Stimulateur synchrone de systèmes d'acquisition à plusieurs centaines de liens Gigabit Ethernet

Journée dispositifs expérimentaux IN2P3

Dirk HOFFMANN Julien HOULES

Centre de Physique des Particules de Marseille
IN2P3/CNRS

18 juin 2015

Contexte : CTA

Futur réseau de télescopes dédié à l'observation des rayonnements gamma.

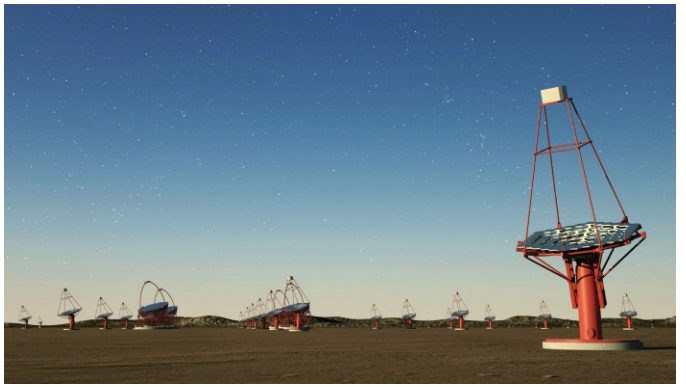
Deux sites : hémisphère nord et hémisphère sud

Une centaine de télescopes de trois types :

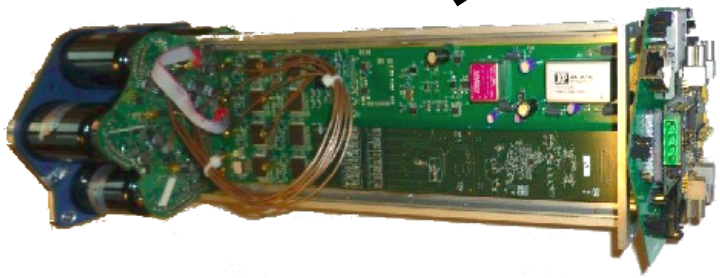
- Small Size Telescopes (4m)
- Middle Size Telescopes (12m)
- Large Size Telescopes (23 m)

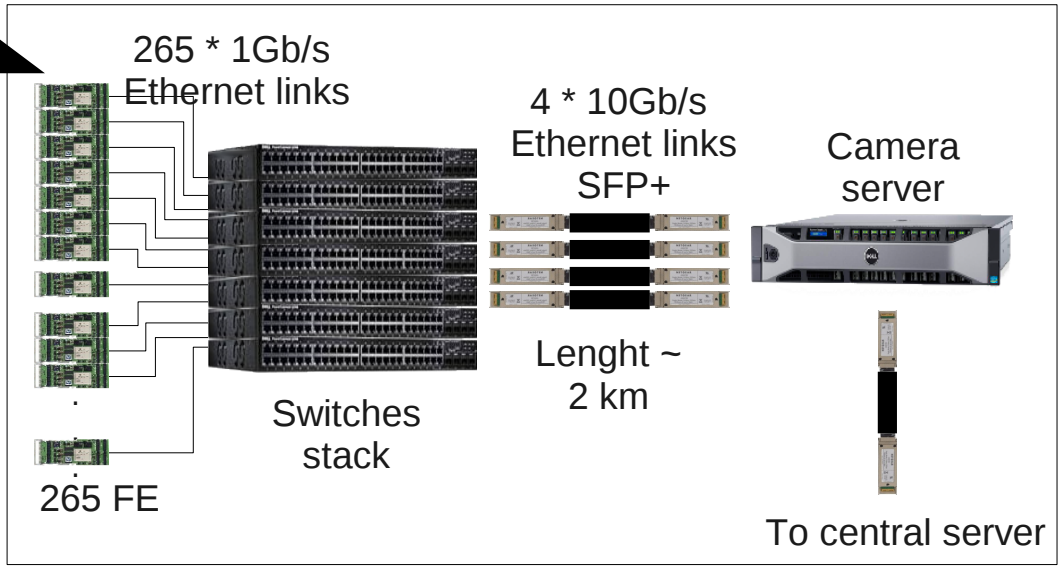
Au CPPM :
DAQ de la caméra NectarCAM qui équipera la moitié des MST -> une vingtaine d'exemplaires

Construction à partir de 2016



Contexte : NectarCAM





- 1855 photomultiplicateurs
- Clusters de 7 PM -> 265 FE
- 1 lien Gigabit Ethernet par cluster -> 265 liens Gigabit
- Fréquence moyenne trigger (distribution de Poisson) : 9kHz -> débit total 32 Gb/s, 120 Mb/s par FE
- Concentration dans un stack de 6 switches puis transfert vers le camera server via 4 liens 10 Gb
- Le camera server se charge de la réception, de l'event building, de la réduction des données en temps réel puis les envoie vers un serveur central

Contexte : NectarCAM

La **dispersion temporelle** de l'envoi des paquets d'un même évènement par les 265 FE est d'environ **6 ns**

Le **débit en entrée** des switches peut être bien **supérieur aux 40 Gb/s de la connexion en sortie** vers le camera server (distribution de Poisson)
Les switches doivent donc stocker les paquets pour absorber les pics de débit.

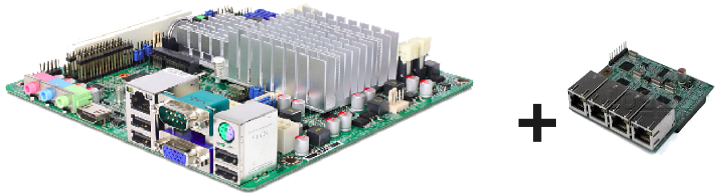
Nécessité de valider la chaine d'acquisition avec un stimulateur qui génère le flot de données de physique de manière reproductible. Caractéristiques :

- Débit de 32 Gb/s réparti sur 265 ports Gigabit Ethernet
- Simultanéité de l'envoi des paquets
- Cout minimal

Le stimulateur : principe

Solution sur FPGA trop onéreuse, peu flexible et manque de compétences d'électronicien

Solution adoptée : Single Board Computers (SBC)

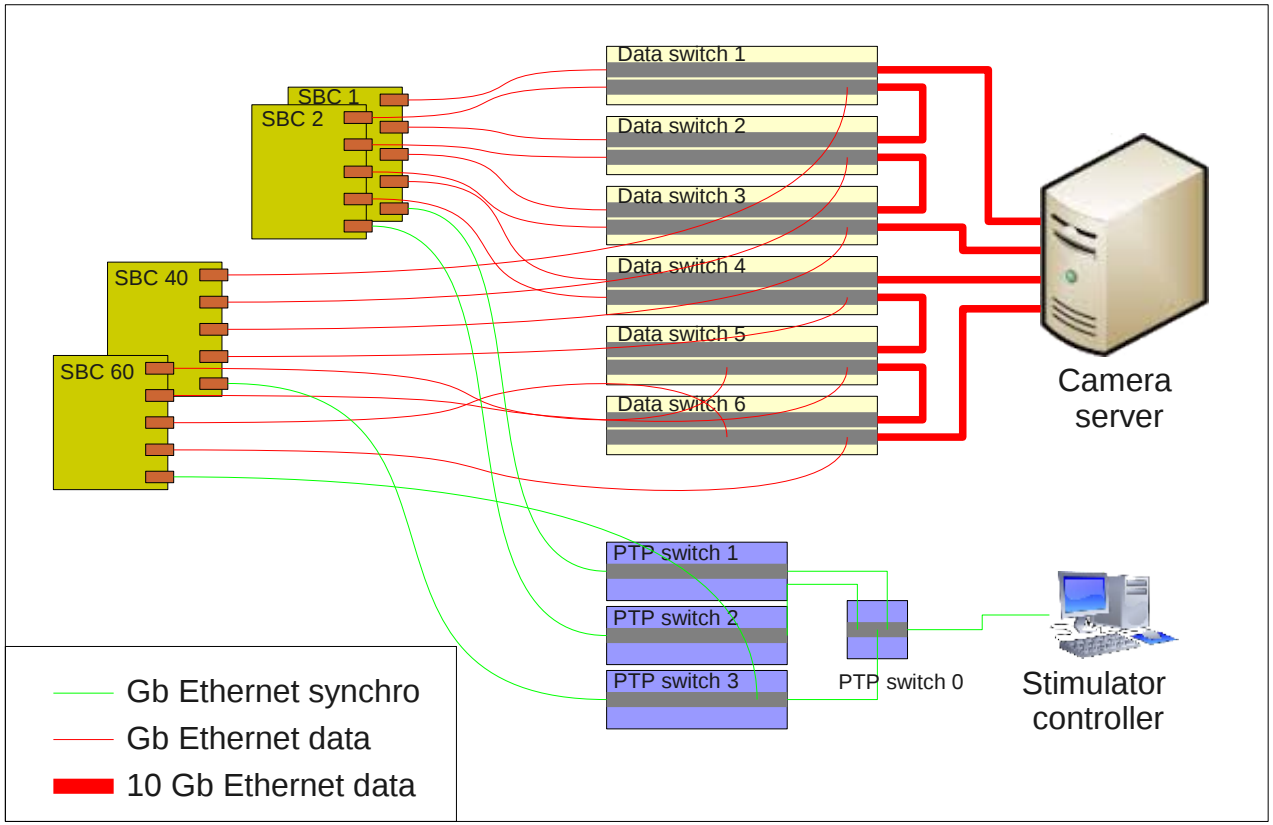


- Atom dual core
- 5 ports Gigabit Ethernet
- 4 Go RAM

Pas de système sur la carte : boot PXE d'un Linux customisé, le système est intégralement chargé en mémoire -> état initial maitrisé

Le stimulateur : principe

- L'horloge de chaque SBC est synchronisée par le Precision Time Protocol via un des ports Gigabit
- Les 4 autres ports servent à l'envoi des données. La couche de communication de Linux a été modifié pour améliorer la dispersion temporelle de l'envoi des paquets

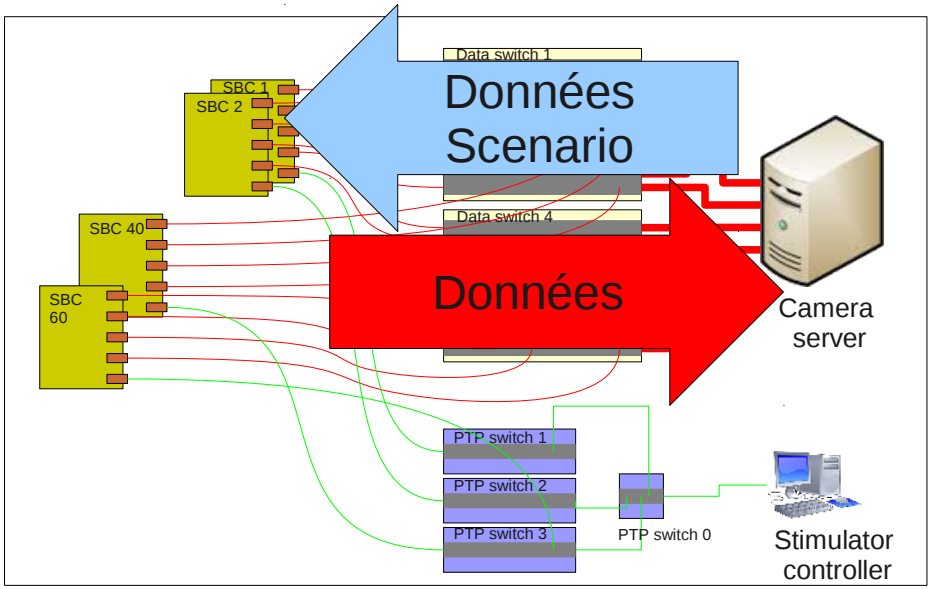


Le stimulateur : principe

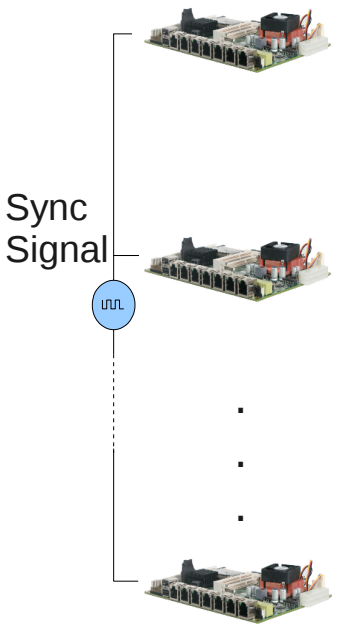
Un run de stimulation comporte plusieurs étapes :

- Téléchargement des données de simulation du camera server vers les SBC. Chaque port de chaque SBC reçoit un jeu de données différent
- Téléchargement d'un scenario temporel commun à tous les SBC, il contient le temps relatif des actions à effectuer (préparer paquets, envoyer paquet...)
- Le stimulator controller envoie l'heure de démarrage du scenario à tous les SBC. La stimulation démarre. Les horloges étant synchronisées, les cartes envoient les paquets au même moment.

t(s)	*t(ns)*	*action*	*nb_packets*
0	*0*	*SCENARIO_PREPARE_DESCRIPTORS*	*1*
0	*100000*	*SCENARIO_SEND_PACKETS*	*1*
0	*190000*	*SCENARIO_PREPARE_DESCRIPTORS*	*1*
0	*200000*	*SCENARIO_SEND_PACKETS*	*1*
0	*290000*	*SCENARIO_PREPARE_DESCRIPTORS*	*1*
0	*300000*	*SCENARIO_SEND_PACKETS*	*1*
0	*390000*	*SCENARIO_PREPARE_DESCRIPTORS*	*1*
0	*400000*	*SCENARIO_SEND_PACKETS*	*1*
1	*0*	*SCENARIO_PREPARE_DESCRIPTORS*	*10*
1	*100000*	*SCENARIO_SEND_PACKETS*	*5*
1	*200000*	*SCENARIO_SEND_PACKETS*	*5*

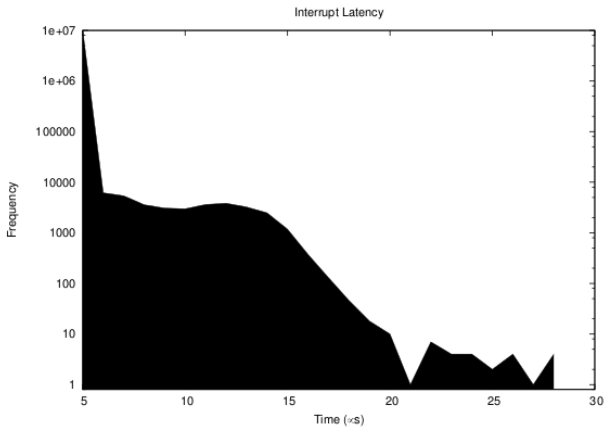


Synchro inter-cartes

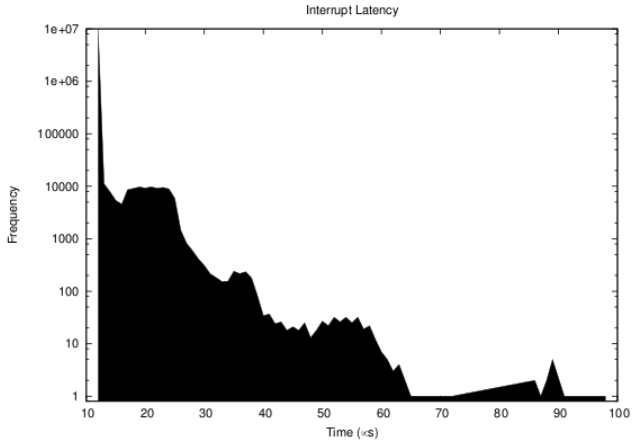


Première solution envisagée : synchronisation de l'envoi des paquets via un signal périodique ou aléatoire injecté sur le port GPIO ou parallèle des cartes. Un paquet est envoyé à chaque pulse.

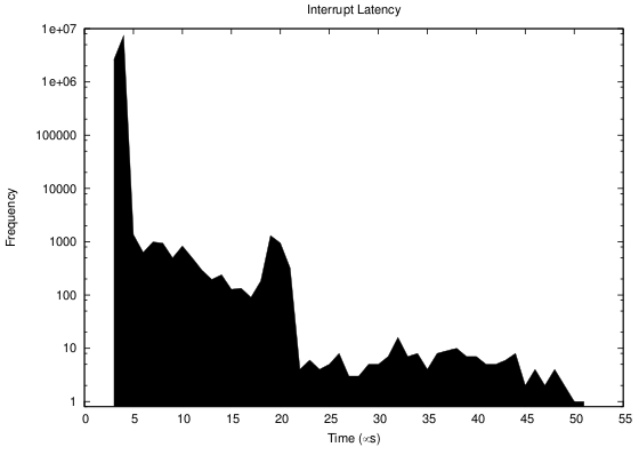
Problème : la latence d'interruption (temps entre le signal sur la pin et l'exécution effective de la routine d'interruption) introduit un jitter de plusieurs dizaines de µs, trop important pour nos besoins. Idem pour la latence d'ordonnancement.



Xenomai Idle



Linux PREEMPT_RT Idle



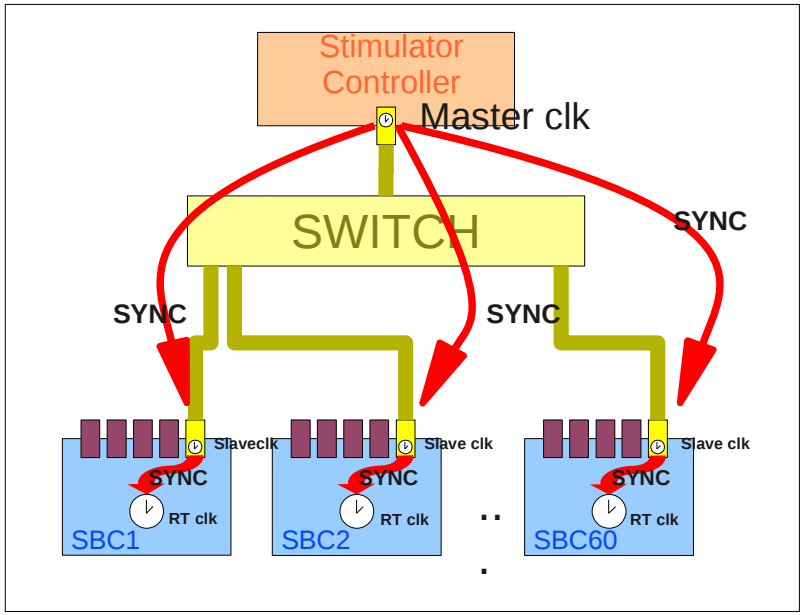
Mainline Linux Idle

Performance Evaluation of GNU/Linux for Real-Time Applications. Knutsson, Tobias. Uppsala University

Synchro inter-cartes

La solution adoptée : synchronisation des horloges des SBC via Precision Time Protocol et envoi de paquets au moment indiqué dans un scénario commun à toutes les cartes.

Horloge PTP matérielle embarquée dans les chips réseau des cartes SBC. 1 port par carte dédié à la synchronisation.

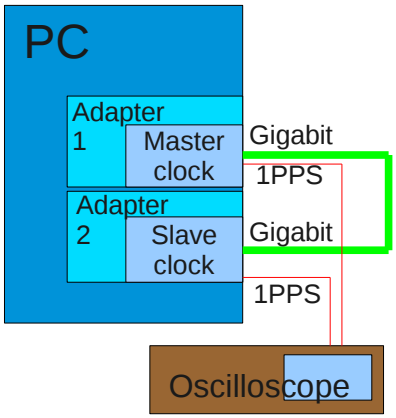


Chaque Slave Clock se synchronise sur la Master Clock qui se situe dans le stimulator controller (syntonisation). Ensuite l'horloge RT de chaque SBC se synchronise sur sa Slave Clock.

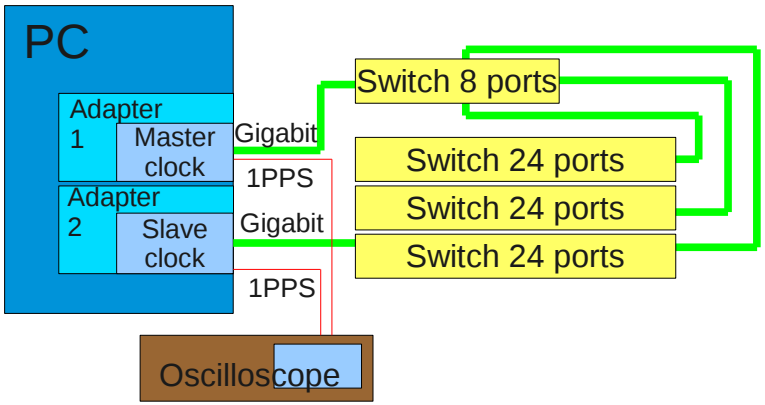
Synchro inter-cartes

Evaluation du Precision Time Protocol

Modification de cartes réseau Intel I210 pour sortir un signal 1PPS puis mesure du déphasage entre le signal du Master et du Slave avec les deux montages suivants :



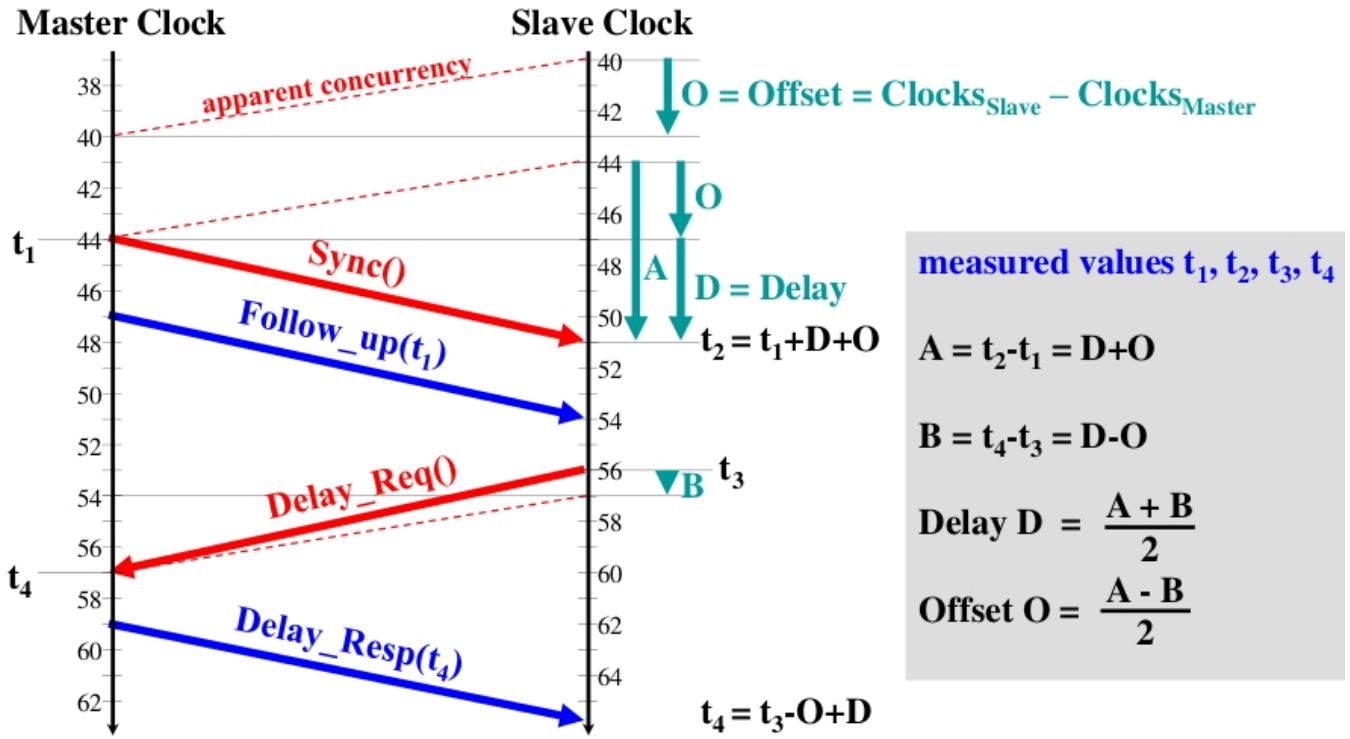
En point à point, les deux horloges sont synchronisées à une dizaine de nanosecondes



Avec l'introduction d'une cascade de switches, la synchronisation est de l'ordre d'une centaine de nanosecondes

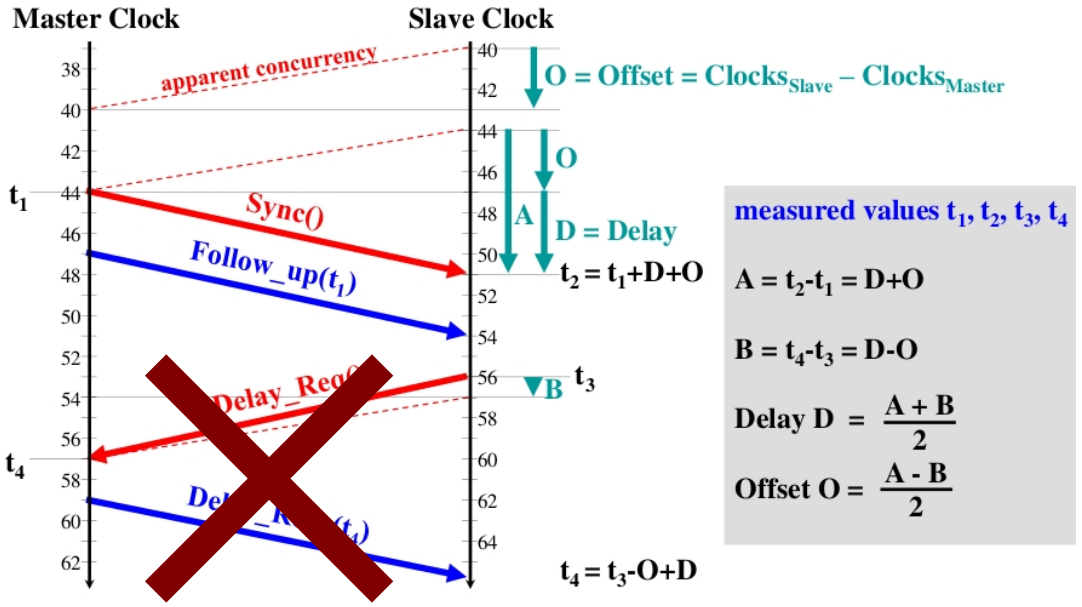
Synchro inter-cartes

Modification du Precision Time Protocol pour pouvoir utiliser des switches ordinaires : Suppression du message DELAY_REQ pour éviter l'encombrement dans le switch vers la master clock (et des mesures fausses).



Synchro inter-cartes

Modification du Precision Time Protocol pour pouvoir utiliser des switches ordinaires : Suppression du message DELAY_REQ pour éviter l'encombrement dans le switch vers la master clock (et des mesures fausses). Mesure initiale propagée à tous les esclaves.



La synchronisation mesurée entre l'horloge maitre et une horloge esclave parmi 48 est un peu moins bonne, de l'ordre de 200 ns.

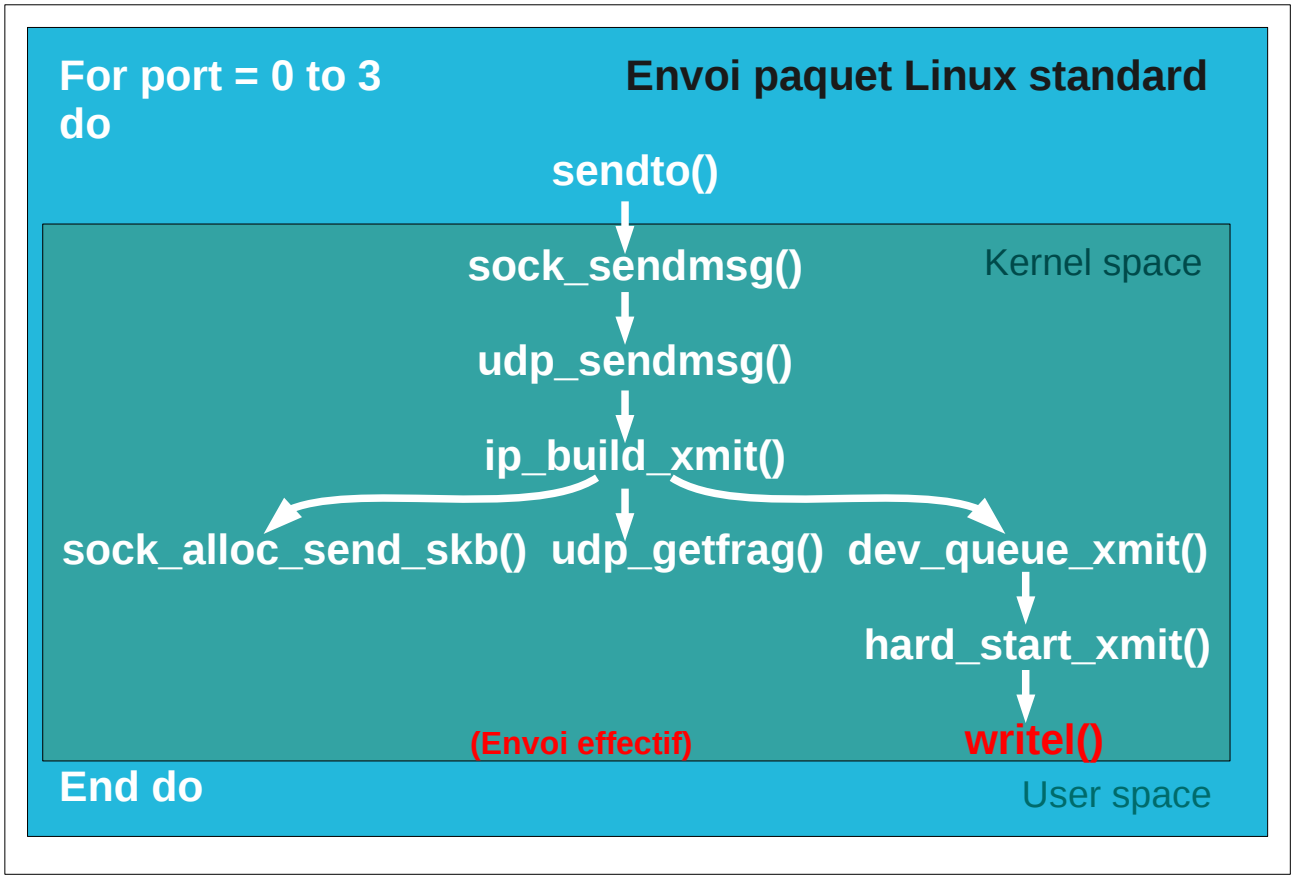
Optimisation réseau

Premiers essais avec la couche réseau standard de Linux mauvais : les paquets d'un même évènement sont étalés sur plusieurs dizaines de µs entre les cinq ports d'une même carte avec un jitter important.

Raisons principales :

- Allocation dynamique des buffers de paquets
- Copie inutile des paquets depuis l'espace kernel vers l'espace user
- Grand nombre d'appels à des fonctions qui nous sont inutiles (routage, construction headers...)

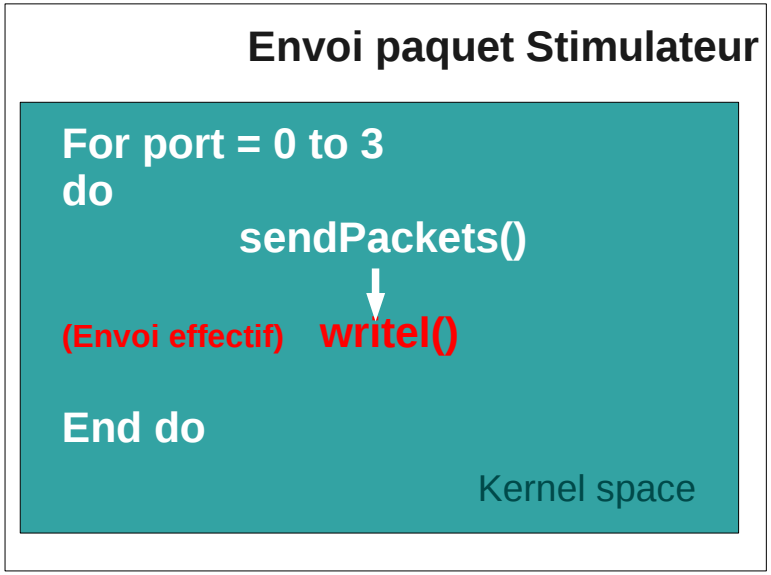
Ces défauts s'additionnent car même arbre d'appel de fonctions parcouru pour chaque port



Optimisation réseau

Solution : by-pass de la couche réseau de Linux

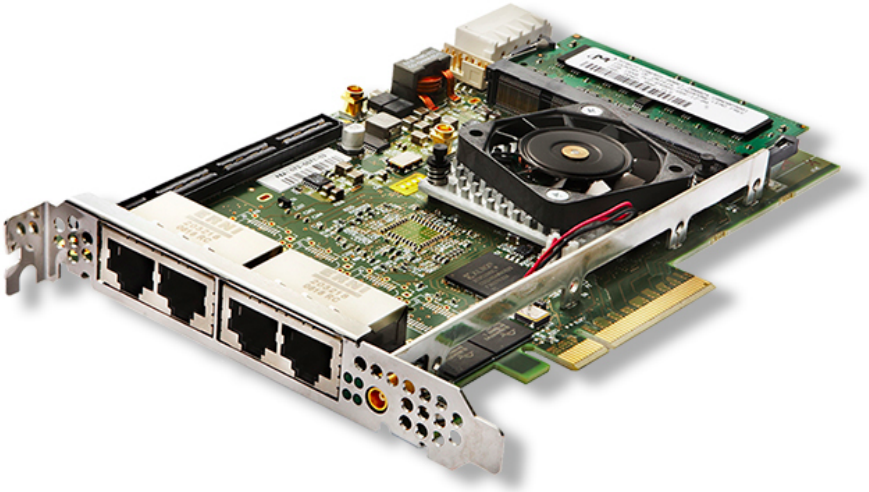
- Le stimulateur est un module qui tourne dans l'espace Kernel
- Allocation statique des buffers de paquets (zones DMA-able), remplis avant la stimulation avec les données à envoyer
- Modification du driver réseau pour accès direct aux registres d'initiation du transfert DMA
- La sérialisation est limitée à l'écriture dans ces registres du rang du buffer à envoyer
- Polling dans une tâche à priorité maximale sur l'horloge temps réel jusqu'au temps de l'action suivante dans le scénario
- Inhibition des interruptions



Caractérisation

Moyens de caractérisation

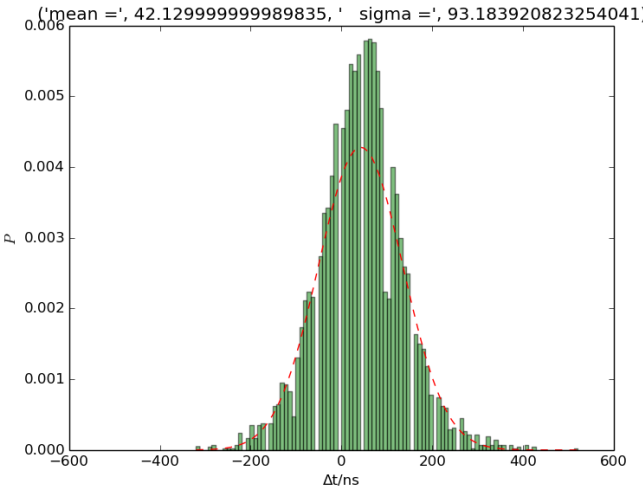
Napatech NT4E-4T



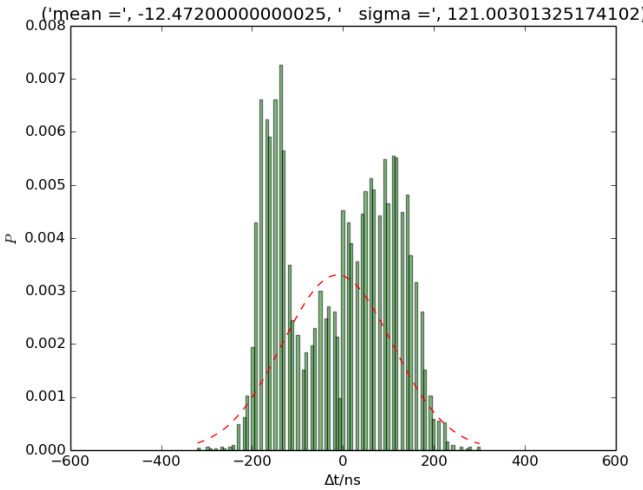
- 4 ports Gigabit Ethernet
- Précision datation paquets : 7 ns

Caractérisation

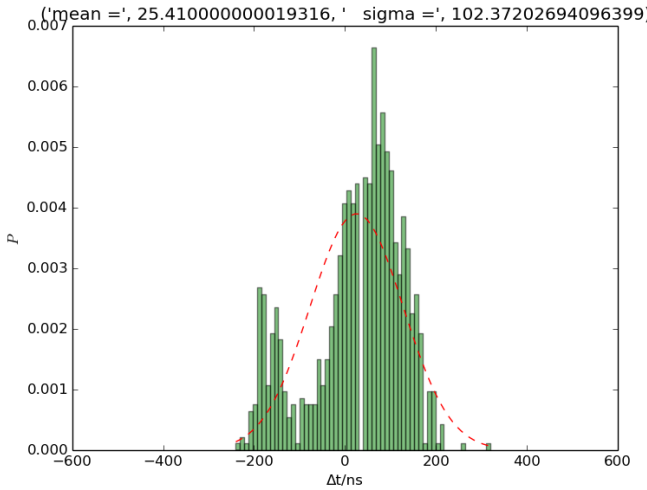
Résultats préliminaires



Étalement temporel pour l'envoi de paquets de 1024 octets à 50 kHz depuis 1 port par carte sur 2 cartes parmi 40



Étalement temporel pour l'envoi de paquets de 1024 octets à 50 kHz depuis 2 ports par carte sur 2 cartes parmi 40



Étalement temporel pour l'envoi de paquets de 1024 octets à 50 kHz depuis 3 ports (pour les deux premiers ports) par carte sur 2 cartes parmi 40

- Tous les paquets d'un même évènement envoyés en quelques centaines de ns
- Synchronicité meilleure entre les ports d'un même indice sur les différences cartes

Nouvelles mesures avec plus d'échantillons à venir

Ouverture

- Logiciel assez léger : les modifications sont aisées
- Le contenu des paquets est totalement libre
- La mémoire disponible est de 500 Mo/port
- Les patterns sont totalement libres dans le scénario
- Possibilité de synchroniser via signal externe si faible dispersion moins cruciale, dans ce cas 320 ports Gigabit disponibles
- Générateur de pulse depuis scénario à l'étude
- Adaptation au TCP/IP assez simple en utilisant des sockets standard mais perte de précision temporelle

En bref

Souplesse :
peut être utilisé dans d'autres contextes

256 ports Gigabit, extensible
simplement en ajoutant des cartes SBC

2 Go/port
disponibles pour le stockage des données de stimulation

Faible coût :
50 euros/port

Synchronicité inférieure à la µs
(quelques centaines de ns)

Programmation en C, base Scientific Linux 6