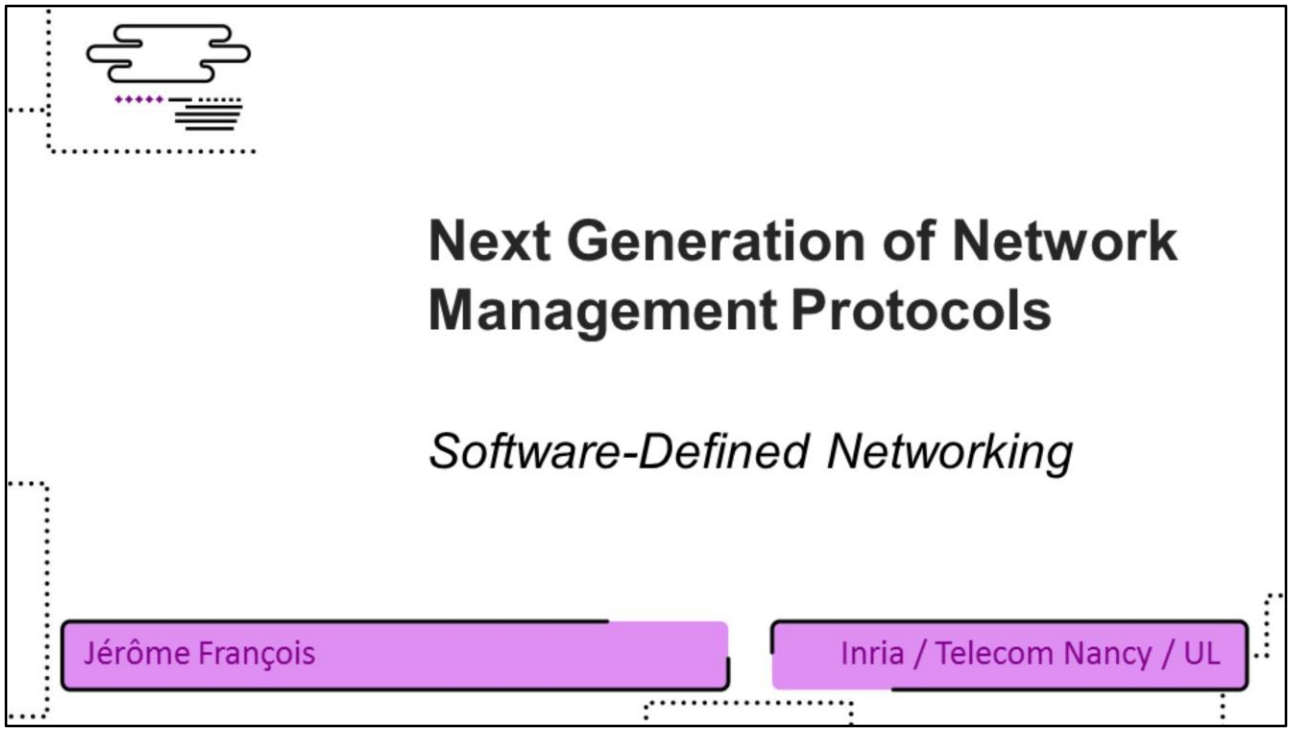
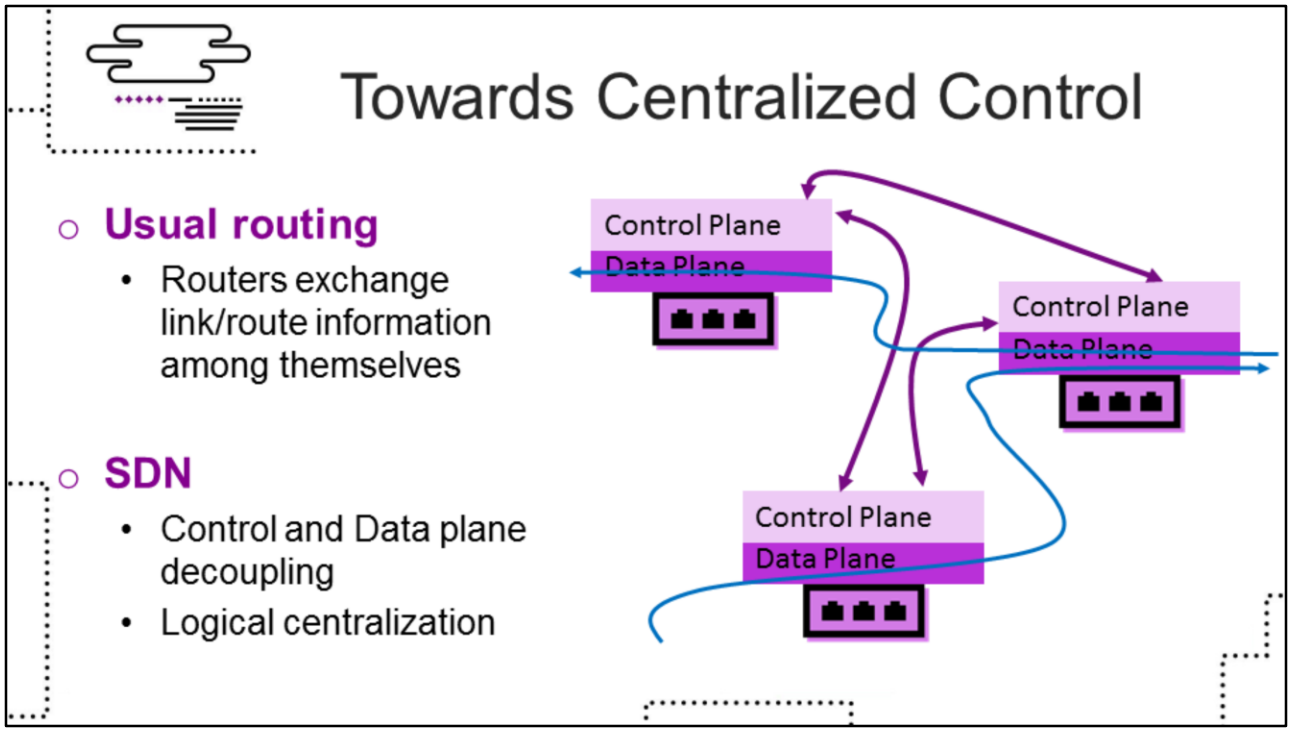
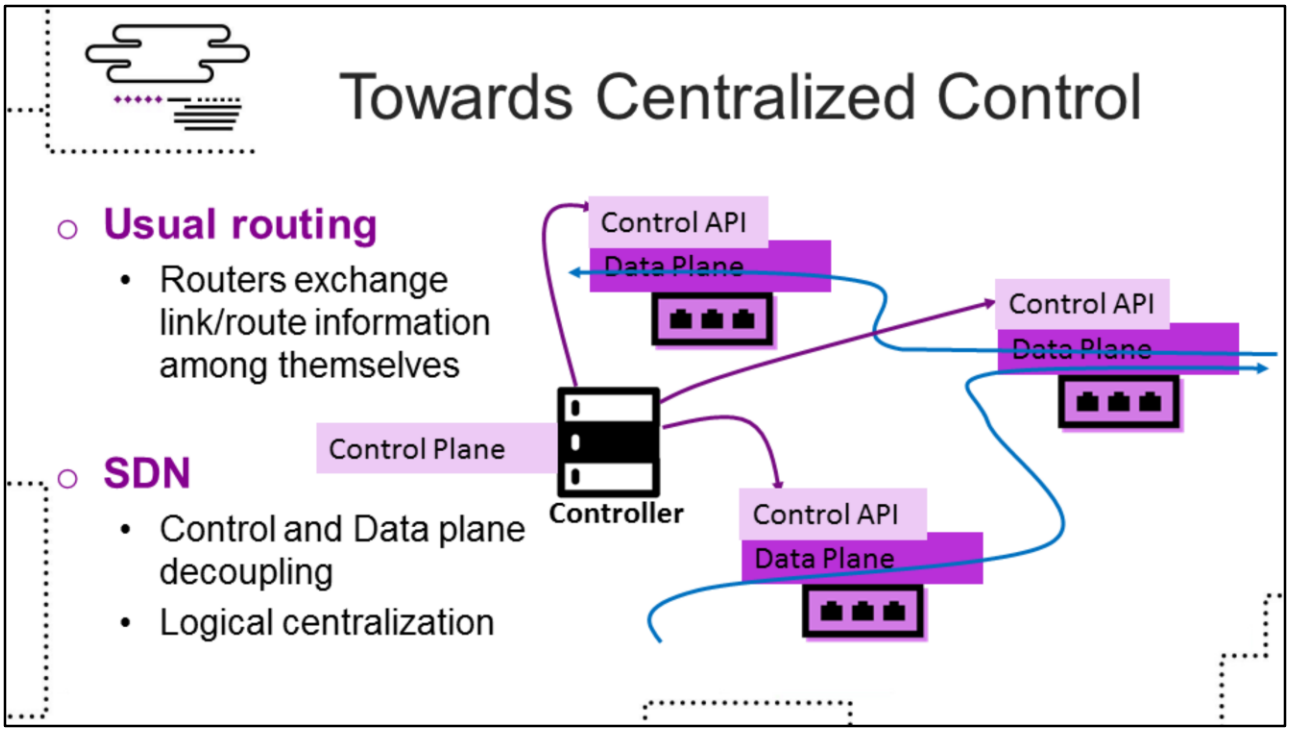


Bonjour à tous, ce cours est une introduction au Software Defined Networking ou réseau logiciel que l'on désigne habituellement par l'acronyme SDN.



Pour rapidement en comprendre son principe, prenons un petit exemple.
Sur un réseau traditionnel, les routeurs s'échangent des information de routes ou de liens et chacun d'entre eux est ensuite responsable de déterminer sa table de routage en fonction. C'est ici le propre des algorithmes de routage qui doivent être implémentés sur les routeurs. Ce sont donc des algorithmes distribués. Ainsi, le trafic peut être routé dans le réseau et les paquets de données transmis

Le plan de contrôle et de données sont donc colocalisés au niveau de chaque routeur.



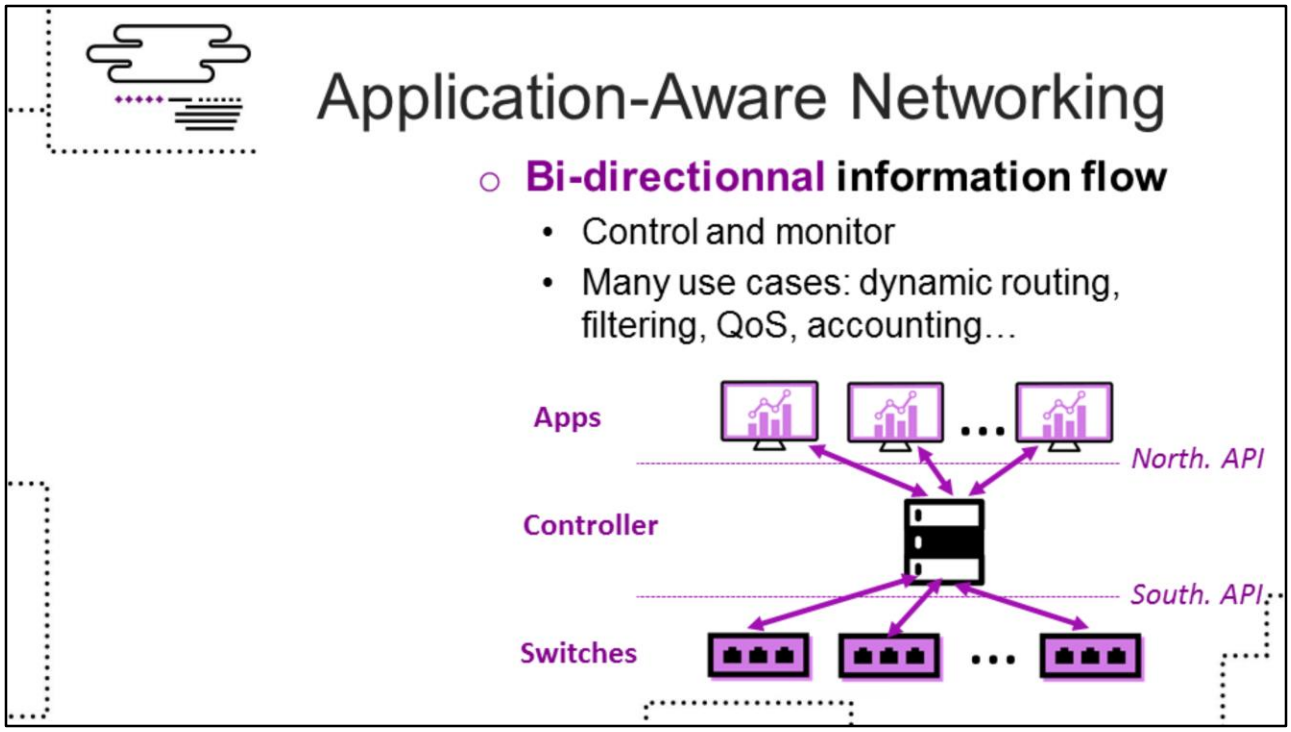
Dans le cas de SDN, ceux-ci sont découplés et les routeurs deviennent alors de simples élément destinés à transmettre les paquets. Dans la terminologie SDN, on ne parle plus de routeurs mais simplement de forwarding devices ou switch. On retrouve alors le plan de contrôle sur un contrôleur logiquement centralisé. Celui-ci est maintenant responsable d'exécuter un algorithme de routage et de déterminer les tables de routage de chacun des switchs, mais qui vont bel et bien jouer le rôle de routeur. Un fois calculé le contrôleur va donc transmettre les tables de routage par le biais d'une interface. Les routeurs peuvent ensuite faire transiter le trafic comme précédemment

Naturellement, on se pose la question de l'intérêt d'une telle approche. Le but principal du paradigme SDN est de déplacer le plan de contrôle sur une machine dites classique, type x86, avec des capacités de traitement supérieures aux équipements réseaux et avec des prix beaucoup plus compétitifs. En effet, les équipements réseaux sont en principe dédiés avec du matériel spécifique. On obtient aussi plus de flexibilité car un contrôleur peut servir à la fois de routeur, de firewall ou de load-balancer par exemple. C'est aussi un moyen de rapidement tester un nouvel algorithme sans modifier tous les équipements réseaux.

Le contrôleur présente aussi l'avantage d'avoir une vue globale du réseau et est donc en position de prendre les meilleurs décisions. On peut notamment éviter facilement les boucles de routage tout en utilisant un maximum de lien, ce qui n'est pas le cas

avec un algorithme classique comme SPT (shortest path tree) qui exclut certains liens physique de la topologie logique.

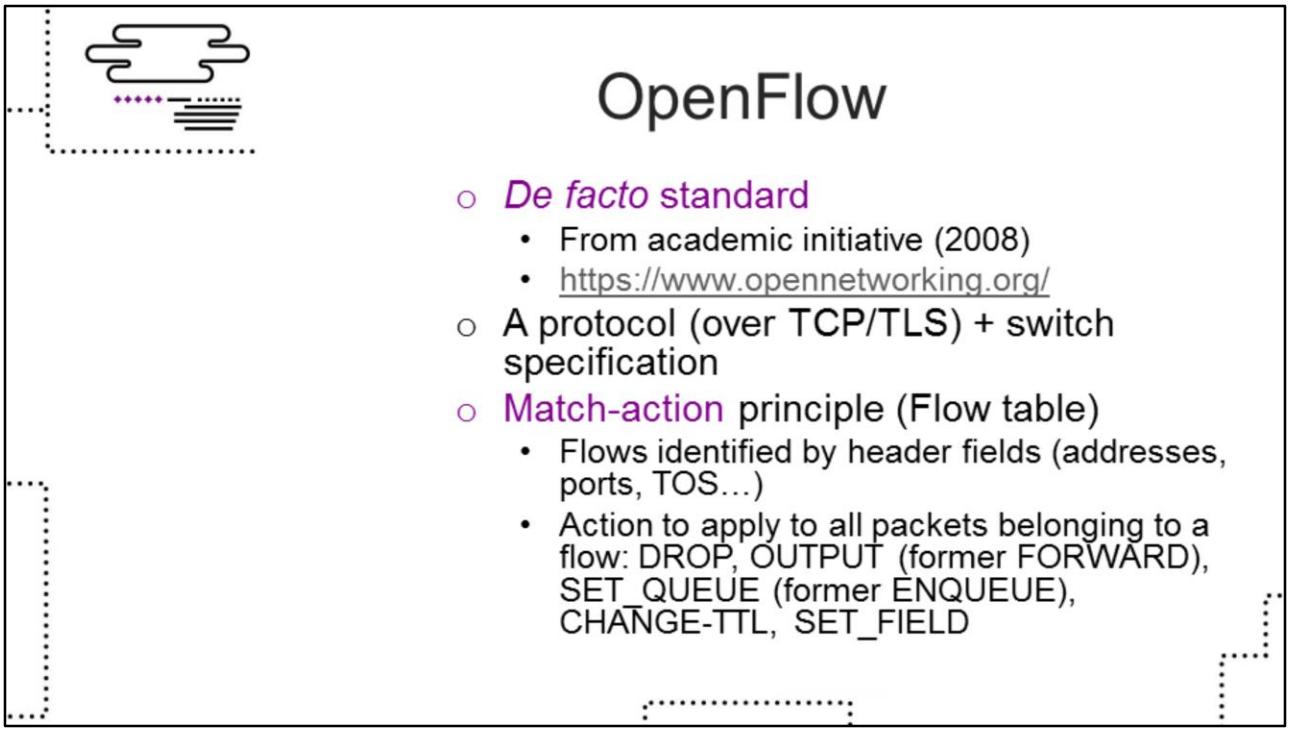
Notez cependant que lorsque je parle de capacités de traitement supérieures au niveau du contrôleur, je fais état ici des capacités de calculs. Un équipement réseau type routeur, a peut être moins de capacité sur ce plan mais est beaucoup plus performant à décoder, traiter et à transférer des paquets grâce à du hardware spécifique.



Avec SDN, deux interfaces principales au niveau du contrôleur sont définies.
L'interface Sud qui permet d'interagir avec les switchs.
L'interface Nord permet quant à elle d'interagir avec les applications. On a donc indirectement un fort couplage entre le réseau et les applications. Les applications peuvent donc à la fois avoir une vue de l'état de l'infrastructure réseau sur lesquelles elles s'exécutent et s'adapter pour améliorer leurs performances.

A l'inverse, les applications peuvent interagir avec le réseau pour qu'il s'adapte. Bien entendu c'est une vue très réductrice et il est clair qu'on ne souhaite pas que n'importe quelle application puisse modifier la configuration du réseau. Cela s'effectue dans un environnement très contrôlé, par exemple dans un datacenter où l'opérateur définit quelques politiques précises pour certaines applications qui lui sont propres.

Souvent, on introduit une couche supplémentaire qui est le plan de gestion se retrouvant donc au dessus du plan de contrôle. Cela sert alors d'interface, par exemple pour prendre en compte les différents critères de qualité de service requis par les applications ou les politiques d'accès, et en déduire les opérations à réaliser sur le réseau par le biais du contrôleur.

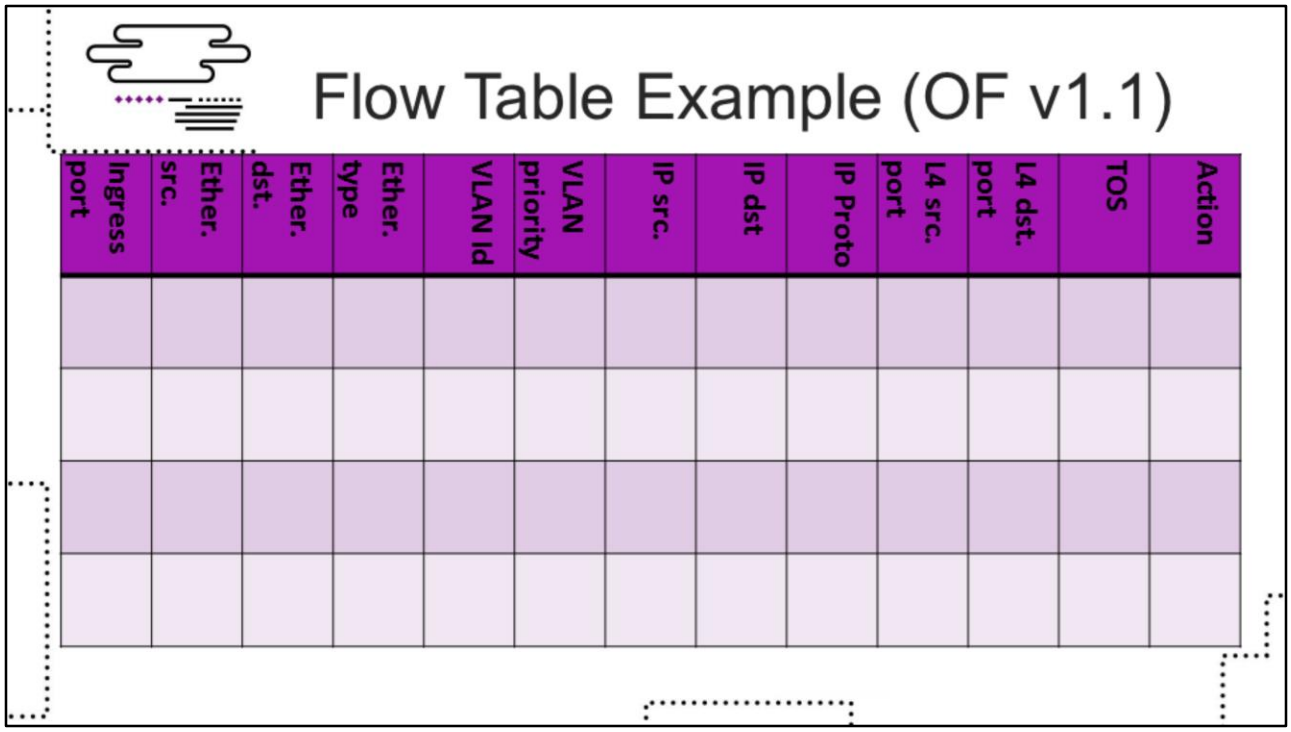


OpenFlow est le standard de fait en SDN. Il est soutenu par l'Open Networking Foundation qui en publie les spécifications. OpenFlow définit à la fois un protocole d'échange pour la partie contrôle ainsi que le mécanisme de match-action au niveau des switch. Notez que la sécurité et la fiabilité du protocole est garantit par l'utilisation de TCP et TLS.

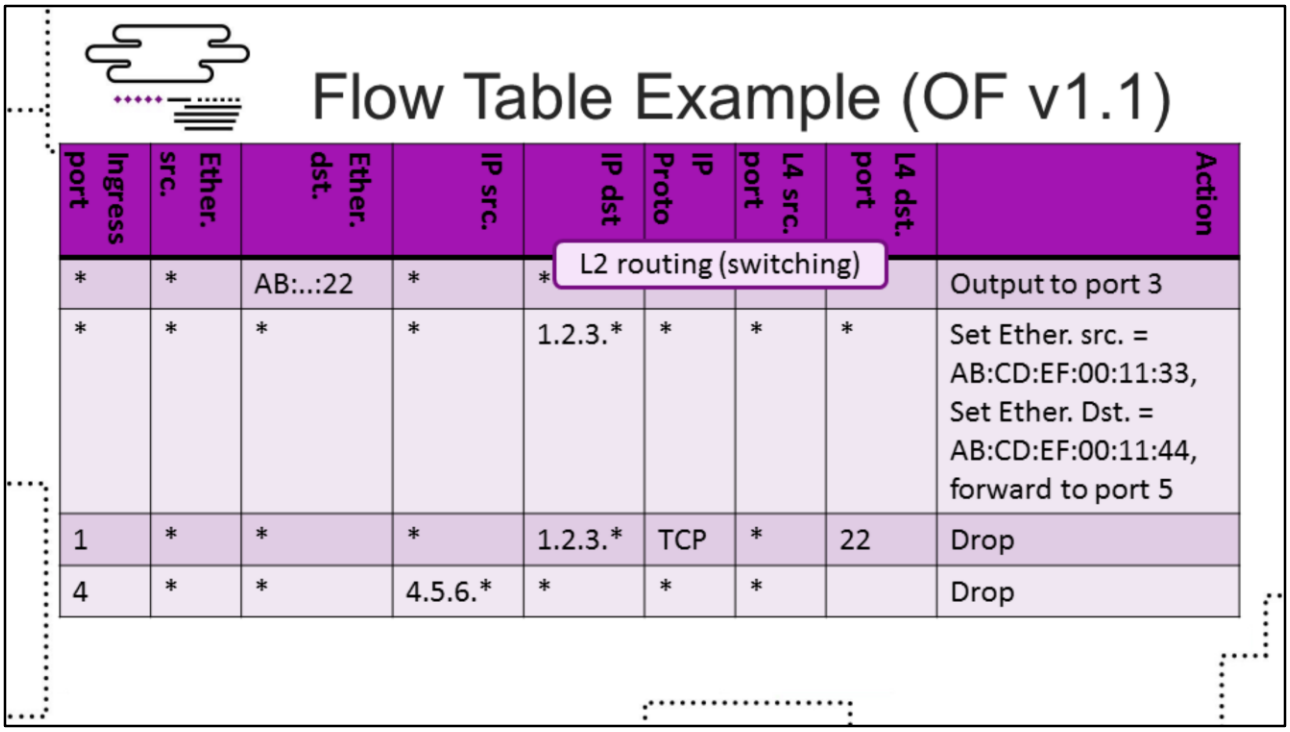
Le principe de match-action se base sur une table de flux ou flow table. Celle-ci est mise à jour par le contrôleur qui définit pour chaque entrée un critère de correspondance ou match et des actions à appliquer. Ainsi chaque paquet entrant correspondant à ces critères se verra traiter conformément aux actions définies. N'oublions pas une fois de plus que l'objectif est de garder des switchs très simple dans leur exécution. Il s'agit alors de pouvoir rapidement faire le matching, c'est-à-dire la correspondance paquet – entrée dans la flow table - en se limitant à des champs d'entête facilement récupérables de la couche 2 à 4, même s'il est en réalité possible d'étendre le domaine de matching.

De la même façon, les actions sont relativement simples: on peut jeter un paquet avec DROP, transmettre le paquet avec OUTPUT, le mettre dans une file d'attente spécifique pour appliquer de la QoS, mais également modifier le paquet comme par exemple le TTL, le VLAN voire un autre champ d'entête en supposant qu'une telle spécification est supportée par le switch bien entendu.

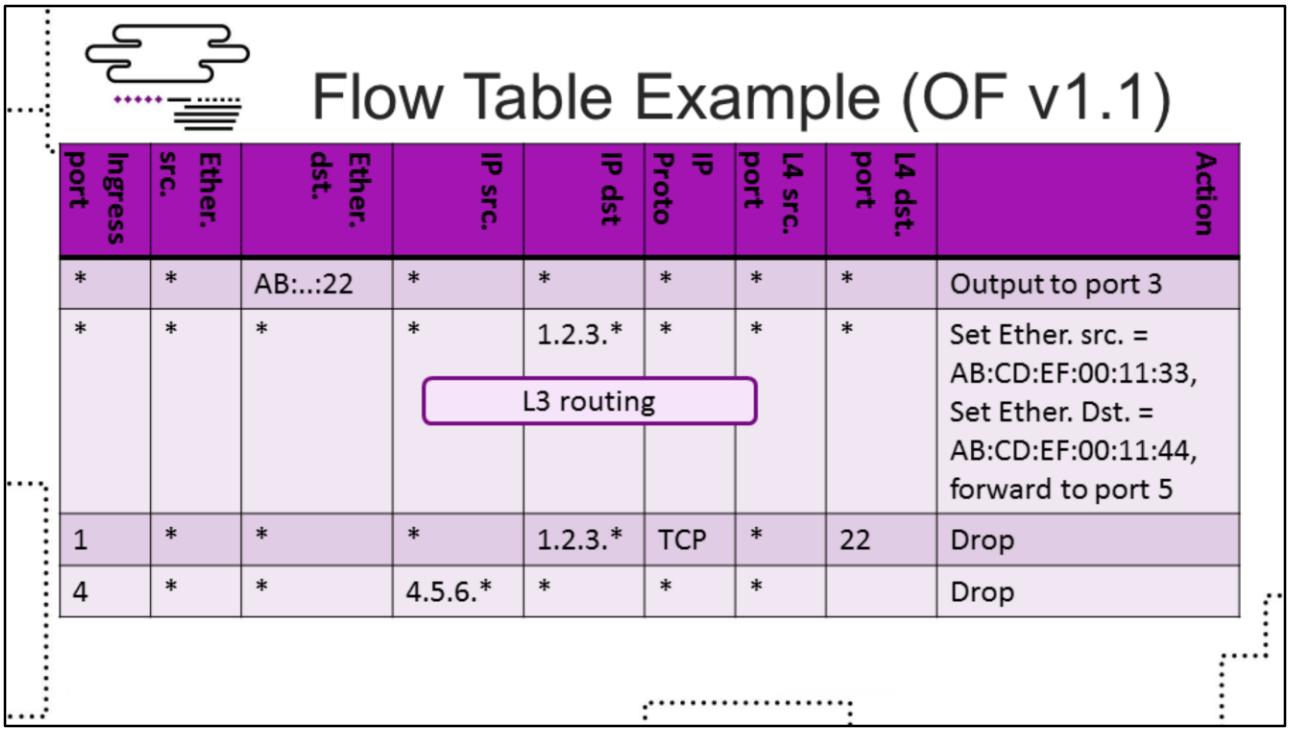
Cette liste n’est pas exhaustive et toutes les actions ne son pas obligatoires. DROP et OUTPUT sont les actions de base ainsi que GROUP qui permet de regrouper dans une même classe plusieurs flux pour mutualiser ensuite la définition des actions à appliquer par groupe.



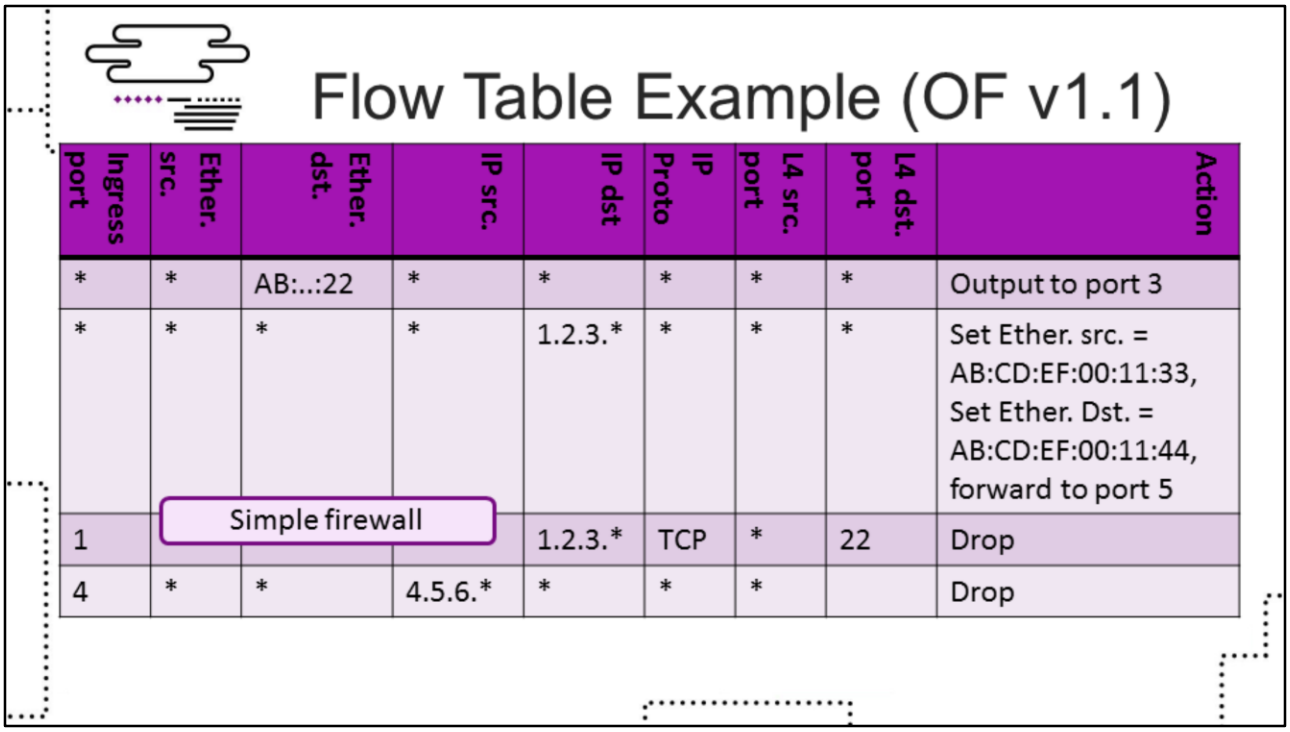
Voici un petit exemple de Flow Table d'après la spécification 1.1 mais qui omet certains champs d'entête par simplification. Notez que la version courante est la 1.4. Les mises à jour de la spécification ont permis d'ajouter de la flexibilité dans les champs qui peuvent être utilisés pour faire le matching.



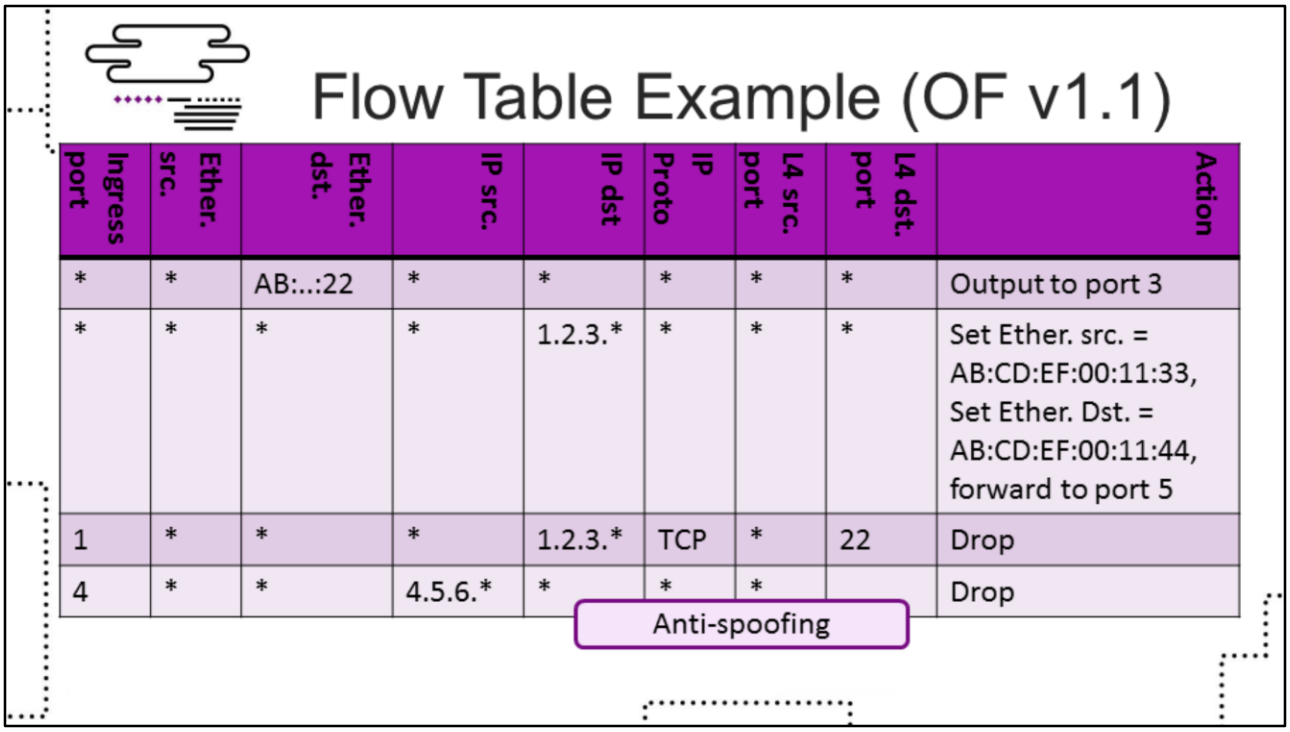
Voici maintenant la table remplie. Premier cas, on filtre d'après l'adressse ethernet de destination pour transmettre le paquet sur le port 3 du switch. Il s'agit simplement d'un switch de niveau 2.
Petite précision: le port spécifié dans l'action de sortie OUTPUT est un port physique du switch et n'a rien à voir avec un port TCP ou UDP.



Deuxième cas, on va filtrer ici les paquets à destination du sous-réseau 1.2.3.0/24 et transmettre le paquet sur le port 5 tout en modifiant les adresses ethernet. C'est le comportement classique d'un routeur IP qui appliquerait une règle de routage.

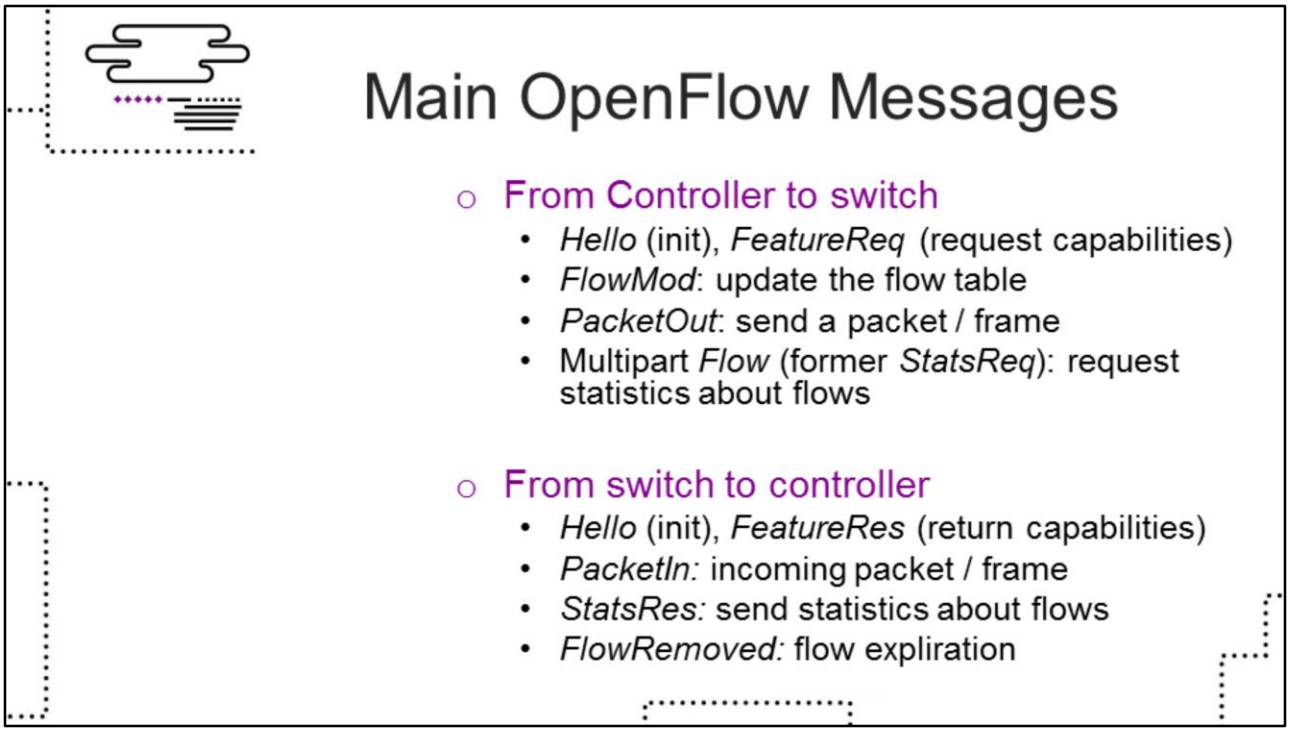


Troisième cas, très simple, on élimine tout paquet vers un sous réseau spécifique pour le port TCP 22. C'est un firewall qui bloque SSH.



On effectue le même traitement dans le dernier cas, mais pour un sous réseau donné et sur un port d'entrée spécifique. Cela peut s'apparenter à une technique anti-spoofing si l'on sait qu'aucun paquet du sous réseau 4.5.6.0/24 ne peut arriver sur ce port pour des raison topologiques par exemple.

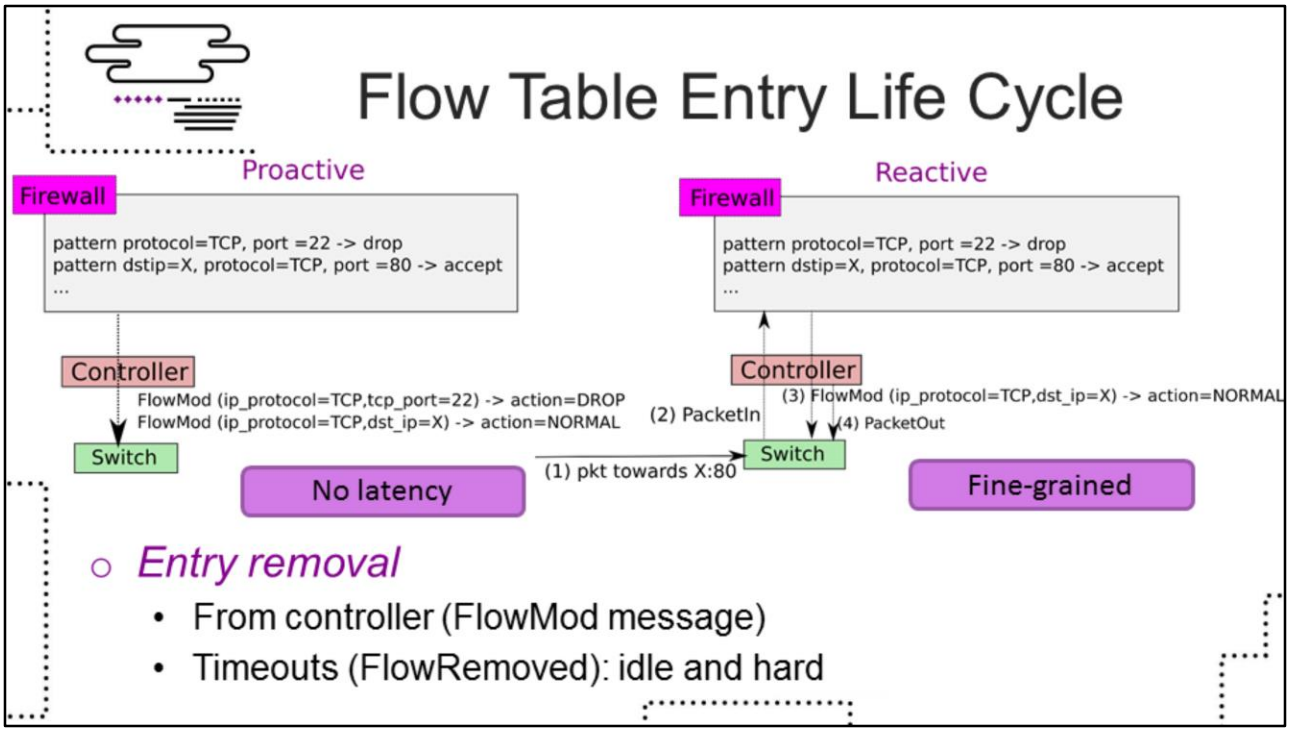
Je ne vous montre ici qu'une table Flow Table mais en réalité plusieurs co-existent et un même paquet peut être traité par plusieurs d'entres elles en cascade. De plus , certaines paquets peuvent correspondre à plusieurs règles, par exemple le routage en ligne 2 et le firewall en ligne 3. Cela représente un conflit qui est résolu par l'utilisation de priorité. Dans ce cas, on assignerait une plus grande priorité au firewall probablement.



Regardons d'un peu plus près le protocole et notamment les messages échangés entre contrôleur et switch.
La configuration des switch est faite à la demande du contrôleur. Après une initialisation, il peut notamment demander les capacités du switch, par exemple, pour savoir s'il est possible de modifier certaines entêtes dans les paquets. La modification de la table flow table s'effectue grâce au message FlowMod. Le message PacketOut ordonne au switch d'envoyer un paquet et il est également possible de récupérer des statistiques (nombre de paquet, durée du flux) .

Cependant, le switch peut aussi notifier le contrôleur en cas d'évènement important comme la réception d'un paquet pour lequel il n'y a pas encore de règle dans la flow table, et ce avec PacketIn. Ou pour informer le contrôleur lorsqu'une entrée elle vient d'expirer. Bien entendu, cette courte liste n'est pas exhaustive et la spécification complète est disponible sur le site de l'Open Networking foundation.

J'ai utilisé ici le terme de paquet mais en réalité comme vous pouvez le voir, les messages PacketIn et PacketOut concernent des trames qui peuvent encapsuler un paquet IP ou tout autre protocole.



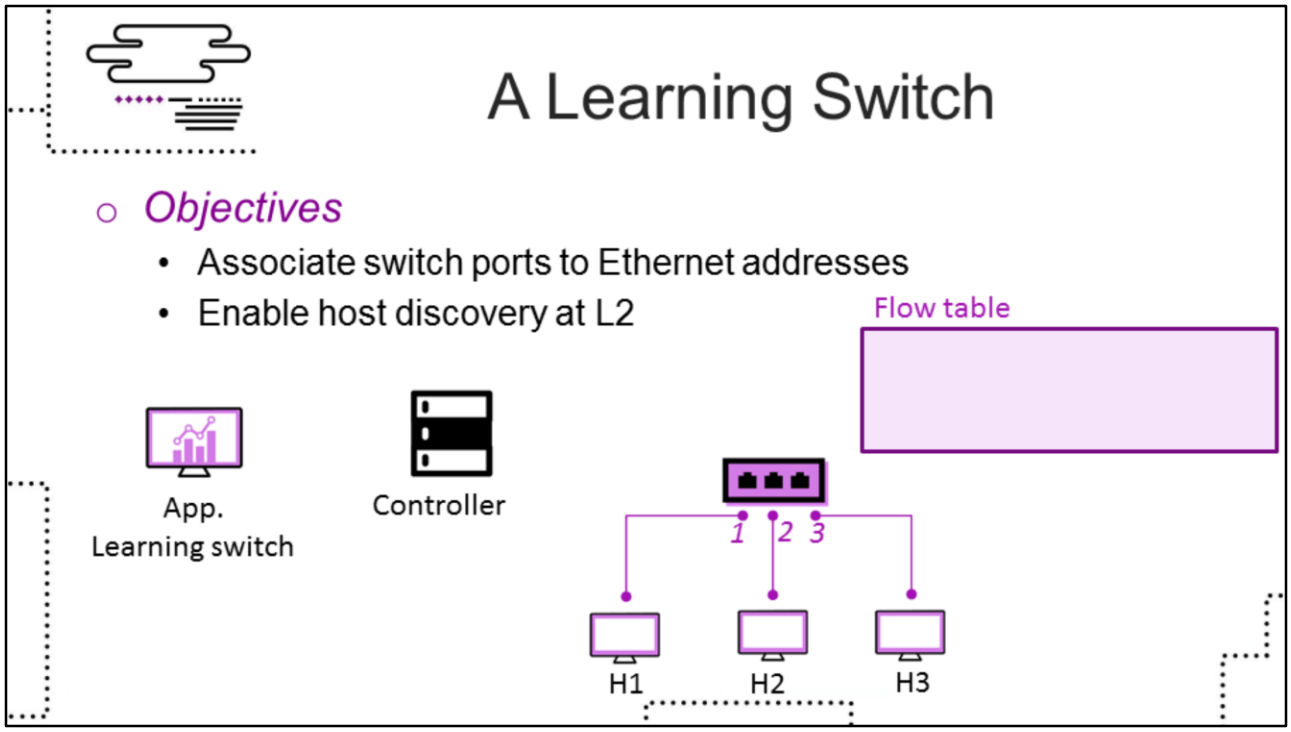
Dans le cas proactif, le contrôleur va pousser toutes les règles dans la flow table. En cas de réception d'un paquet, celui-ci sera donc immédiatement comparée à celle-ci pour appliquer les actions prévues. Il n'y a donc aucune latence dû au processus de décision.

Dans le mode réactif, le switch recevant un paquet pour lequel il n'a pas de règle va en informer le contrôleur. Cela est le résultat d'une entrée un peu spéciale nommée table-miss qui spécifie le comportement par défaut. Dans ce dernier cas, le comportement par défaut est la duplication du paquet reçu vers le contrôleur en utilisant le message PacketIn. Ce dernier peut alors en informer le firewall qui va donc chercher dans sa politique de filtrage quelle règle appliquer.

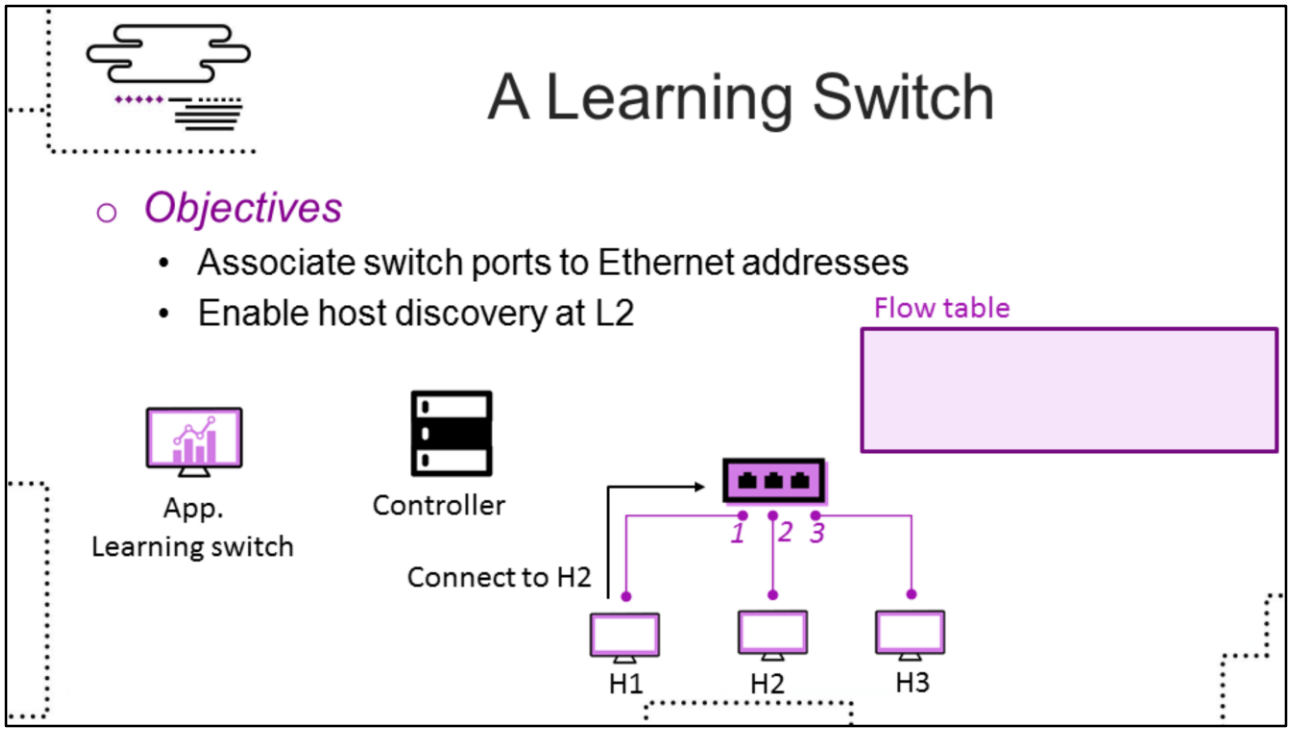
Il installe alors, par le biais du contrôleur la règle correspondante en utilisant FlowMod en (3) puis demande au contrôleur de rejouer le paquet, c'est à a dire de faire ré-entrer le paquet dans le switch avec le message PacketOut. En général, le paquet est gardé en cache et une référence suffit plutôt qu'une recopie complète. Ce type d'approche réactive permet donc d'avoir des règles très fines puisque déduite à partir du trafic. Néanmoins on perd en réactivité, car plus de temps s'écoule le temps que les décisions soient prises. Il est bien entendu possible de mixer les deux modes.

La table flow table, étant sur le switch, a une taille relativement limitée et un mécanisme permet de supprimer les entrées sous certaines conditions, lorsque le

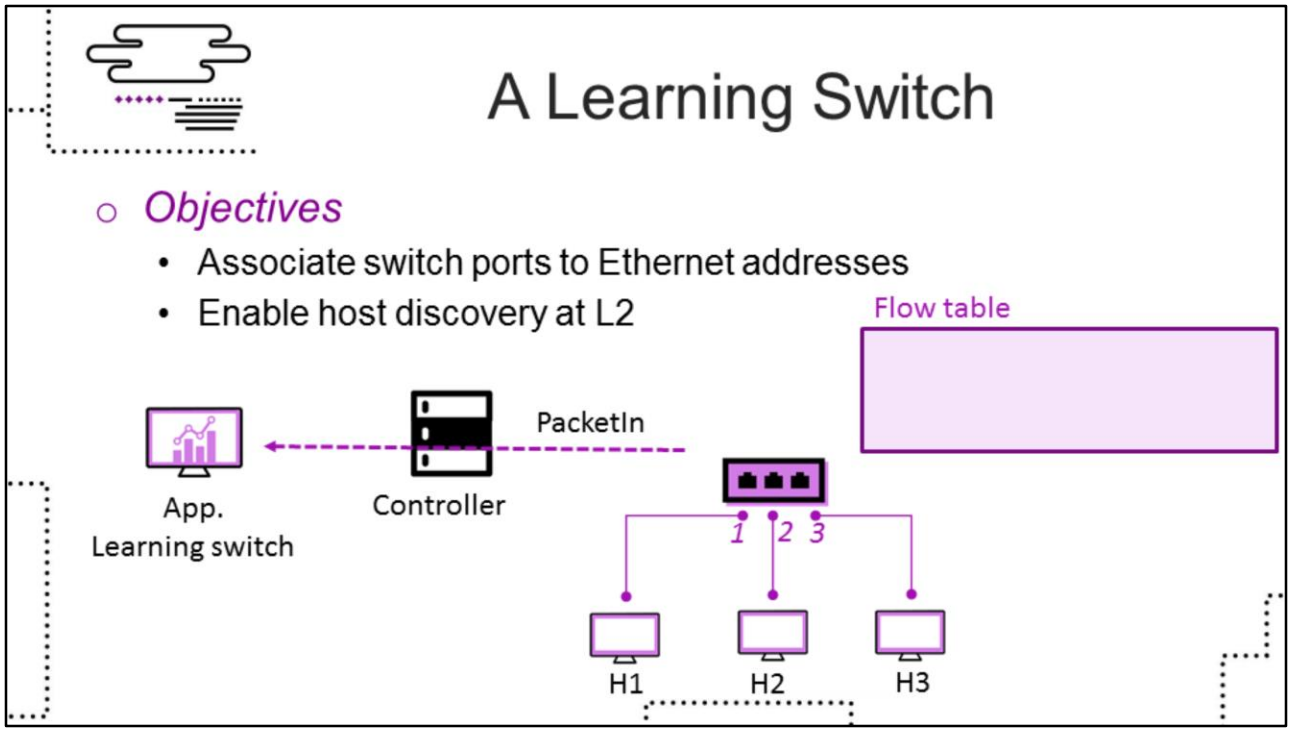
contrôleur le demande, si un délai d'inactivité est observé (idle timepout) ou après un temps arbitraire (hard timeout). Dans ce dernier cas, un des intérêt sous-jacent est de forcer le contrôleur à reconfirmer une règle installée depuis longtemps.



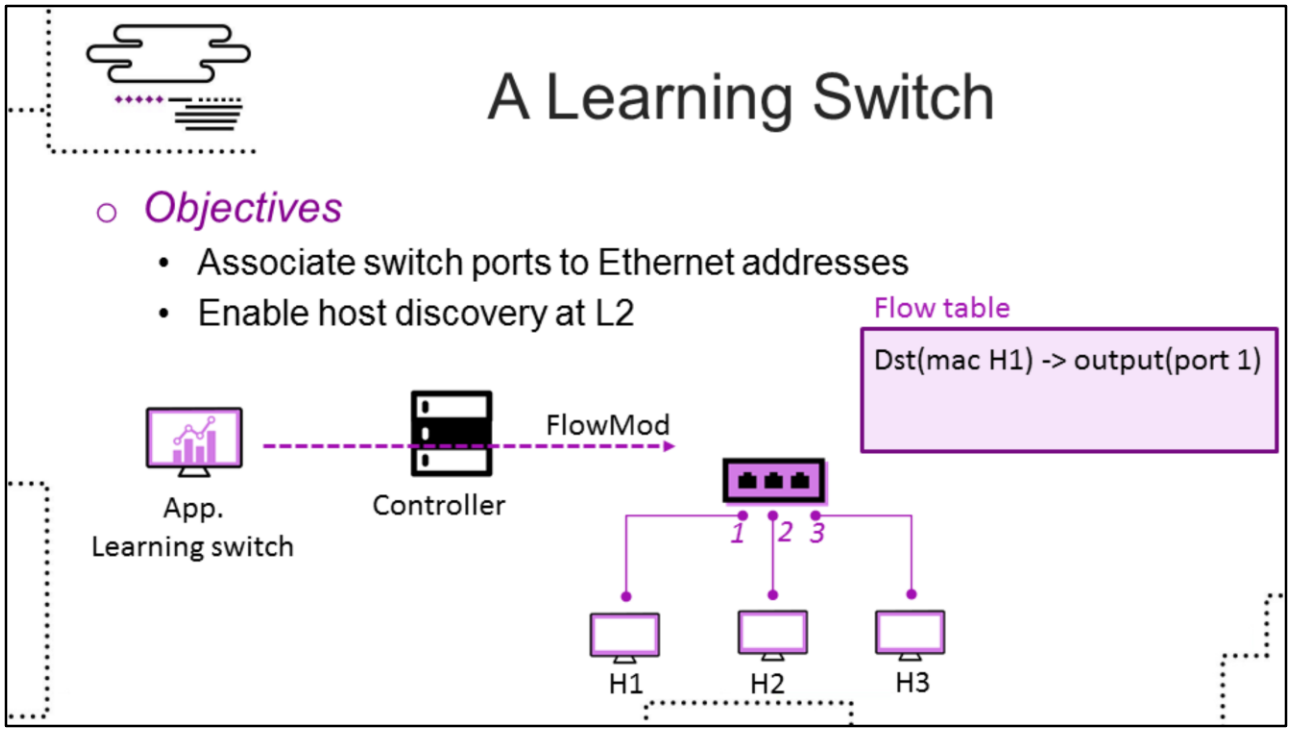
Sur cette diapositive, un exemple classique de switch L2 dont l'objectif est d'apprendre automatiquement sur quel port physique du switch chaque machine, identifiée par son adresse physique, est connectée.



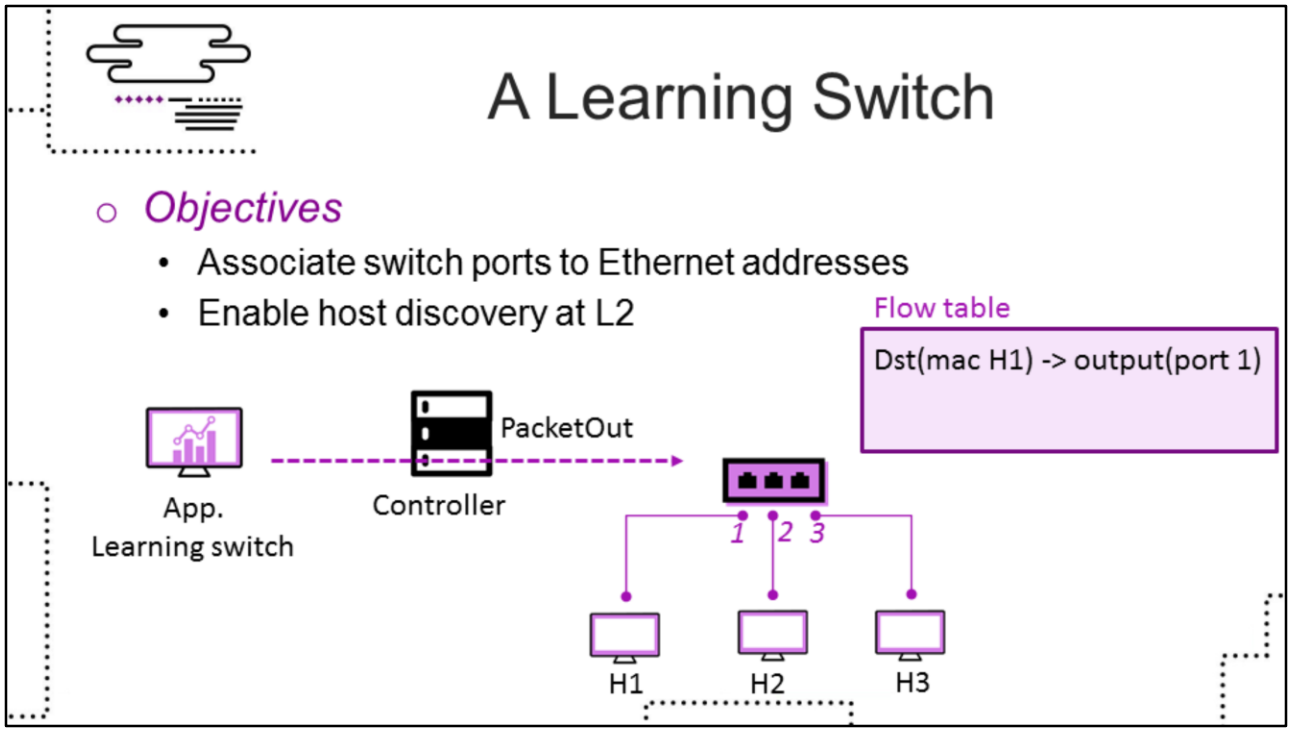
Imaginons que H1 veut envoyer une trame à H2, celle-ci arrive au niveau du switch qui n'a aucune idée où est H2.



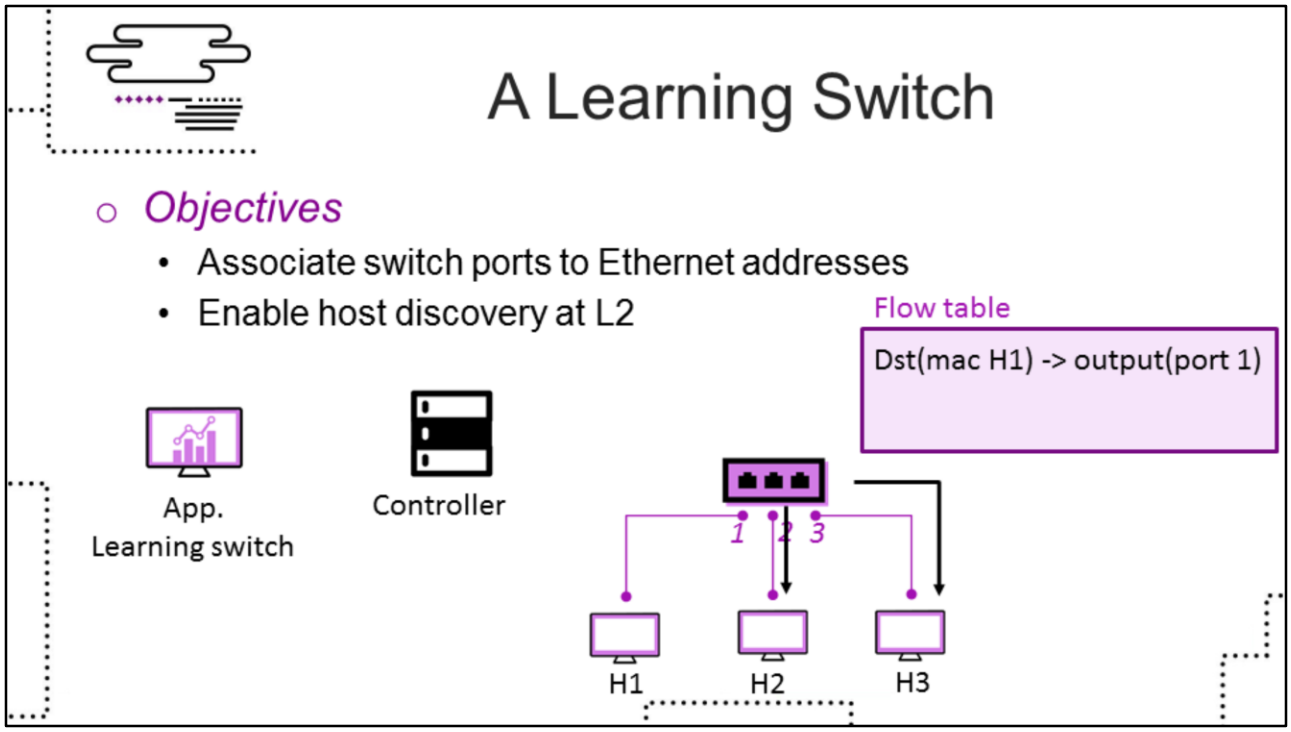
Il demande alors au contrôleur, associé à une application spécifique pour ce cas d'utilisation.



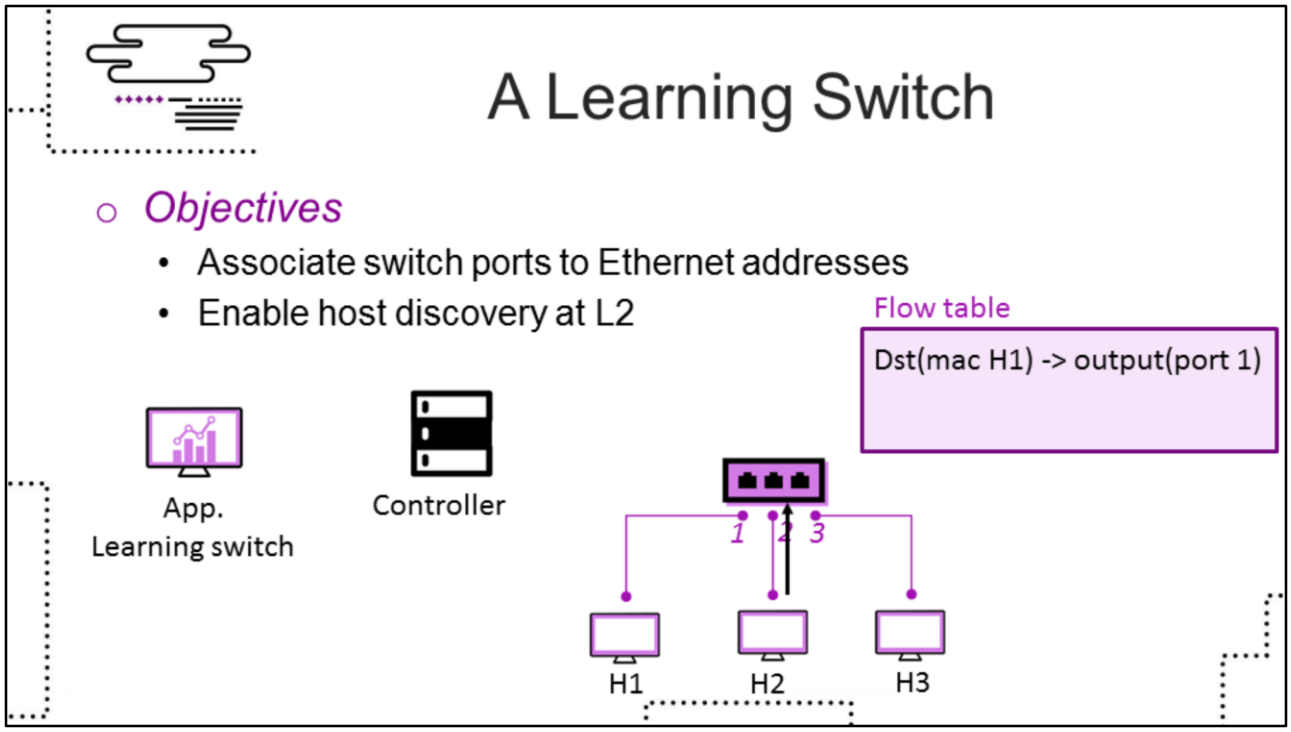
Celui-ci ne connait pas la réponse a priori. Par contre, il sait maintenant que H1 se trouve sur le port 1 et ajoute une règle au switch pour permettre aux paquet suivants à destination de H1 d'être envoyé sur le port 1



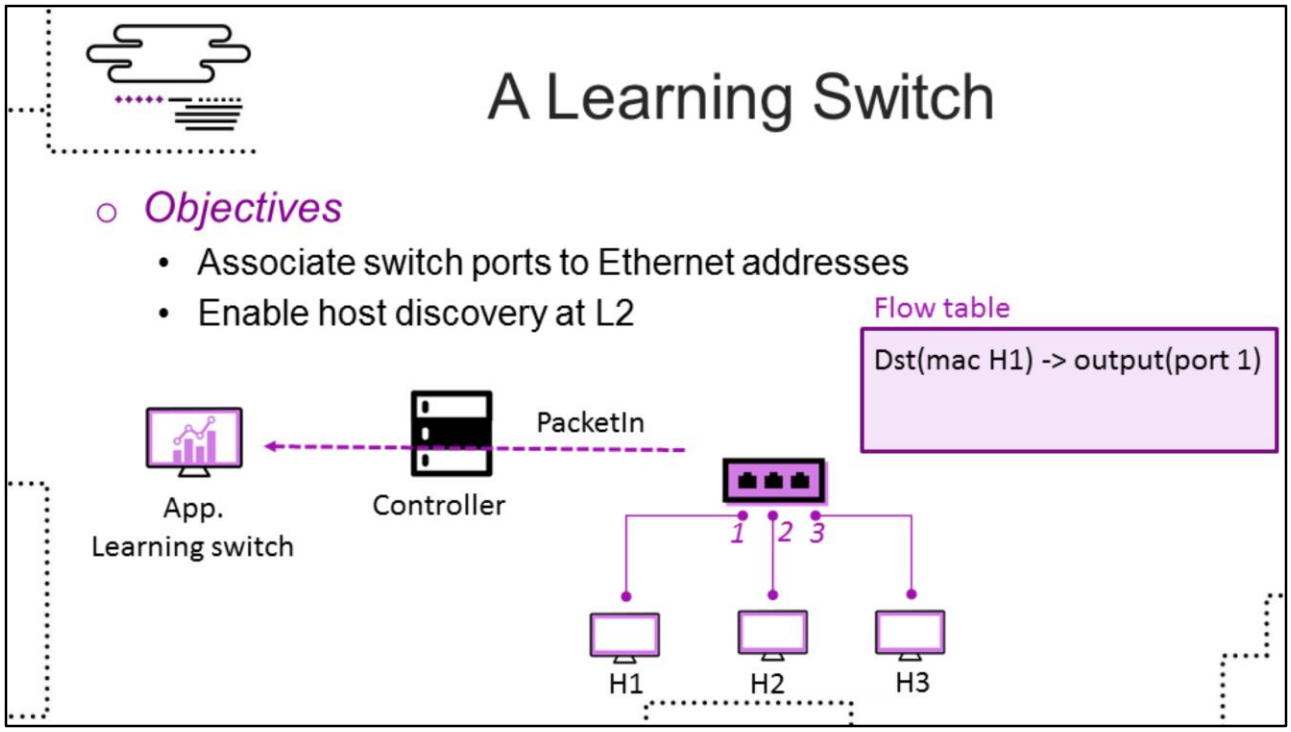
De plus il demande au switch de broadcaster la trame sur l'ensemble des ports sauf celui d'origine.



H2 et H3 vont la recevoir et H2, étant la destination, va le traiter, le décoder et bien souvent y répondre. Cela dépend de l'application mais pensez à un simple simple ping.



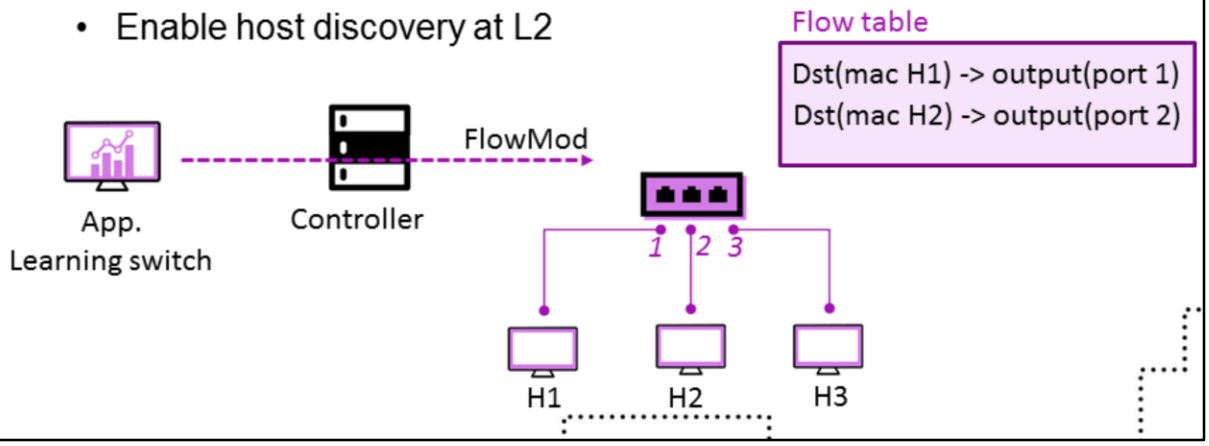
H2 envoie alors une trame à H1



Le port associé à H2 est appris de la même façon que pour H1 (PacketIn, FlowMod, PacketOut).

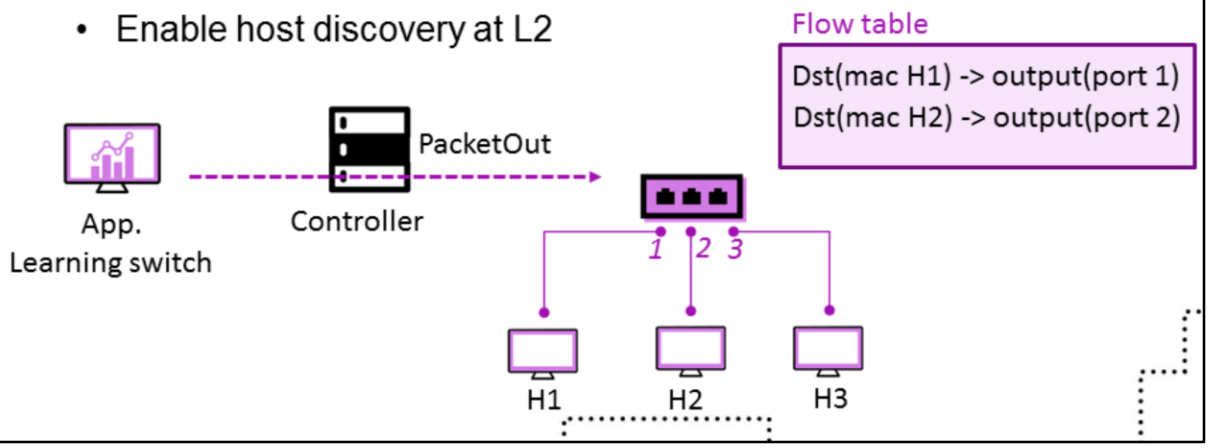
A Learning Switch

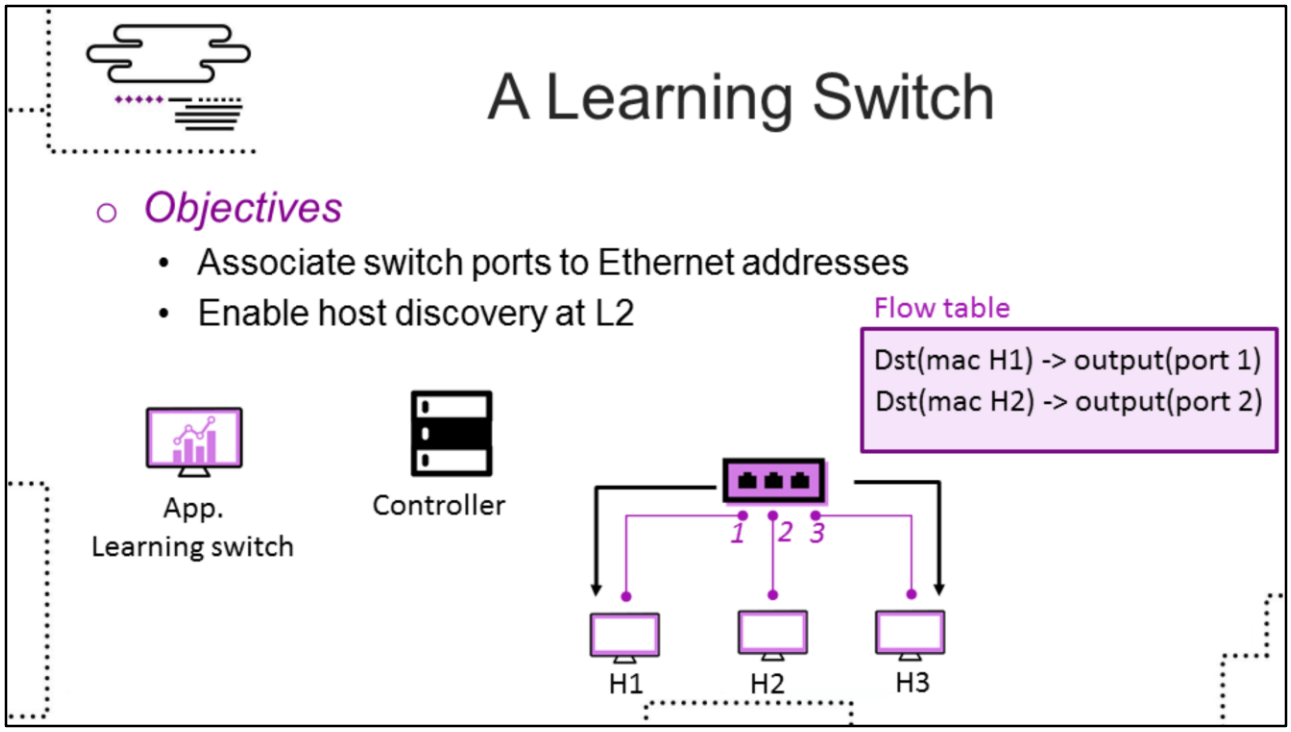
- *Objectives*
  - Associate switch ports to Ethernet addresses
  - Enable host discovery at L2



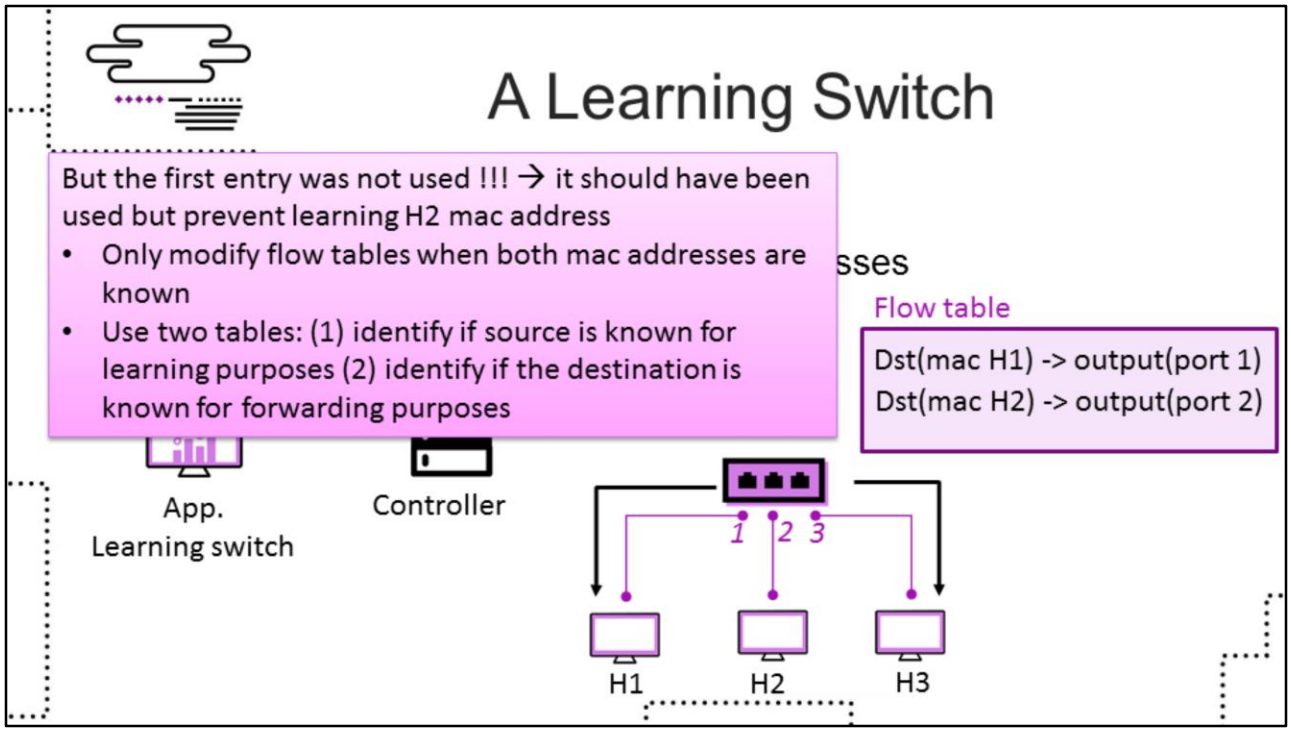
A Learning Switch

- *Objectives*
  - Associate switch ports to Ethernet addresses
  - Enable host discovery at L2

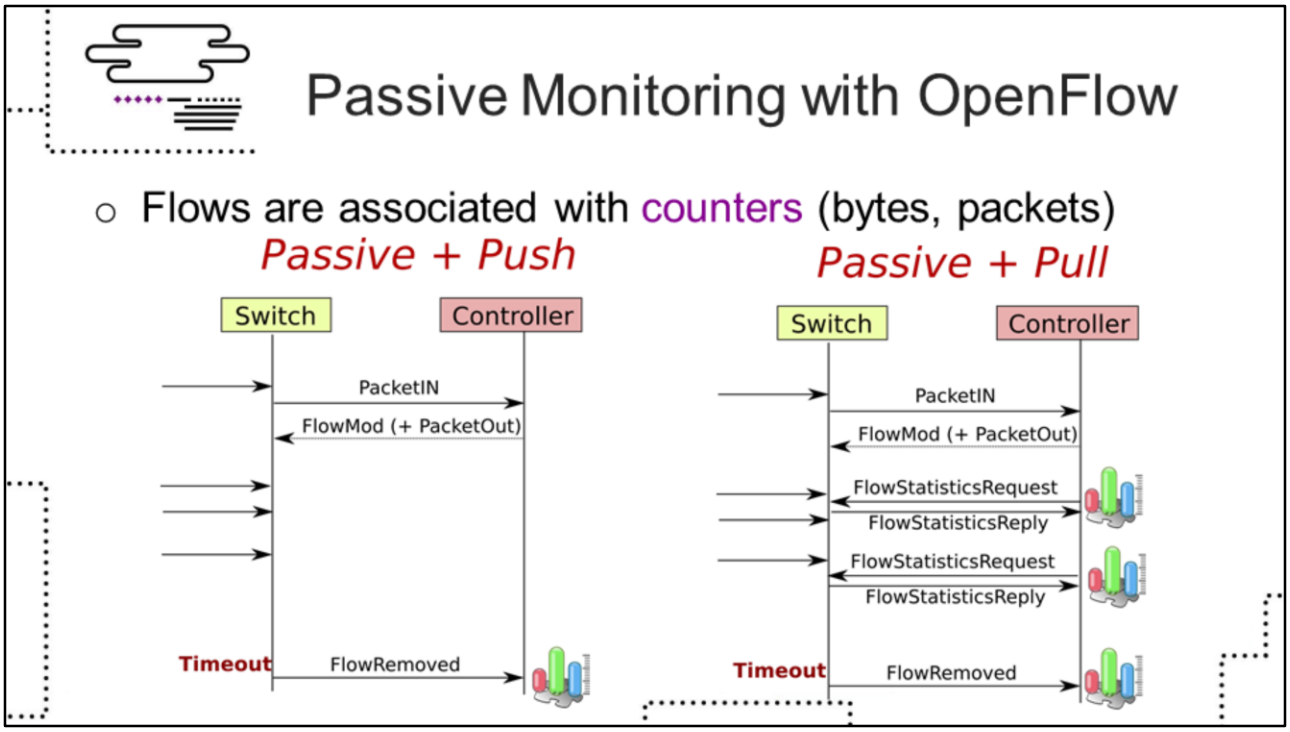




Bizaremment cette dernière trame est aussi broadcastée alors qu'H1 est déjà connue par le switch. En effet, le raisonnement est simplifié. Il ne faut pas lors de la première trame ajouter une règle pour H1, sinon la réponse d'H2 sera directement envoyer à H1 sans que le contrôleur ne soit au courant et ne puisse donc apprendre le port associé à H2

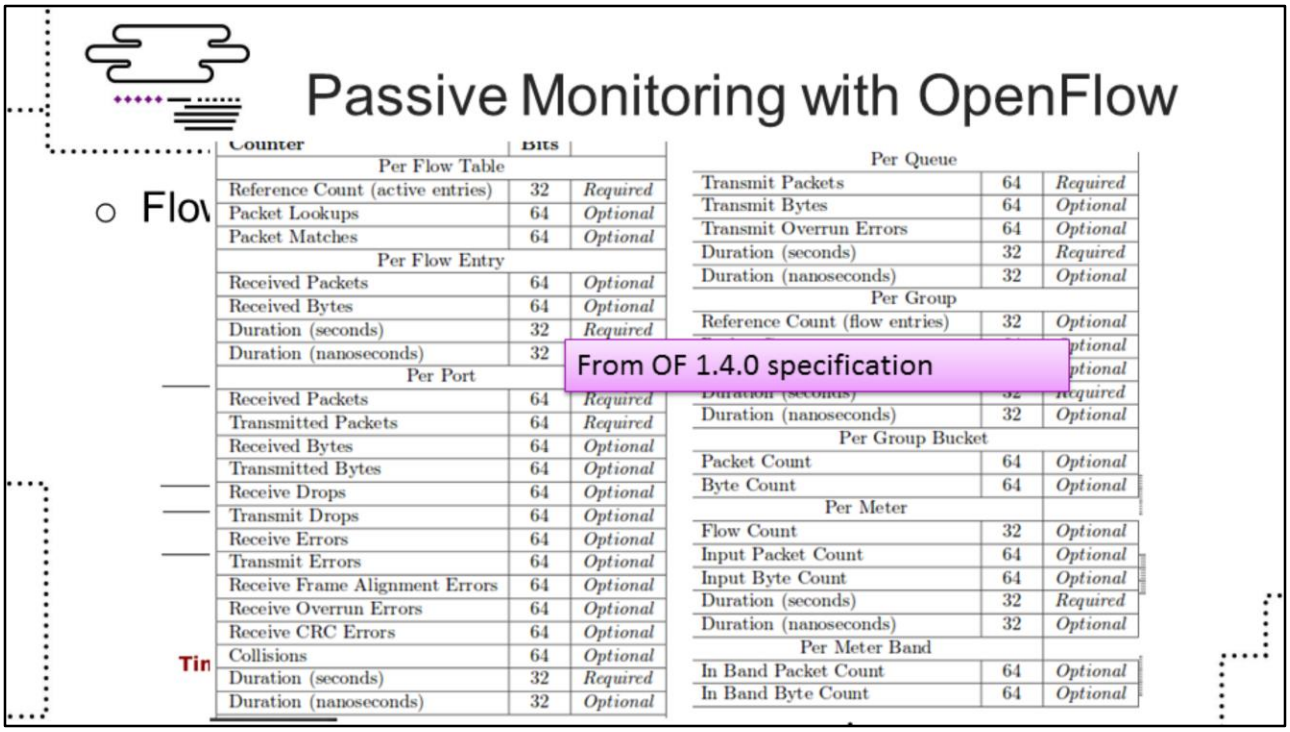


Il faut donc uniquement remplir la flow table lorsque les deux ports sont connus. Cela nécessite donc d'avoir une table intermédiaire, dans notre cas, en attendant la réponse de H2.



OpenFlow peut également être utilisé pour faire du monitoring passif sur le réseau. En effet, des compteurs peuvent être associés au flux comme le nombre de paquets ou d'octets

Ces statistiques sont, soit envoyées à l'expiration d'un flux, soit à la demande du contrôleur. Dans ce second cas, on pourra alors évaluer la dynamique d'un flux dans le temps puisque l'on obtiendra une série temporelle pour la métrique demandée. Sachez qu'il existe un grand nombre de compteurs possibles mais la plupart reste optionnel et leur support est donc dépendent de l'implémentation

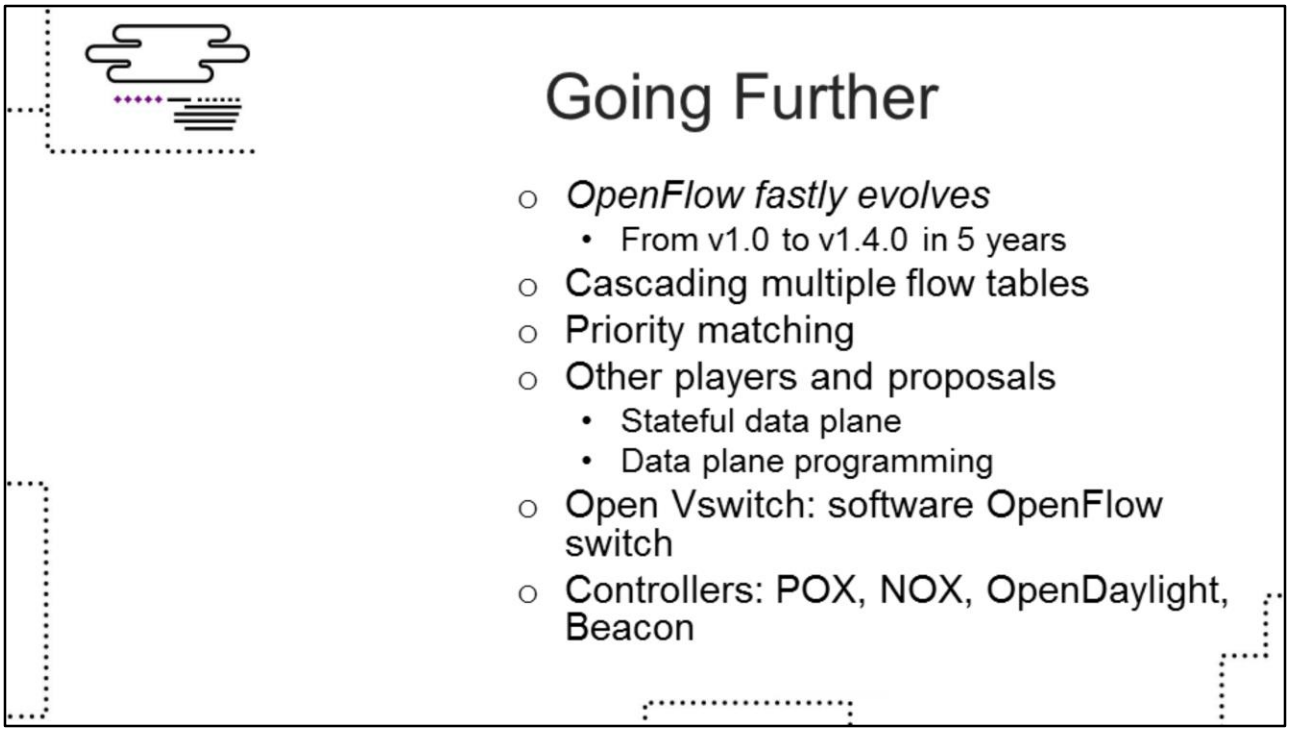


Counter	Bits	
Per Flow Table		
Reference Count (active entries)	32	*Required*
Packet Lookups	64	*Optional*
Packet Matches	64	*Optional*
Per Flow Entry		
Received Packets	64	*Optional*
Received Bytes	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Port		
Received Packets	64	*Required*
Transmitted Packets	64	*Required*
Received Bytes	64	*Optional*
Transmitted Bytes	64	*Optional*
Receive Drops	64	*Optional*
Transmit Drops	64	*Optional*
Receive Errors	64	*Optional*
Transmit Errors	64	*Optional*
Receive Frame Alignment Errors	64	*Optional*
Receive Overrun Errors	64	*Optional*
Receive CRC Errors	64	*Optional*
Collisions	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*

Per Queue		
Transmit Packets	64	*Required*
Transmit Bytes	64	*Optional*
Transmit Overrun Errors	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Group		
Reference Count (flow entries)	32	*Optional*
		Optional
		Optional
		Required
Duration (nanoseconds)	32	*Optional*
Per Group Bucket		
Packet Count	64	*Optional*
Byte Count	64	*Optional*
Per Meter		
Flow Count	32	*Optional*
Input Packet Count	64	*Optional*
Input Byte Count	64	*Optional*
Duration (seconds)	32	*Required*
Duration (nanoseconds)	32	*Optional*
Per Meter Band		
In Band Packet Count	64	*Optional*
In Band Byte Count	64	*Optional*

OpenFlow peut également être utilisé pour faire du monitoring passif sur le réseau. En effet, des compteurs peuvent être associés au flux comme le nombre de paquets ou d'octets

Ces statistiques sont, soit envoyées à l'expiration d'un flux, soit à la demande du contrôleur. Dans ce second cas, on pourra alors évaluer la dynamique d'un flux dans le temps puisque l'on obtiendra une série temporelle pour la métrique demandée. Sachez qu'il existe un grand nombre de compteurs possibles mais la plupart reste optionnel et leur support est donc dépendent de l'implémentation



Pour conclure. OpenFlow étant une technologie récente, elle ne cesse d'évoluer avec l'ajout de fonctionnalités mais aussi la refonte de certaines. Les noms des messages échangés peuvent d'ailleurs différés d'une norme à l'autre. Comme énoncé, il est possible d'avoir plusieurs Flow-Tables, ce qui permet de hiérarchiser les traitements, et des règles de priorité pour lever les ambiguïtés potentielles sur les actions.

Retenez bien qu'OpenFlow n'est pas synonyme de SDN. C'est une proposition parmi d'autres mais celle la plus utilisée actuellement car ouverte et soutenu par un consortium fort. D'autres approches existent et plusieurs reprochent notamment à OpenFlow de ne pas maintenir d'états, comme pour retenir les connexions TCP ouvertes. Cette simplicité est garante d'un filtrage rapide au niveau des règles de la flow table mais reste encore un obstacle à une vraie flexibilité et programmabilité. C'est actuellement un des axes de réflexion dans le monde SDN et il est probable qu'OpenFlow intègre dans des versions futures cette fonctionnalité.

Dans le monde OpenFlow, Open vSwitch est un switch software largement utilisé et de nombreux contrôleurs existent.