**Distribution des paquets entrants par le NIC :**

* RSS : Receive Side Scaling

Distribution des paquets parmi les multiples CPU. Sans RSS, distribution à un seul CPU.

RSS n’utilise pas les processeurs hyper-threaded vu que ces processeurs utilisent le même étage exec.

RSS s’assure qu’un flow TCP reste sur le même processeur, via un hash de certaines parties du flow (load-balancing du flow par rapport au résultat du hash).

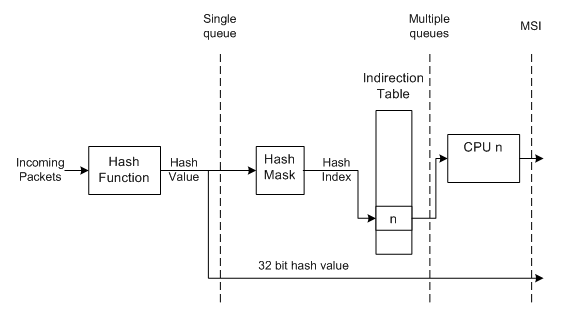
Ce hash porte sur n’importe quelle partie ou groupement de parties de la trame reçue.

On s’assure donc qu’aucun CPU n’est 100 % actif pendant qu’un autre est idle.

RSS préserve l’ordre d’arrivée des paquets d’un flow.

Marche en entrée et sorties des paquets

Hash sécurisé : signature



3 modes RSS :

* + Calcul du Hash avec une queue unique : calcul du hash puis on met la trame reçue dans une queue d’un CPU.
  + Calcul du Hash avec multiples queues : Idem sauf que les paquets entrants sont dans plusieurs queues.
  + Message Signaled Interrupts (MSI) : IRQ sur le CPU en question

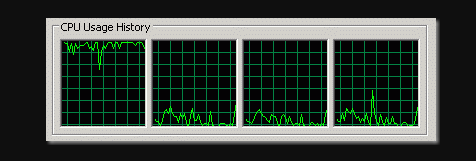
**Possibilité d’utiliser IRQ-balance en conjonction avec RSS.**

Bien se renseigner sir RSS est symétrique : un flow entrant n’aura pas le même hash qu’un flow sortant.

Problème, les paquets peuvent ne pas arriver dans le bon ordre au niveau de Suricata, donc risque de mauvaise interprétation.

* Non-RSS :

IRQ sur un seul CPU puis distribution sur plusieurs CPU sans tenir compte du hash, donc non optimisé.



Vers les 1 Gbps pas besoin de RSS mais à 10, ça devient obligatoire, un CPU ne pouvant traiter la charge réseau.

PF\_RING :

* 1 queue RSS par CPU
* cluster type : cluster\_flow
* Désactiver offloading sauf rx/tx csum
* Offloading :

Le kernel délègue certaines tâches au NIC : calcul de checksum, connexions…