adhérents de l'hébergement (de machines physiques ou de machines virtuelles sur un cluster OpenStack + ceph qu'elle gère) et de l'accès internet via du matériel radio. L'ensemble des services proposés sont desservis par une interconnexion professionnelle à l'internet comprenant pour l'instant deux fournisseurs de transit (Cogent et Fullsave) et un peering (TouIX) (voir figure 1).

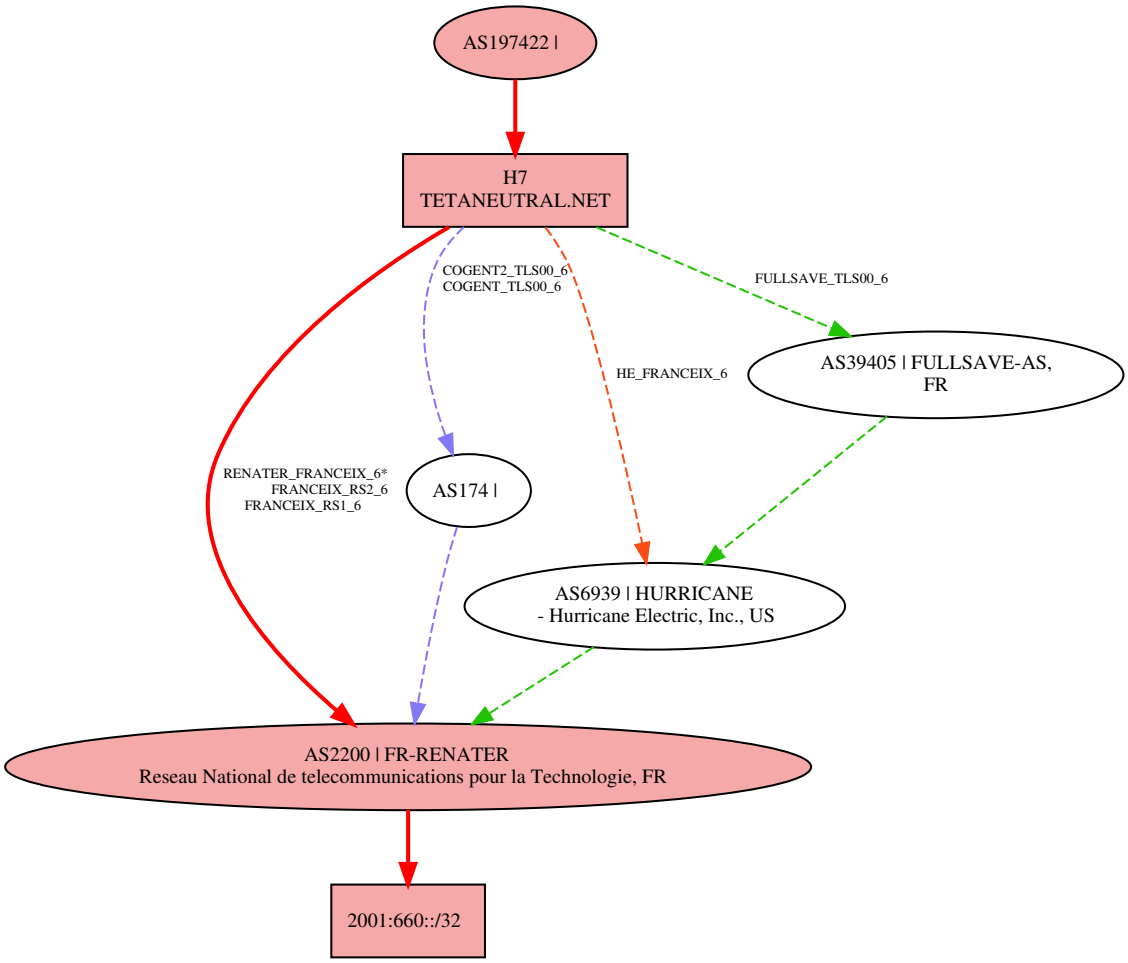


*Figure 1 - Exemple de routes depuis Tetaneutral.net*

Les liens vers le principal fournisseur de transit ont une capacité physique de $2 \times 10$ Gb/s, mais le tarif est sur une base de 1 Gb/s au maximum. Les attaques de déni de service distribué observés vers les machines des adhérents sont en majorité des attaques simples, utilisant des mécanismes d'amplification connus (serveurs DNS récursifs ouverts, serveurs NTP, etc) [1]. De plus elles sont généralement de relativement courte durée (5, 10, 15 minutes maximum) et n'affectent en général pas trop l'infrastructure réseau de l'association, même si elles arrivent à bloquer le serveur attaqué. Et elles sont absorbées dans le mécanisme de facturation des

opérateurs, qui ne compte que le 95 percentile.

Il apparaît donc possible de protéger les services hébergés par les machines des adhérents en mettant en place un mécanisme capable de filtrer les paquets identifiés comme appartenant à une attaque de type DDOS. Il faut cependant pour cela un routeur capable de traiter l'intégralité des paquets à la vitesse de l'interface réseau.

Le routeur actuellement en place ne fait pas du tout de filtrage, et cela contribue à sa capacité à traiter le trafic sans interruption. Par contre comme cela est détaillé ci-dessous (section 1.2), il n'est pas envisageable d'utiliser un mécanisme de filtrage classique (type IP-Tables ou Packet Filter) en restant capable de traiter 10 Gb/s.

## 1.2 Routage haute performances

Un équipement réseau (routeur ou commutateur) est composé de deux éléments principaux (appelés « plans » dans le vocabulaire des fabricants de ce type de matériels) : le *plan de contrôle* et le *plan de données*. Le plan de contrôle a en charge la gestion générale de l'équipement et de sa configuration. Le plan de données gère l'acheminement des paquets entre les différentes interfaces réseau, soit au niveau 2, soit au niveau 3. Il récupère du plan de contrôle un jeu de règles et la table de routage afin de déterminer comment traiter les paquets arrivant sur une interface. Le plan de données est implémenté dans les routeurs des constructeurs de matériels dédiés sur des circuits spécialisés (ASIC) pour obtenir les meilleurs temps de traitement possibles. Seuls les paquets nécessitant des traitements qui ne peuvent être réalisés sur l'ASIC principal sont traités par le processeur principal de l'équipement.

Lorsqu'on utilise un calculateur standard (un PC) avec plusieurs cartes réseau comme routeur, les deux plans sont réalisés par un ou plusieurs processeurs généraux. Dans l'approche la plus courante, un noyau Linux ou BSD est utilisé comme socle pour l'implémentation des deux plans. Un ou plusieurs démons dédiés (routage, serveur d'application web, serveur SSH) réalisent le plan de contrôle, tandis que le plan de données est réalisé via les fonctions du noyau dédiées à la réception du trafic réseau et à son routage, avec éventuellement l'ajout de fonctions de filtrage via IP-Tables ou pf par exemple.

Des routeurs utilisant cette approche sont largement utilisés de nos jours et remplissent leur rôle sur des réseaux de capacité standard jusqu'à 1 Gb/s [2, 3]. Cependant, lorsqu'on considère des réseaux plus rapides, cette approche arrive à ses limites [4, 5, 6]. En effet on ne peut plus compter sur l'augmentation exponentielle de la vitesse de traitement des processeurs pour absorber le trafic. Depuis quelques années l'augmentation de la puissance ne se fait presque plus via l'augmentation de la fréquence de l'horloge système, mais plutôt par l'ajout de cœurs de traitement supplémentaires. Une bonne partie de l'augmentation de l'efficacité de traitement des circuits est également utilisée plutôt pour diminuer leur consommation électrique.

Dans un réseau à 10 Gb/s, un paquet de 64 octets est transféré approximativement en 67 nanosecondes. En l'absence de traitement parallèle, le système doit donc pouvoir traiter un paquet en moins de 64 nanosecondes pour tenir la cadence maximale de 15 millions de paquets par seconde que l'on peut obtenir à 10 Gb/s.

| Taille du paquet | 64 octets | 128 octets |
|---|---|---|
| Paquets/s à 10 Gb/s | 14.88 M ($\times 2$) | 1.2 M ($\times 2$) |
| Temps inter-paquets | 67,2 $\mu s$ | 835 $\mu s$ |
| Cycles horloge à 2GHz | 135 | 1670 |
| Cycles horloge à 3GHz | 201 | 2505 |

Du point de vue du CPU, une lecture mémoire dans le cache de dernier niveau (L3) prend de l'ordre de 40 cycles d'horloge sur un processeur Xeon 5500. Une lecture en mémoire principale (cache miss) prend jusqu'à 200 cycles. On voit donc que le traitement des paquets à pleine vitesse ne peut se faire qu'avec du code extrêmement optimisé pour éviter tout accès à la mémoire principale, sous peine de ne pas suivre la cadence.

# 2 Data Plane Development Kit

Parmi toutes les approches existantes pour gérer cette contrainte temporelle très serrée [7, 8, 9, 10, 11], DPDK [12] est la plus aboutie et celle qui offre le plus de souplesse pour le traitement des paquets.

DPDK est un projet qui a démarré chez Intel en 2010, en coopération avec d'autres constructeurs tels que la société 6Wind, le projet est devenu un logiciel libre sous licence BSD en 2013. Depuis début 2017, la gouvernance du projet a été transmise à la Linux Foundation.

Par ailleurs, en permettant de s'abstraire des traitements effectués dans la pile réseau de la machine hôte, DPDK est également une technologie de base pour la virtualisation de fonctions réseau [13]. Cet aspect ne sera pas développé ici.

## 2.1 Principe de fonctionnement d'une application DPDK

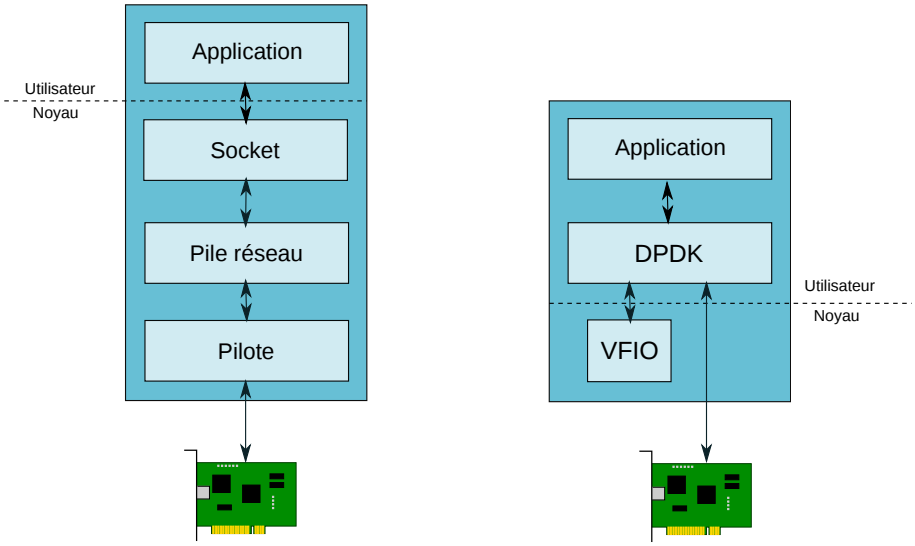


*Figure 2 - Architecture traditionnelle comparée à DPDK*

Le principe de DPDK est de s'abstraire au maximum des services du noyau pour prendre en charge directement en mode utilisateur les différents mécanismes nécessaires le plus efficacement possible. En particulier DPDK s'appuie sur un ensemble de *Polled Mode Drivers* (PMD), pilotes en mode utilisateur, sans interruption (c'est-à-dire avec attente active) qui supportent les cartes réseau les plus courantes dans le monde des serveurs. Ces pilotes sont rendus possibles par les interfaces *UIO* ou *VFIO* du noyau qui rendent accessible les registres de commande d'un périphérique depuis une bibliothèque utilisateur (Voir figure 2).

DPDK prend également directement en charge les cœurs du CPU où il s'exécute et ne fait pas du tout appel à l'ordonnanceur système. La configuration d'une application DPDK décrit quel(s) cœur(s) vont être réservés pour DPDK. Pour obtenir les meilleures performances en fonction des traitements à réaliser sur chaque paquet, un ou plusieurs cœurs peuvent être réservés pour chaque interface réseau.

Enfin la même philosophie est appliquée pour la gestion de la mémoire. DPDK utilise l'interface *huge pages* [14] pour gérer directement la mémoire en « volant » au gestionnaire mémoire standard du système une zone de mémoire physique contiguë dans laquelle il va gérer les buffers pour stocker les files d'attente des paquets en cours de traitement.

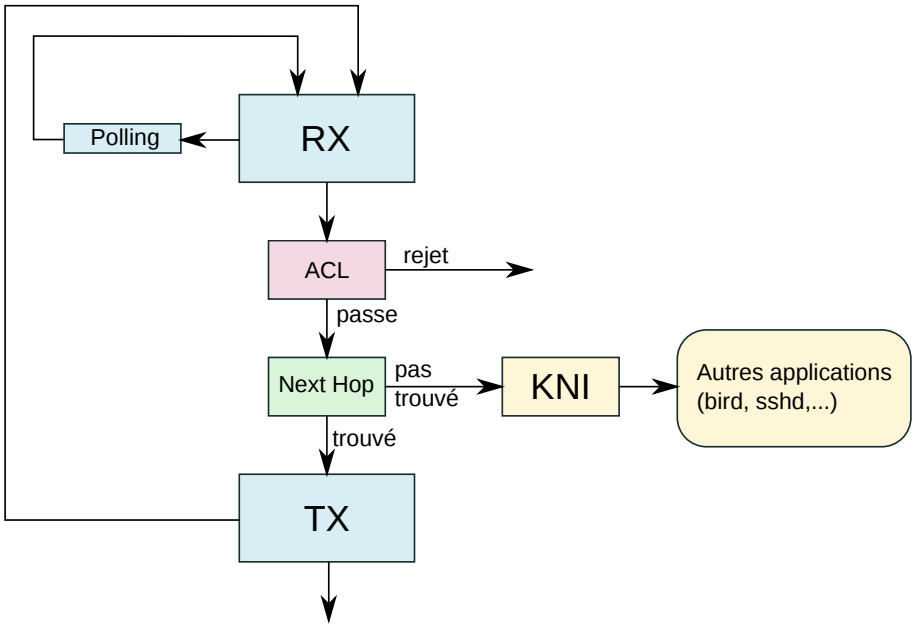


*Figure 3 - L'algorithme de traitement d'un paquet de Packet Journey*

Dans le cas d'architectures à mémoire non uniforme (NUMA), la configuration d'une application DPDK doit prendre en compte la topologie de la carte mère. Elle va allouer pour chaque interface réseau un cœur et une zone mémoire connectés directement à l'interface PCI Express correspondante.

## 2.2 Architecture d'une application DPDK

Une application DPDK est construite autour de trois composants principaux :

- EAL *Environment Abstraction Layer* la couche de portabilité entre les différents systèmes (supporte FreeBSD et Linux)
- PMD *Polled Mode Drivers* les pilotes de carte réseau en mode polling. DPDK supporte un certain nombre de cartes réseau 1 Gb/s, 10 Gb/s et 40 Gb/s de plusieurs constructeurs (Intel, Mellanox...), ainsi que des interfaces virtuelles pour des architectures de type Xen, KVM ou VMWare.
- Bibliothèques de support : l'environnement DPDK fourni des bibliothèques hautement optimisées pour réaliser des opérations de base dans une application réseau : classification des paquets, gestion de files d'attente, gestion de la mémoire

De nombreuses applications d'exemple sont fournies dans un but pédagogique pour montrer les bonnes pratiques et valider le fonctionnement sur une architecture donnée.

L'ensemble de ces composants permet de construire des applications qui répondent aux besoins de performance évoqués dans l'introduction.

# 3 Packet Journey

L'exemple d'application *l3fwd* fournie dans DPDK montre comment réaliser le plan de données d'un routeur en utilisant DPDK. Il manque cependant un plan de contrôle suffisamment générique pour en faire un vrai

routeur. La société Gandi a développé un logiciel libre qui comble ce besoin et l'a appelé *Packet Journey*.

Ce logiciel permet d'injecter dans les tables de décision de DPDK le contenu de la table de routage du noyau Linux (récupérée via l'interface Netlink, c'à-d le socket de routage). Il est donc possible de faire tourner un démon de routage tel que bgpd pour construire une table de routage et d'utiliser le résultat sur le plan de données DPDK. Une difficulté réside dans le fait qu'avec BGP les sessions sont établies via les mêmes interfaces réseau que celles au travers desquelles le trafic va être routé. Or DPDK prenant le contrôle de ces interfaces, il n'est plus possible pour le démon BGP de créer des sockets connectés aux interfaces de la machine pour établir ses sessions. DPDK fournit une solution via le mécanisme de KNI (Kernel Network Interface) qui permet de créer une interface virtuelle liée à une interface physique qui pourra être utilisée pour établir des connexions.

Le logiciel Packet Journey utilise un fichier de configuration pour établir les interfaces et configurer leurs adresses IP.

Pour obtenir les meilleures performances, il est recommandé d'allouer un cœur pour chaque thread de gestion des paquets arrivant sur une interface. L'algorithme exécuté dans chacun de ces threads est montré figure 3.

```
[pktj]
callback-setup  = /opt/packet-journey/up.sh
rule_ipv4       = /dev/null
rule_ipv6       = /dev/null
promiscuous = 1

; Port configuration
[port 0]
eal queues      = 0,2 ; queue,lcore
kni             = 1,0 ; lcore,kthread
[port 1]
eal queues      = 1,3 ; queue,lcore
kni             = 1,0 ; lcore,kthread
```

*Figure 4 - Exemple de fichier de configuration de Packet Journey*

```
#!/bin/bash
C=88
while [ $# -ge 2 ]; do
  link1=$1
  mac1=$2
  shift
  shift
  echo link $link1 mac $mac1
  ip link set $link1 address $mac1
  ip link set $link1 up
  ip addr add 192.168.$C.1/24 dev $link1
  ip -6 addr add 2a03:7220:1:${C}::1/64 dev $link1
  C=$((C+1))
done
```

*Figure 5 - Exemple de script de configuration des interfaces KNI*

La configuration présentée figures 4 et 5 utilise les deux cœurs 2 et 3 pour les threads qui vont traiter les

paquets en provenance des interfaces 0 et 1 respectivement. Le thread qui gère l'interface KNI pour traiter les paquets non routés est associé au cœur numéro 1.

# 4 Configuration de test

Afin de tester l'approche, Tetaneutral.net s'est équipé d'une nouvelle machine configurée ainsi (figure 6) :

- carte mère ASROCK EP2C612 WS [1]
- 2 processeurs Intel(R) Xeon(R) E5-2603 v3 @ 1.60GHz, 6 cœurs, 15 Mo cache, 40 lanes PCIe[2]
- 4 slots PCIe $\times$16
- 128GB RAM = 4$\times$32GB RAM DDR4 ECC
- Samsung SSD 850 EVO 500GB

Cette configuration est en-deça de celle utilisée pour qualifier les performances brutes de DPDK sur le matériel Intel [15], où il est montré que DPDK permet d'atteindre la vitesse théorique maximale d'une liaison à 10 Gb/s, y compris avec des paquets de 64 octets. En effet ces tests utilisent un processeur Xeon E5-2699v3 à 2,30 GHz, disposant de 45 Mo de cache.

## 4.1 Évaluation des performances brutes

DPDK fournit des outils pour tester les performances en termes d'acheminement de paquets. `pktgen`[3] est un outil de génération de paquets capable également de servir de récepteur pour l'analyse des performances.

L'infrastructure de test (figure 7) comprend trois machines en plus du routeur DPDK (`h8`) permettant de réaliser des tests dans plusieurs topologies.

Cependant, en raison de difficultés diverses, il n'a pas été possible dans les délais de rédaction de cet article de conduire des tests de performance significatifs, pouvant être présentés ici.

## 4.2 Routage avec Packet Journey

Le retard pris par l'évaluation des performances n'a pas permi de déployer une configuration de Packet Journey à grande échelle comme cela était initialement prévu. Seuls des tests des fonctionnalités de base ont pu être réalisés, montrant que le logiciel packet journey répond à nos attentes.

Il faut cependant noter que pour des performances optimales, les possibilités de filtrage offertes sont limitées à du filtrage sans état assez basique. Cela s'explique par la nécessité de faire tenir le code de traitement et de décision sur l'acheminement d'un paquet dans le cache du CPU qui l'exécute pour tenir dans les quelques centaines de cycles d'horloge disponible lorsqu'on travaille à 10 Gb/s.

[1] https://frama.link/C4ahPzHL
[2] https://ark.intel.com/fr/products/83349/Intel-Xeon-Processor-E5-2603-v3-15M-Cache-1_60-GHz
[3] http://pktgen-dpdk.readthedocs.io/en/latest/

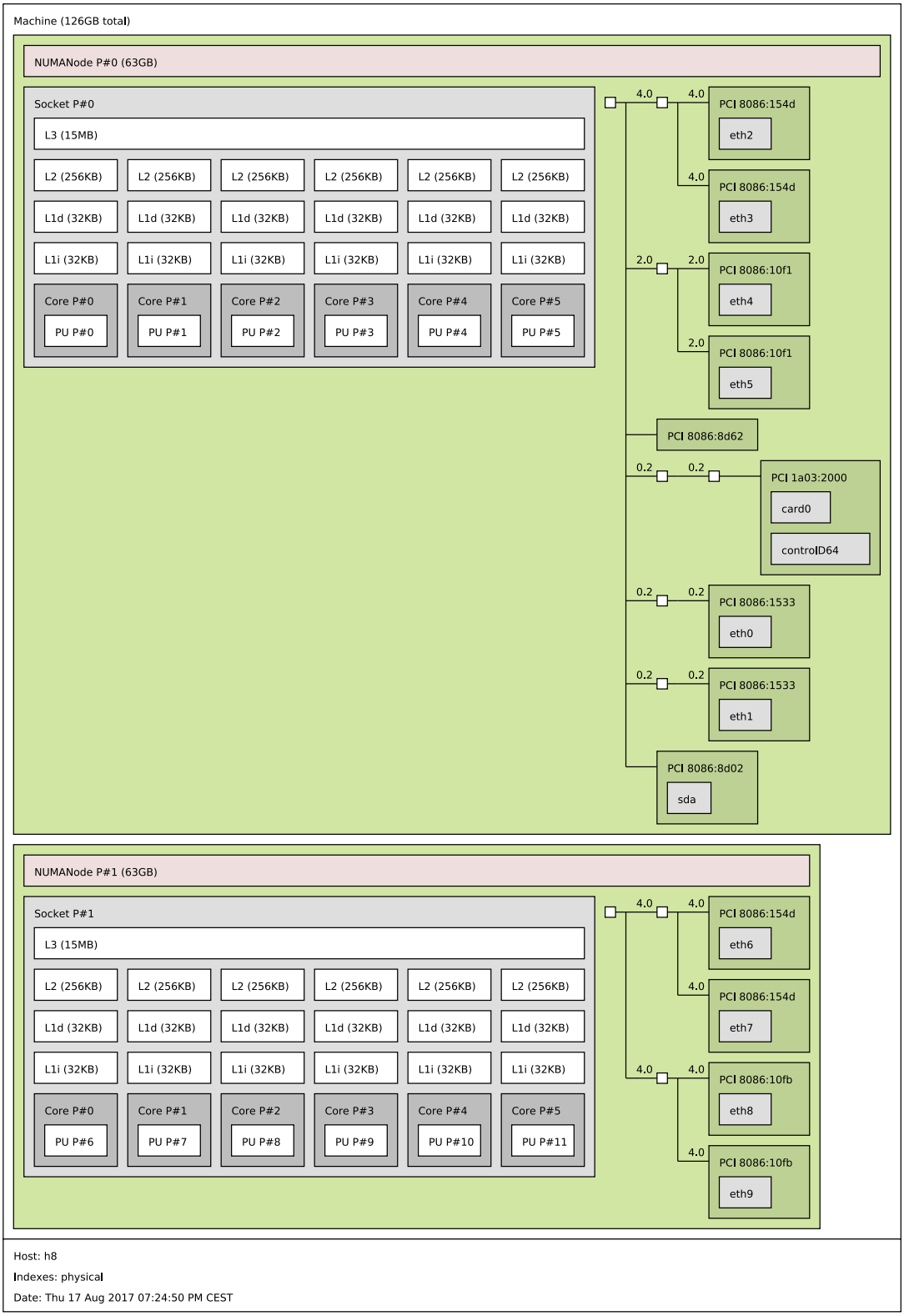


*Figure 6 - Architecture du routeur pour DPDK / Packet Journey*

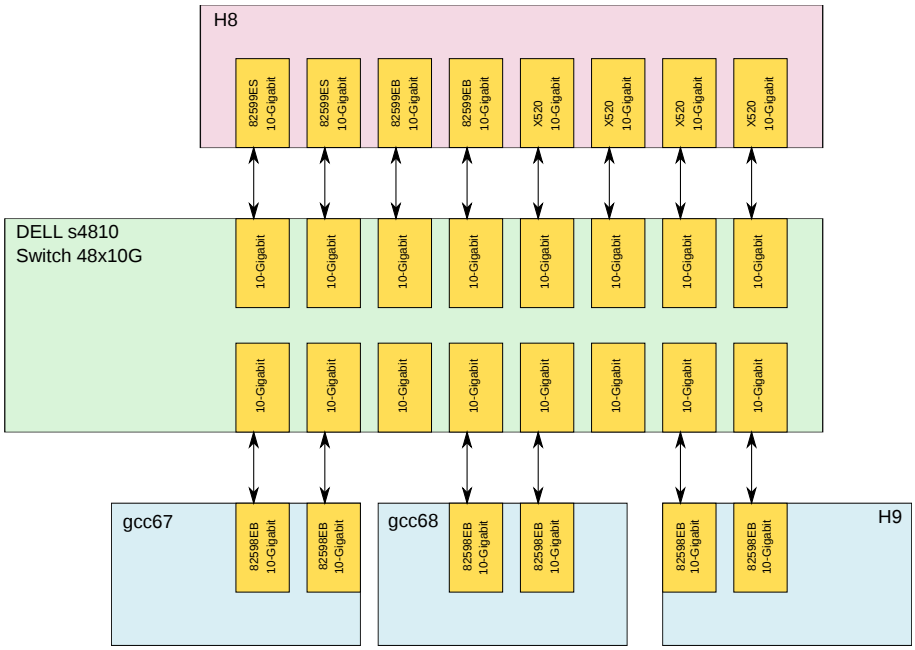


*Figure 7 - Infrastructure dpdk à Tetaneutral.net*

# 5 Conclusions et perspectives

Dans cet article il a été montré qu'une configuration de PC relativement standard permet, avec l'aide des outils libres DPDK et Packet Journey de réaliser un routeur hautes performances pour un coût raisonnable.

La fonctionnalité de base a été validée sur la configuration de test de l'association Tetaneutral.net et permet d'envisager de traiter les cas d'attaques de déni de service distribuées simples sans impact sur les serveurs de production. Cependant, il reste des tests de performance à effectuer, ainsi qu'un test de passage à l'échelle de packet-journey avant d'envisager un passage en production.

Par ailleurs, la technologie DPDK, indépendemment des gains de performance qu'elle apporte, permet aussi d'envisager des applications de type virtualisation de fonctions réseau ou *Software Defined Network* en s'abstrayant des contraintes de la pile réseau du noyau et en donnant aux applications du plan de contrôle un accès plus simple au contenu des paquets [13]. Il est probable que Tetaneutral.net utilise ces outils dans le futur pour gérer plus finement le trafic de ses adhérents.

# Bibliographie

[1] Korben. Les attaques DOS. Juin 2017. https://korben.info/attaques-ddos.html.

[2] M. Herrb. Outils de sécurité réseau dans un laboratoire avec OpenBSD et PF. Dans *JRES*, Toulouse, 2011. https://2011.jres.org/archives/130/index.htm.

[3] X. Laure, L. Catherine, T. Nodimar, F. Vivet et F. Elie. Migration vers l'open-source d'une infrastructure de pare-feu sur un campus CNRS. Dans *JRES*, Montpellier, 2013. https://2013.jres.org/archives/112/index.htm.

[4] P. Lamaiziere. Migration des pare-feu Internet de l'Université de Rennes 1 sous OpenBSD / PacketFilter. Retour d'expérience. Dans *JRES*, Montpellier, 2013. https://2013.jres.org/archives/31/index.htm.

[5] D. Salopek. Challenges in high speed packet processing. 2015. http://www.fer.unizg.hr/_download/repository/KDI,_Denis_Salopek.pdf.

[6] G.V. Neville-Neil et J. Thompson. Measure Twice, Code Once : Network Performance Analysis for FreeBSD. Dans *AsiaBSDCon 2015*, Février 2015. https://github.com/freebsd-net/netperf/blob/master/Documentation/Papers/ABSDCon2015Paper.pdf.

[7] L. Rizzo. netmap : A Novel Framework for Fast Packet I/O. Dans *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 101–112, Boston, MA, 2012. USENIX Association. https://www.usenix.org/conference/atc12/technical-sessions/presentation/rizzo.

[8] M. Majkovski. How to receive a million packets per second. Juin 2015. https://blog.cloudflare.com/how-to-receive-a-million-packets/.

[9] K. Yasukata, M. Honda, D. Santry et L. Eggert. Stackmap : Low-latency networking with the OS stack and dedicated nics. Dans *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 43–56, Denver, CO, 2016. USENIX Association. https://www.usenix.org/conference/atc16/technical-sessions/presentation/yasukata.

[10] FD.io - Vector Packet Processing. *White Paper*, Juillet 2017. https://fd.io/wp-content/uploads/sites/34/2017/07/FDioVPPwhitepaperJuly2017.pdf.

[11] J. Thompson. Can a BSD system replicate the performance of high-end router appliance ? *Reddit – /r/networking*, Août 2017. https://www.reddit.com/r/networking/comments/6upchy/can_a_bsd_system_replicate_the_performance_of/.

[12] R. Rosen. Network acceleration with DPDK. *Linux Weekly News*, Juillet 2017. https://lwn.net/Articles/725254/.

[13] Network Functions Virtualisation – Introductory White Paper. Dans *SDN and OpenFlow World Congress*, Darmstadt, Octobre 2012. https://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[14] M. Gorman. Huge pages. *Linux Weekly News*, Février 2010. https://lwn.net/Articles/374424/.

[15] DPDK Intel NIC Performance Report Release 17.05. Mai 2017. http://fast.dpdk.org/doc/perf/DPDK_17_05_Intel_NIC_performance_report.pdf.