2. SDN - Software Defined Networking

2023-2024
Ecole Supérieure Polytechnique (ESP) / Département Génie Informatique (DGI)
DIC3 TR (Télécommunications et Réseaux) & M2 SRT (Systèmes Réseaux et Télécommunications)
Cours : SDN et NFV
Enseignant : Dr Ousmane SADIO

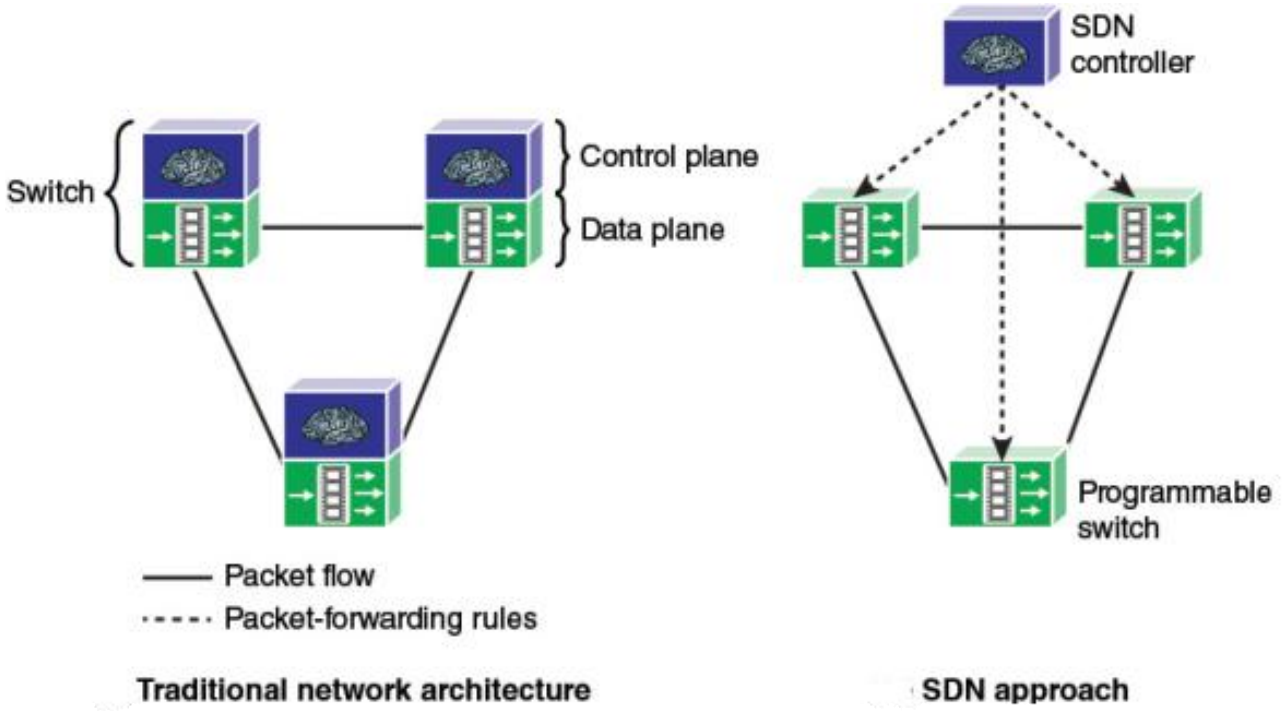
Le Software Defined Networking (SDN) ou réseau défini par logiciel, a atteint un point de basculement qui lui permet de remplacer le modèle réseau traditionnel.

Les réseaux SDN offrent un niveau accru de flexibilité et de personnalisation pour répondre aux besoins des nouvelles tendances en matière de réseaux et d'informatique, telles que le Cloud Computing, le Big Data, la mobilité, les réseaux sociaux...

❑ SDN : définition

Le terme Software-Defined Networking (SDN) se réfère à une architecture réseau qui divise, sur des dispositifs séparés, la fonction de commutation entre un plan de données et un plan de contrôle.

- Plan de données : est simplement responsable de l'acheminement des paquets. Les dispositifs du plan de données deviennent de simples dispositifs d'acheminement de paquets.
- Plan de contrôle : fournit l'"intelligence" en concevant les routes, en définissant les paramètres de priorité et de politique de routage pour répondre aux exigences de QoS et de QoE.



❑ SDN : standards

Contrairement à certains domaines technologiques, comme le Wi-Fi, il n'existe pas d'organisme de standardisation unique chargé de développer des normes ouvertes pour le SDN. Depuis 2014, des efforts de standardisation sont menés, portés notamment par les organismes de standardisation

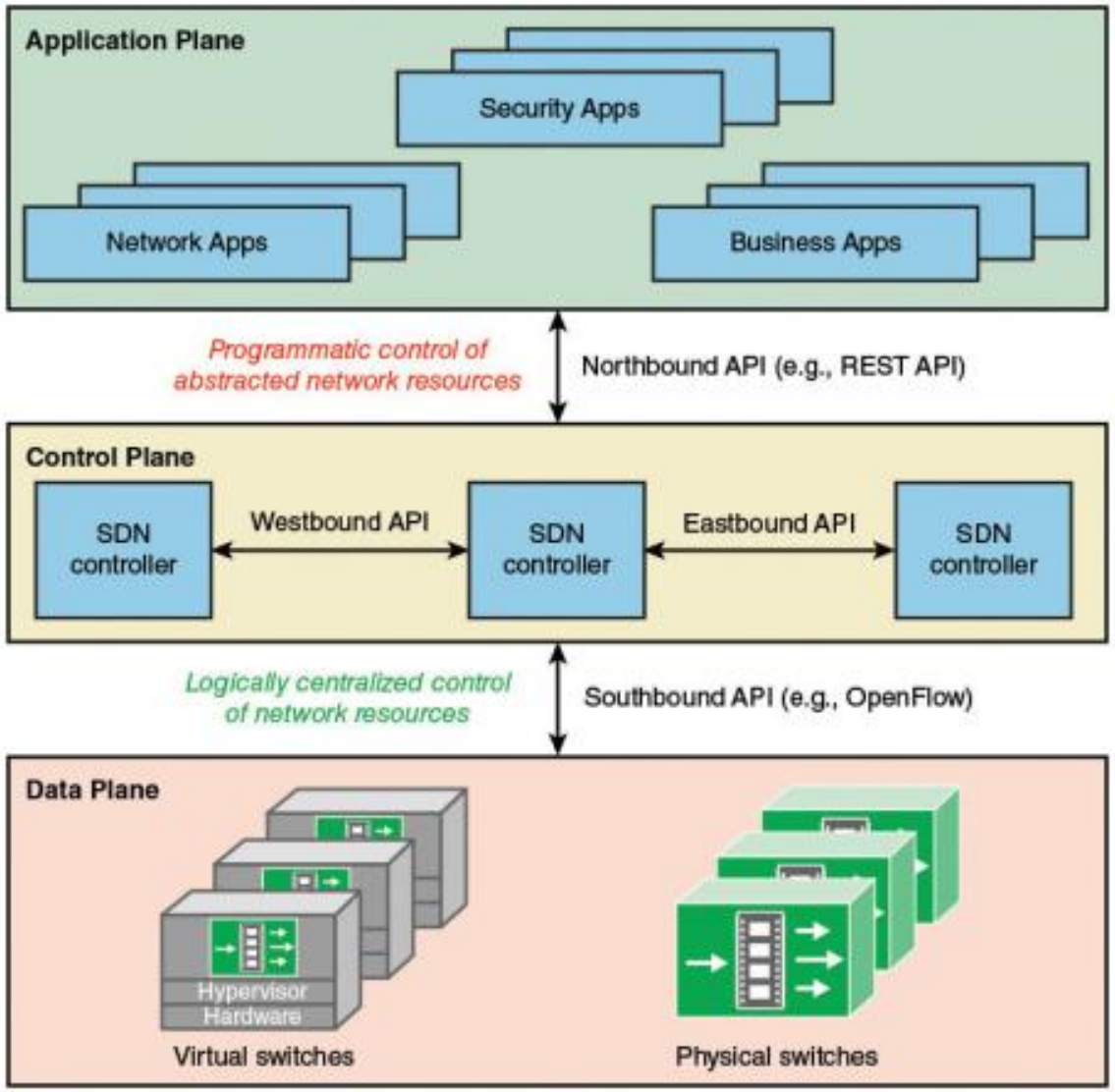
- ONF (Open Networking Foundation)
- IRTF (Internet Research Task Force)
- ITU (International Telecommunications Union)
- IETF (Internet Engineering Task Force)

Malgré des approches et focus différents, ces organismes ont un objectif en commun, permettre la programmabilité des réseaux.

❑ SDN : architecture

L'architecture représentée ici reste le plus répandu.

- Switches : constituent le plan de données. Ils sont responsables de l'acheminement des paquets.
- Contrôleur : implémente le plan de contrôle. Le contrôleur dispose d'une vue centralisée du ou des réseaux qu'il contrôle. C'est un logiciel capable de programmer les switches.
- Applications : sont des programmes qui peuvent utiliser une vue abstraite du réseau pour leurs objectifs décisionnels. Ils transmettent leurs exigences et le comportement souhaité du réseau au contrôleur SDN.
- Interfaces : sont définies pour permettre d'une part, aux applications de communiquer avec les contrôleurs SDN (Northbound API), et d'autres part, aux switches de communiquer avec le contrôleur (Southbound API).

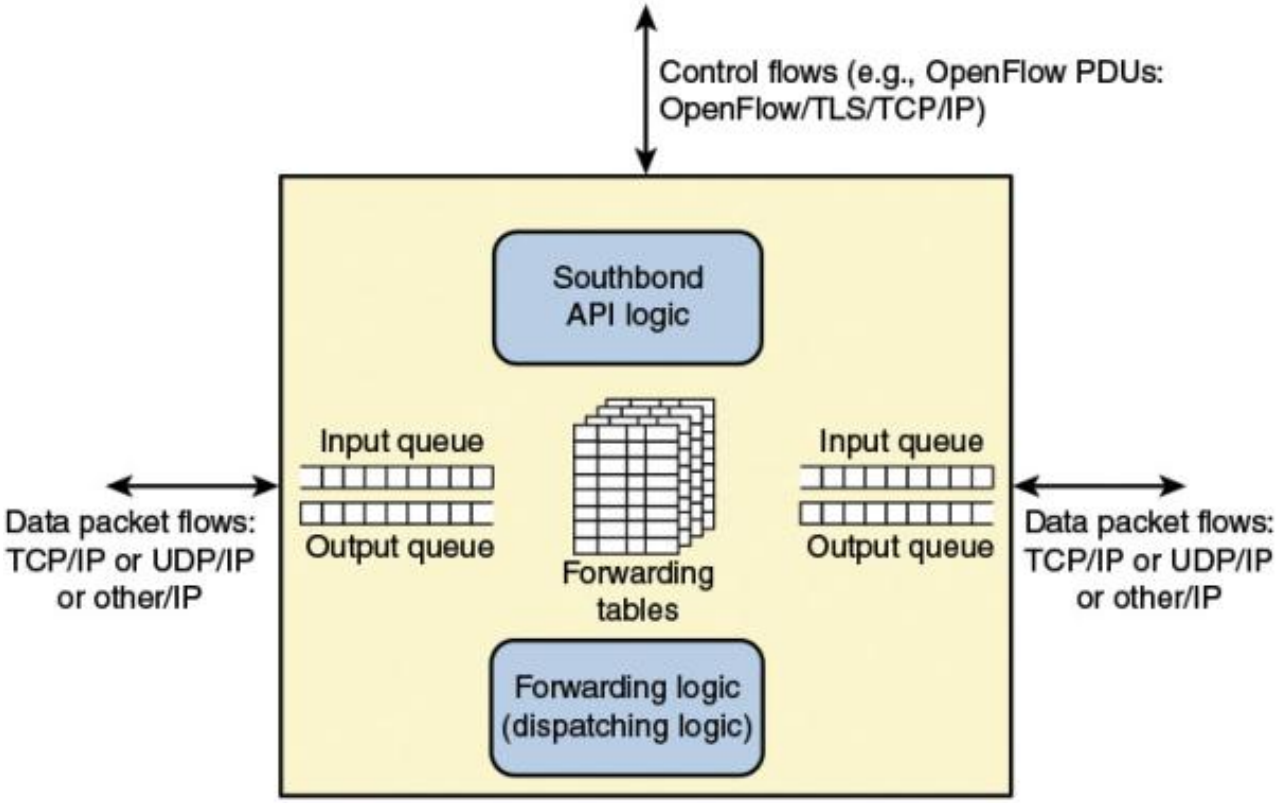


❑ Plan de données

Le plan de données SDN, également appelé couche d'infrastructure, est le lieu où les dispositifs d'acheminement réseau effectuent le transport et le traitement des données conformément aux décisions prises par le plan de contrôle SDN.

Les principales fonctions du switch SDN sont les suivantes :

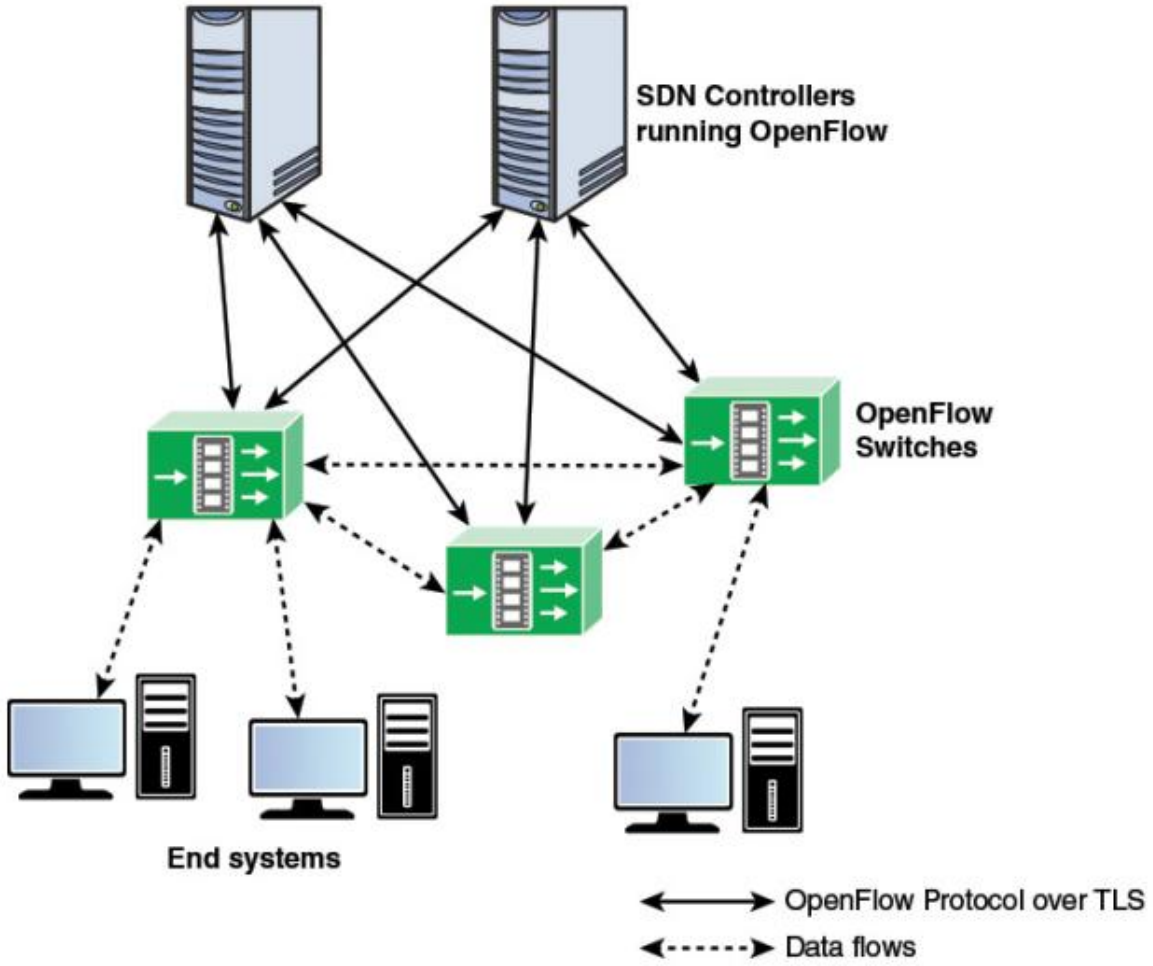
- Fonction de support de contrôle : le switch communique avec le contrôleur. Ce dernier gère le commutateur via le protocole de commutation OpenFlow.
- Fonction d'acheminement de données : accepte les flux de données entrants et les transmet le long des chemins de données qui ont été calculés et établis selon les règles définies par les applications SDN.



❑ Plan de données : OpenFlow

Les exigences d'une implémentation pratique du SDN sont prises en compte par OpenFlow qui est à la fois :

- Un protocole entre le contrôleur SDN et les dispositifs réseau,
- Et une spécification de la structure logique de la fonctionnalité des switches SDN.



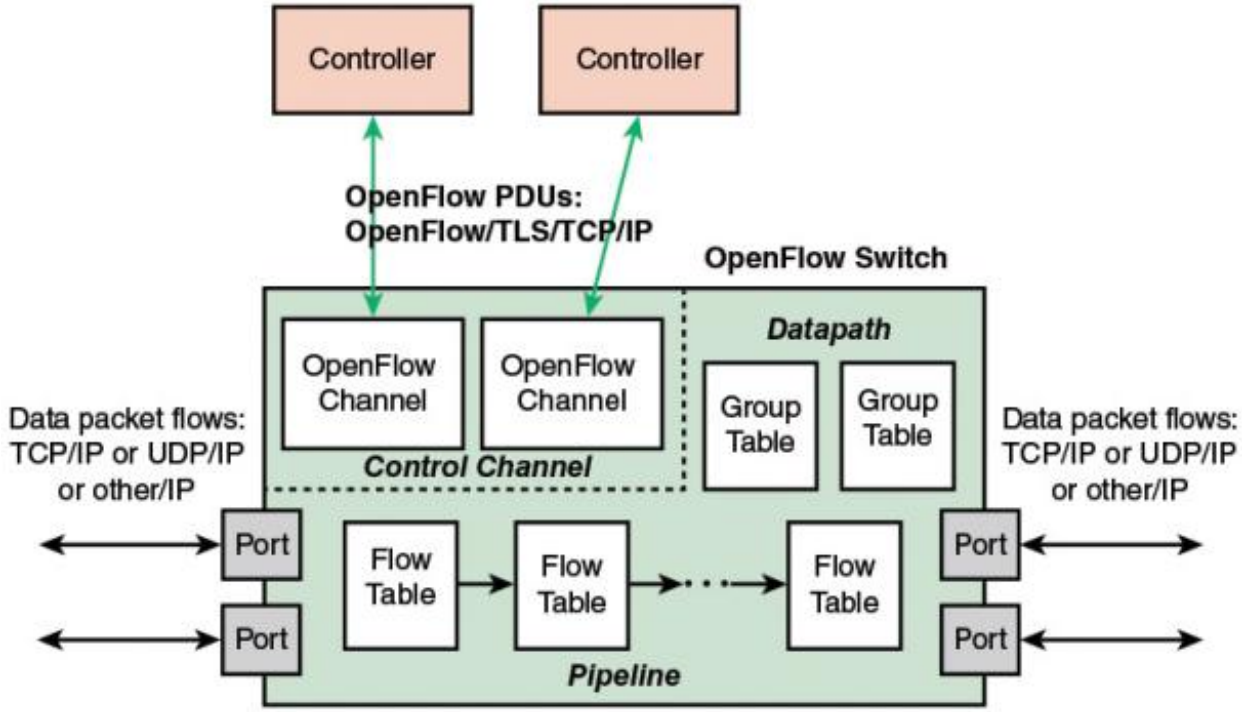
❑ Switch OpenFlow

Chaque commutateur se connecte à d'autres commutateurs OpenFlow via des interfaces appelées canal OpenFlow. Ces connexions se font via des ports OpenFlow.

Le contrôleur SDN peut ajouter, mettre à jour et supprimer des entrées de flux dans les tables, à la fois de manière réactive (en réponse aux paquets) et proactive.

Dans chaque switch, une série de tables est utilisée pour gérer les flux de paquets à travers le commutateur.

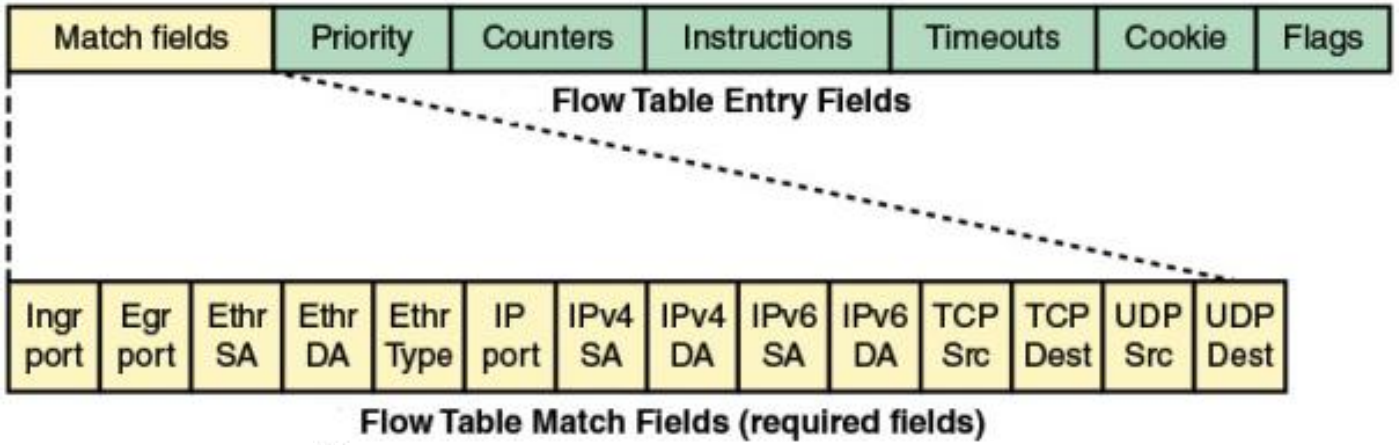
- Table de flux : fait correspondre les paquets entrants à un flux particulier et spécifie les fonctions à exécuter sur les paquets.
- Groupe de table : peuvent déclencher une variété d'actions qui affectent un ou plusieurs flux.
- Table de mesure : peut déclencher une variété d'actions liées à la performance d'un flux.



❑ Switch OpenFlow : structure de la table des flux

Chaque paquet qui entre dans un commutateur passe par une ou plusieurs tables de flux. Chaque table de flux est constituée d'un certain nombre de lignes, appelées entrées.

- Match fields : utilisé pour sélectionner les paquets qui correspondent aux valeurs dans les champs.
- Priority : champ de 16 bits avec 0 comme à la priorité la plus faible.
- Counters : mise à jour pour les paquets correspondants.
- Instructions : à exécuter en cas de correspondance (match).
- Timeouts : durée maximale d'inactivité avant qu'un flux ne s'expire.



- Cookies : valeur de données opaque de 64 bits choisie par le contrôleur.
- Flags : modifient la manière dont les entrées de flux sont gérées.

Un flux est une séquence de paquets qui correspond à une entrée spécifique dans une table de flux.

❑ Switch OpenFlow : structure de la table des flux

- Instructions

Les instructions d'une entrée de table est constitué d'un ensemble d'instructions qui sont exécutées si le paquet correspond à l'entrée.

Les **actions** décrivent les opérations de transfert de paquets, de modification de paquets et de traitement des tables de groupe.

- Output: transférer le paquet au port spécifié.
- Set-Queue : définit l'ID de la file d'attente pour un paquet.
- Group : traiter le paquet par le groupe spécifié
- Push-Tag/Pop-Tag : push ou pop un champ de tag pour un VLAN ou un paquet MPLS.
- Set-Field : modifient les valeurs des champs d'en-tête dans le paquet.
- Change-TTL: modifier les valeurs du TTL IPv4.
- Drop : abandon de paquet.

Les types d'**instructions** peuvent être regroupés en quatre catégories :

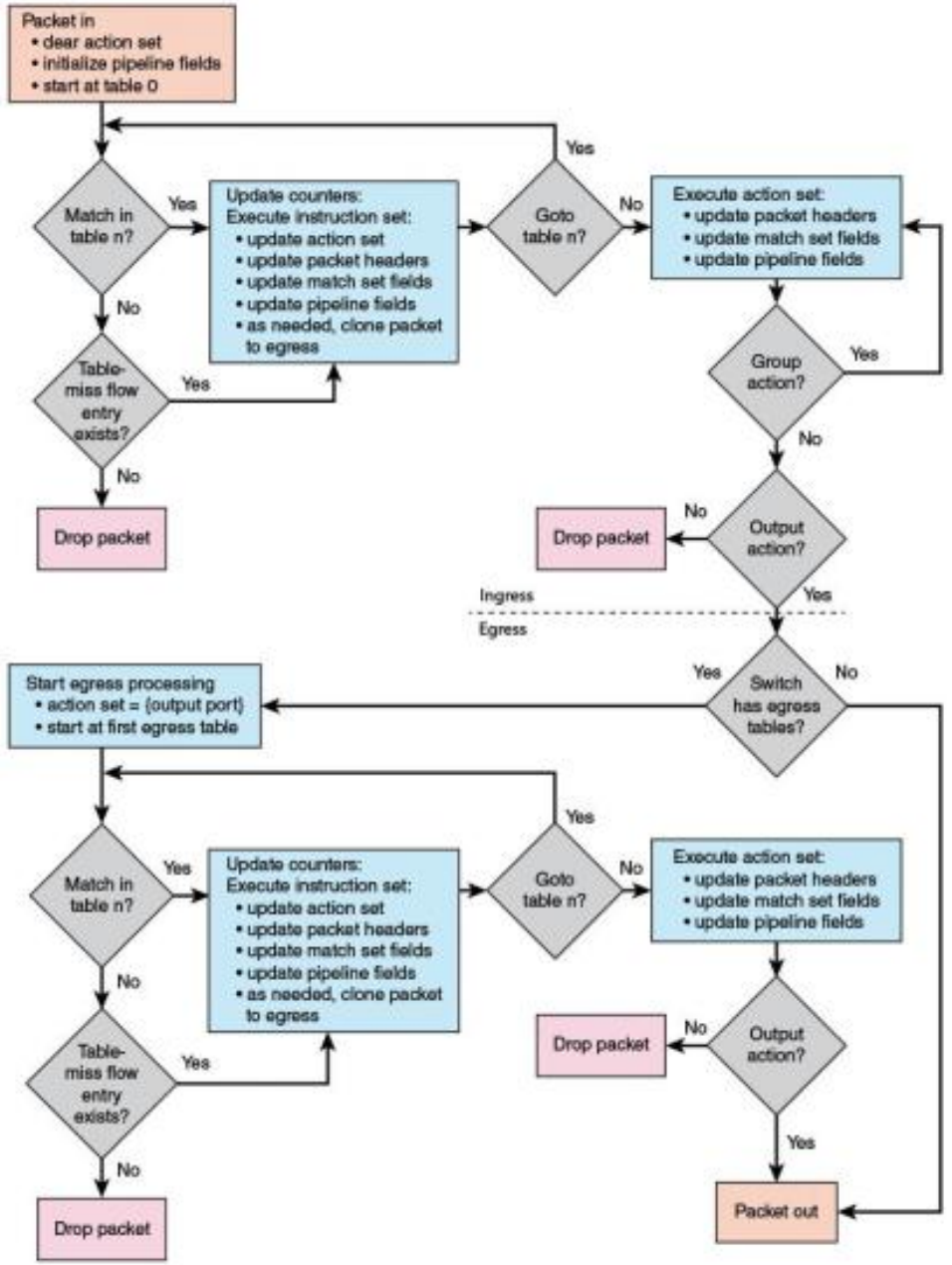
- Direct packet through pipeline : Goto-Table dirige le paquet vers le pipeline.
- Perform action on packet : Apply-Actions applique immédiatement les actions spécifiées.
- Update action set : Write-Actions fusionne les actions spécifiées dans le jeu d'actions actuel pour le paquet. Clear Actions efface toutes les actions du jeu d'actions.
- Update metadata : Write-Metadata met à jour une valeur de métadonnées existante ou crée une nouvelle valeur.

❑ Switch OpenFlow : structure de la table des flux

- Pipeline de la table de flux

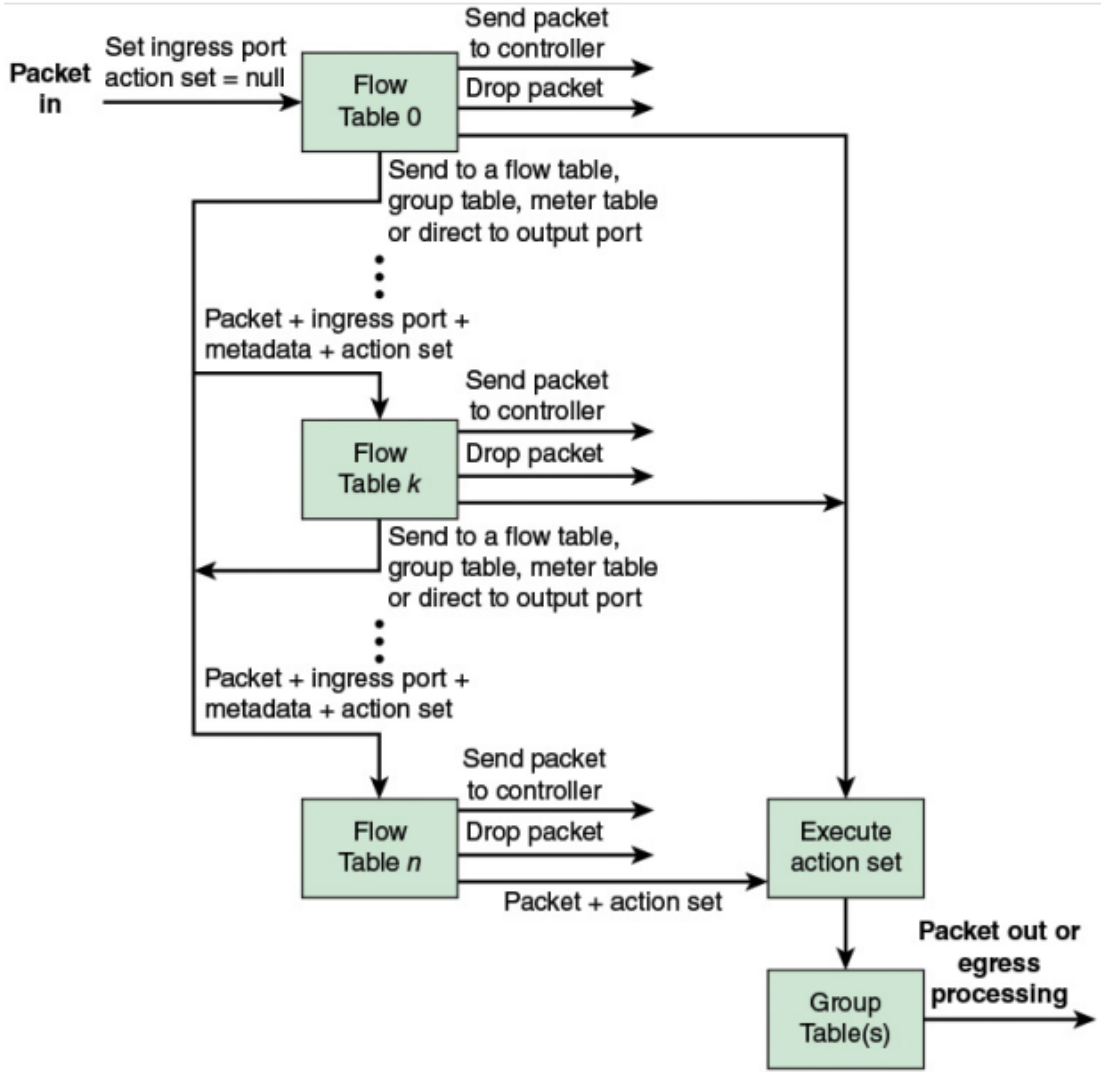
Un commutateur comprend une ou plusieurs tables de flux. S'il y a plus d'une table de flux, elles sont organisées comme un pipeline. La spécification OpenFlow définit deux étapes de traitement :

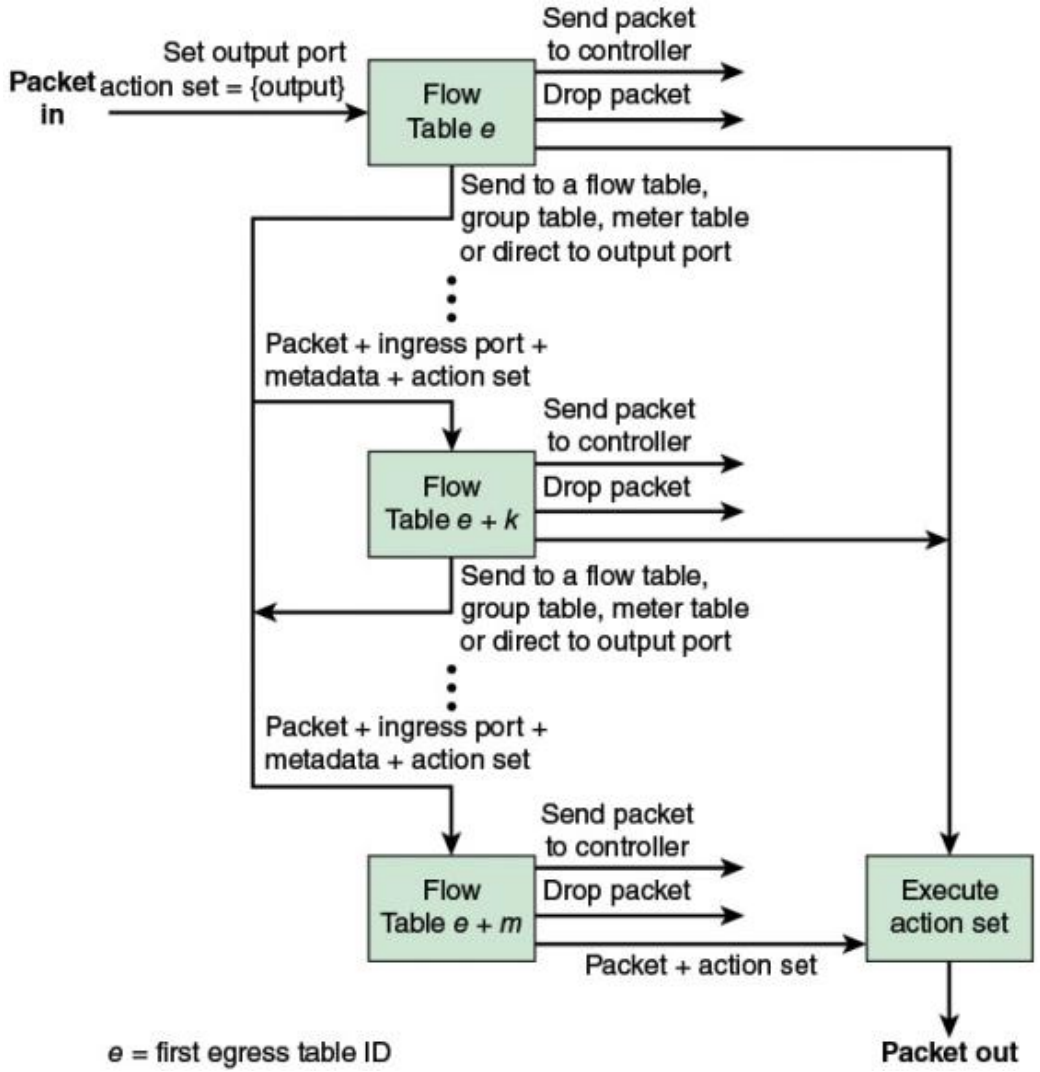
- Ingress processing : le traitement d'entrée a toujours lieu, en commençant par la table 0, et utilise l'identité du port d'entrée. La table 0 peut être la seule table, auquel cas le traitement à l'entrée est simplifié au traitement effectué sur cette seule table, et il n'y a pas de traitement à la sortie.
- Egress processing : le traitement de sortie est le traitement qui se produit après la détermination du port de sortie. Cette étape est facultative. Si elle a lieu, elle peut impliquer une ou plusieurs tables.



❑ Switch OpenFlow : structure de la table des flux

Illustration du processus global des pipelines *ingress* et *egress*.





❑ Protocole OpenFlow

Le protocole OpenFlow décrit les échanges de messages qui ont lieu entre un contrôleur SDN et un commutateur OpenFlow. Il permet au contrôleur d'effectuer des actions d'ajout, de mise à jour et de suppression des entrées dans les tables de flux. Prend en charge trois catégories de messages :

- Contrôleur vers le switch : ces messages sont initiés par le contrôleur et, dans certains cas, nécessitent une réponse du commutateur. Cette classe de messages permet au contrôleur de gérer l'état logique du commutateur, y compris sa configuration et les détails des entrées de la table de flux et de groupe. Le message Packet-out fait également partie de cette classe.
- Asynchrone : ces types de messages sont envoyés sans sollicitation de la part du contrôleur. Cette classe comprend divers messages d'état destinés au contrôleur. Elle comprend également le message Packet-in.
- Symétrique : ces messages sont envoyés sans sollicitation de la part du contrôleur ou du commutateur. Ex : messages *hello* échangés par le contrôleur et le switch lorsque la connexion est établie pour la première fois. Messages *Echo Request / Echo Response* pour mesurer la latence, débit et le statut du switch.

❑ Protocole OpenFlow

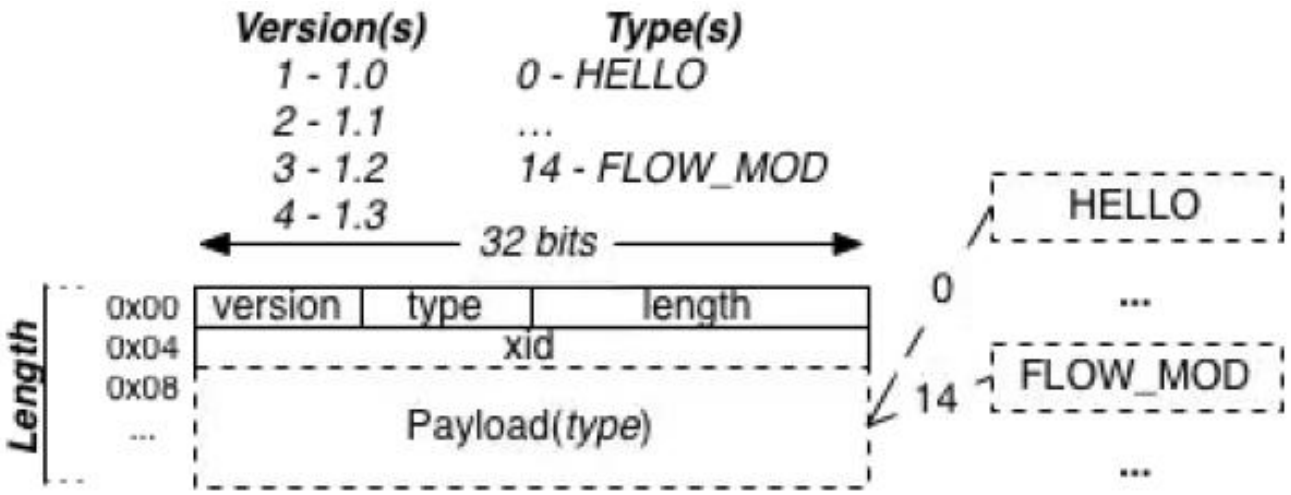
En général, le protocole OpenFlow fournit au contrôleur SDN trois types d'informations à utiliser pour la gestion du réseau :

- Messages basés sur des événements : envoyé par le commutateur au contrôleur lorsqu'un changement de liaison ou de port.
- Statistique sur le flux : généré par le switch en fonction du flux de trafic. Ces informations permettent au contrôleur de surveiller le trafic, de reconfigurer le réseau si nécessaire et d'ajuster les paramètres de flux pour répondre aux exigences de la QoS.
- Paquets encapsulés : envoyé par le switch au contrôleur soit parce qu'il y a une action explicite d'envoi de ce paquet indiquée dans une entrée de la table de flux, soit parce que le switch a besoin d'informations pour savoir le traitement à faire avec ce paquet.

❑ Protocole OpenFlow : messages

Chaque message OpenFlow commence par la même structure d'entête. Cette structure fixe remplit trois rôles qui sont indépendants de la version d'OpenFlow utilisée.

- Version : (8 bits) indique la version d'OpenFlow.
- Type : (8 bits) indique quel type de message est présent et comment interpréter les données utiles (*ce champ dépend des versions OpenFlow*).
- Lenght : (16 bits) indique où le message se terminera dans le flux d'octets à partir du premier octet de l'entête.
- xid : (32 bits) ou identifiant de transaction, est une valeur unique utilisée pour faire le matching requête / réponse.

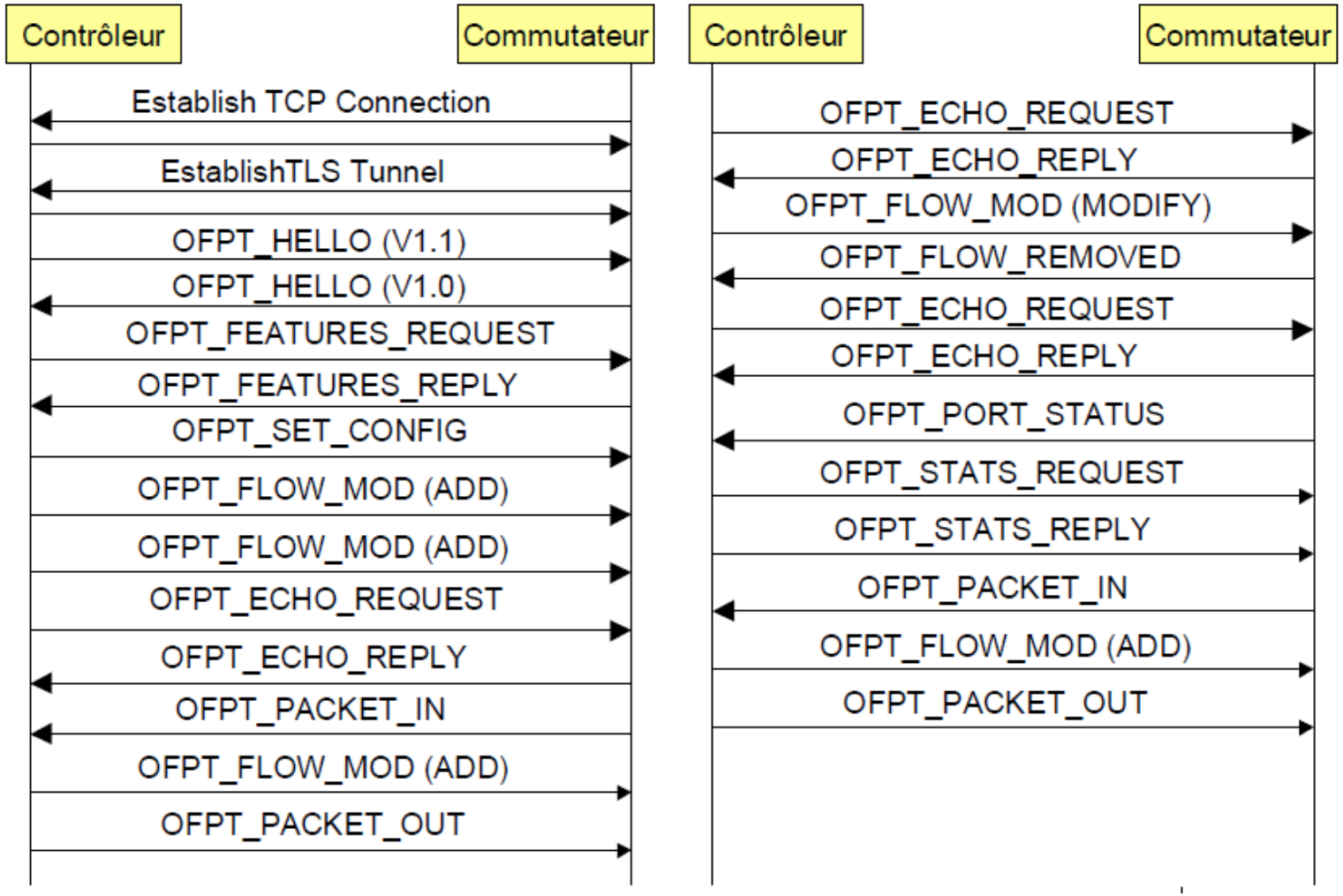


❑ Protocole OpenFlow : messages

```
/* Immutable messages. */
OFPT_HELLO                = 0,  /* Symmetric message */
OFPT_ERROR                = 1,  /* Symmetric message */
OFPT_ECHO_REQUEST         = 2,  /* Symmetric message */
OFPT_ECHO_REPLY           = 3,  /* Symmetric message */
OFPT_EXPERIMENTER         = 4,  /* Symmetric message */

/* Switch configuration messages. */
OFPT_FEATURES_REQUEST     = 5,  /* Controller/switch message */
OFPT_FEATURES_REPLY       = 6,  /* Controller/switch message */
OFPT_GET_CONFIG_REQUEST = 7,  /* Controller/switch message */
OFPT_GET_CONFIG_REPLY     = 8,  /* Controller/switch message */
OFPT_SET_CONFIG           = 9,  /* Controller/switch message */

/* Asynchronous messages. */
OFPT_PACKET_IN            = 10, /* Async message */
OFPT_FLOW_REMOVED         = 11, /* Async message */
OFPT_PORT_STATUS          = 12, /* Async message */

/* Controller command messages. */
OFPT_PACKET_OUT           = 13, /* Controller/switch message */
OFPT_FLOW_MOD             = 14, /* Controller/switch message */
OFPT_GROUP_MOD            = 15, /* Controller/switch message */
OFPT_PORT_MOD             = 16, /* Controller/switch message */
OFPT_TABLE_MOD            = 17, /* Controller/switch message */
```

```
/* Multipart messages. */
OFPT_MULTIPART_REQUEST      = 18, /* Controller/switch message */
OFPT_MULTIPART_REPLY        = 19, /* Controller/switch message */

/* Barrier messages. */
OFPT_BARRIER_REQUEST    = 20, /* Controller/switch message */
OFPT_BARRIER_REPLY      = 21, /* Controller/switch message */

/* Queue Configuration messages. */
OFPT_QUEUE_GET_CONFIG_REQUEST = 22,  /* Controller/switch message */
OFPT_QUEUE_GET_CONFIG_REPLY   = 23,  /* Controller/switch message */

/* Controller role change request messages. */
OFPT_ROLE_REQUEST         = 24, /* Controller/switch message */
OFPT_ROLE_REPLY           = 25, /* Controller/switch message */

/* Asynchronous message configuration. */
OFPT_GET_ASYNC_REQUEST  = 26, /* Controller/switch message */
OFPT_GET_ASYNC_REPLY    = 27, /* Controller/switch message */
OFPT_SET_ASYNC          = 28, /* Controller/switch message */

/* Meters and rate limiters configuration messages. */
OFPT_METER_MOD          = 29, /* Controller/switch message */
```

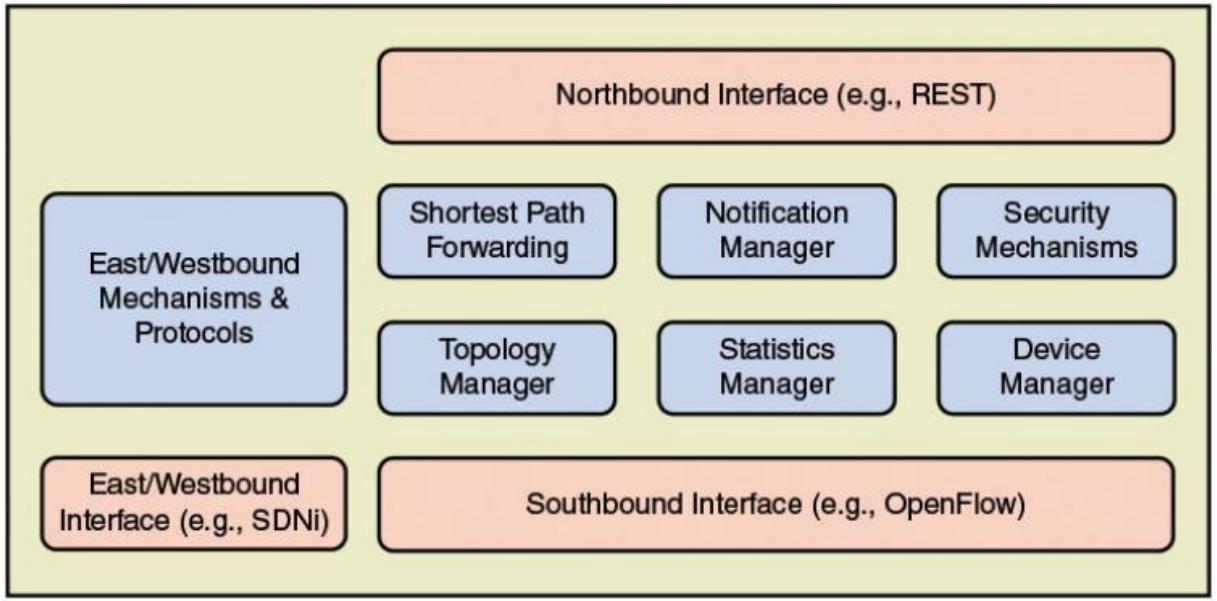
❑ Protocole OpenFlow : échanges Openflow entre contrôleur et commutateur



❑ Plan de contrôle

La couche de contrôle SDN fait correspondre les demandes de services de la couche application en commandes et directives spécifiques aux switches du plan de données. Le plan de contrôle fournit aux applications des informations sur la topologie et l'activité du plan de données. Il est mise en œuvre sous la forme d'un ou d'un ensemble de serveurs coopérants appelés ou contrôleurs SDN. Ces derniers fournis les fonctions suivantes :

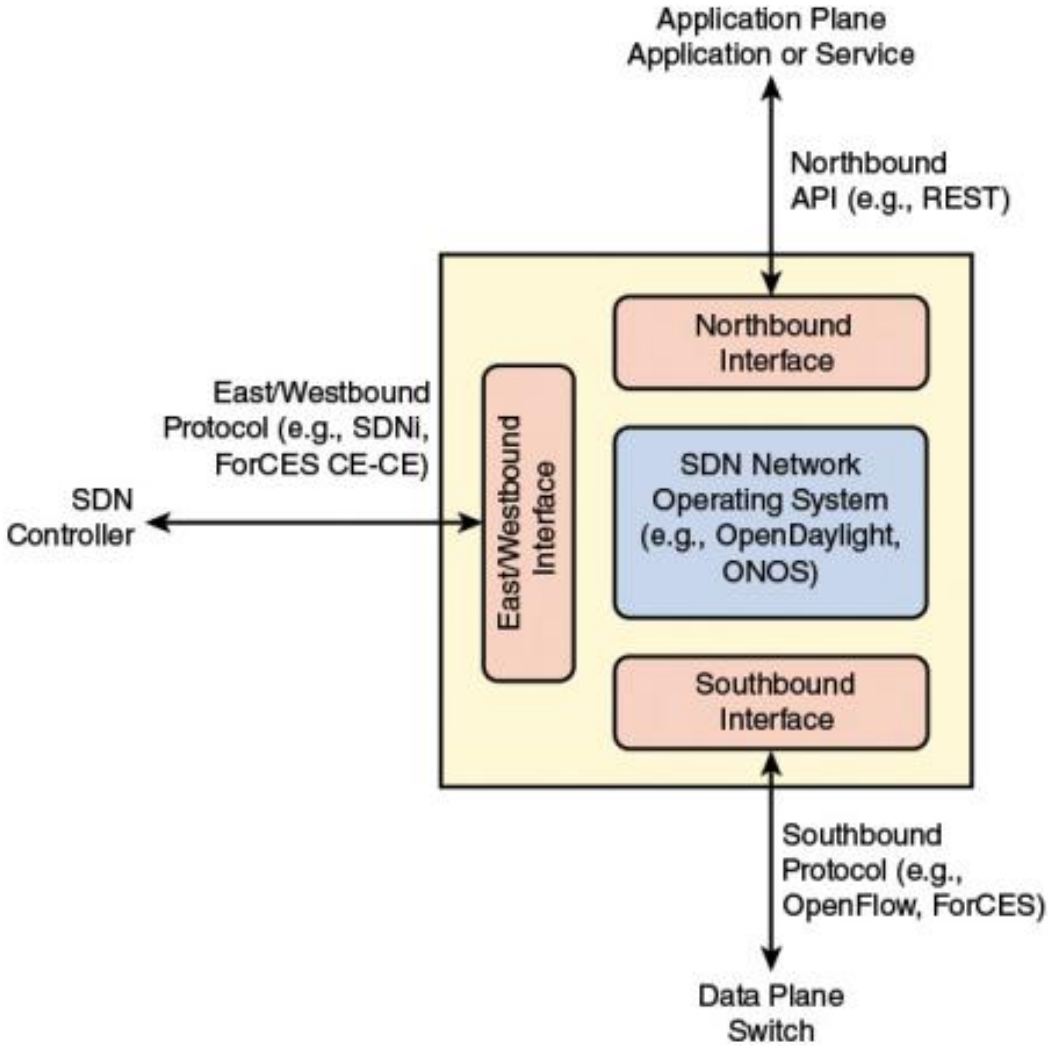
- Shortest path forwarding : routes établies à partir d'informations recueillies auprès des switches SDN.
- Notification manager : reçoit, traite et transmet à une application ces événements.
- Security mechanisms : fournit l'isolation et la sécurité entre les applications et les services.
- Topology manager : construit et maintient les informations de topologie d'interconnexion des switches.
- Statistics manager : collecte des données sur le trafic à travers les commutateurs.
- Device manager : configure les paramètres et les attributs du switch et gère les tables de flux.



❑ Southbound Interface

L'interface Southbound fournit la connexion logique entre le contrôleur SDN et les switches du plan de données. L'API sud la plus couramment mise en œuvre est OpenFlow.

- Open vSwitch Database Management Protocol (OVSDB) : Open vSwitch (OVS) est un swicht virtuel opensource. OVS utilise OpenFlow pour le transfert des messages dans le plan de contrôle. OVSDB est le protocole utilisé pour gérer et configurer les instances OVS.
- Forwarding and Control Element Separation (ForCES): standard IETF qui normalise l'interface entre le plan de contrôle et le plan de données pour les routeurs IP.
- Protocol Oblivious Forwarding (POF): il s'agit d'une amélioration d'OpenFlow qui simplifie la logique du plan de données en un élément de transfert très générique qui ne doit pas nécessairement comprendre la PDU.

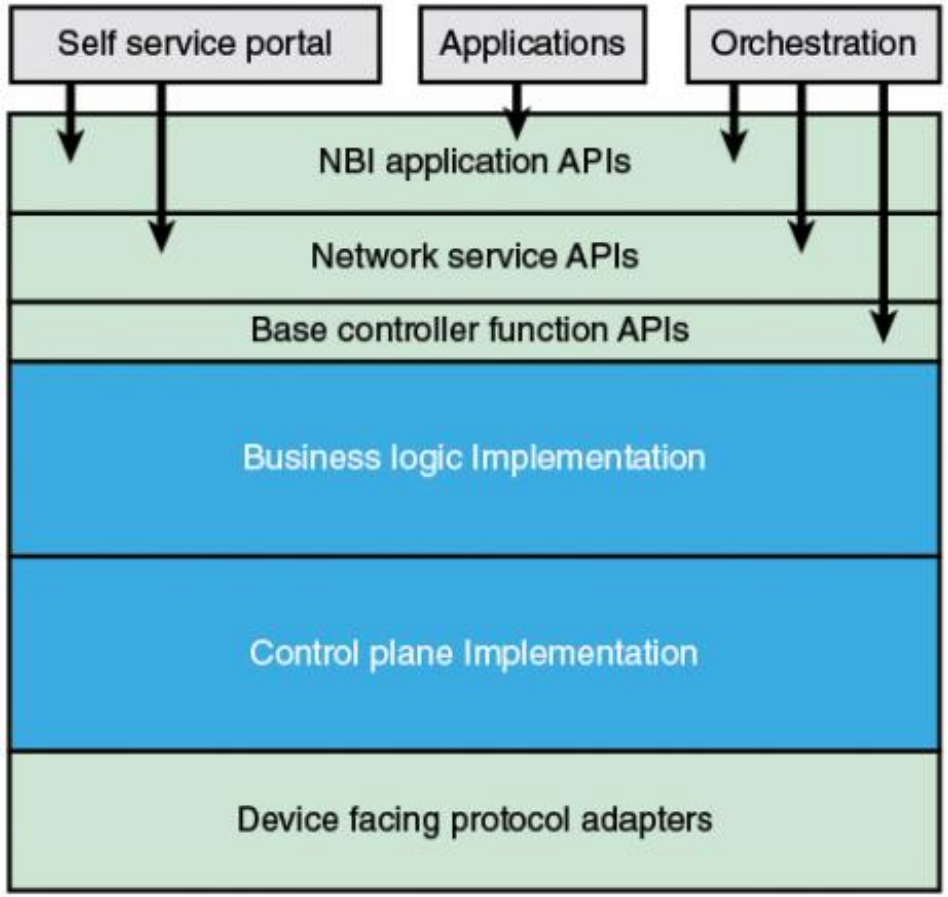


❑ Northbound Interface

L'interface Northbound permet aux applications d'accéder aux fonctions et services du plan de contrôle sans avoir besoin de connaître les détails des commutateurs réseau underlay (physique). L'interface Northbound est plus typiquement considérée une comme une API logicielle plutôt que comme un protocole.

- Base controller function APIs : ces API exposent les fonctions de base du contrôleur et sont utilisées par les développeurs pour créer des services de réseau.
- Network service APIs : ces API exposent les services de réseau au nord.
- Northbound interface application APIs : ces API exposent des services liés aux applications qui sont construits au-dessus des services de réseau.

Un style d'architecture utilisé pour définir les API North est le REST (REpresentational State Transfer).

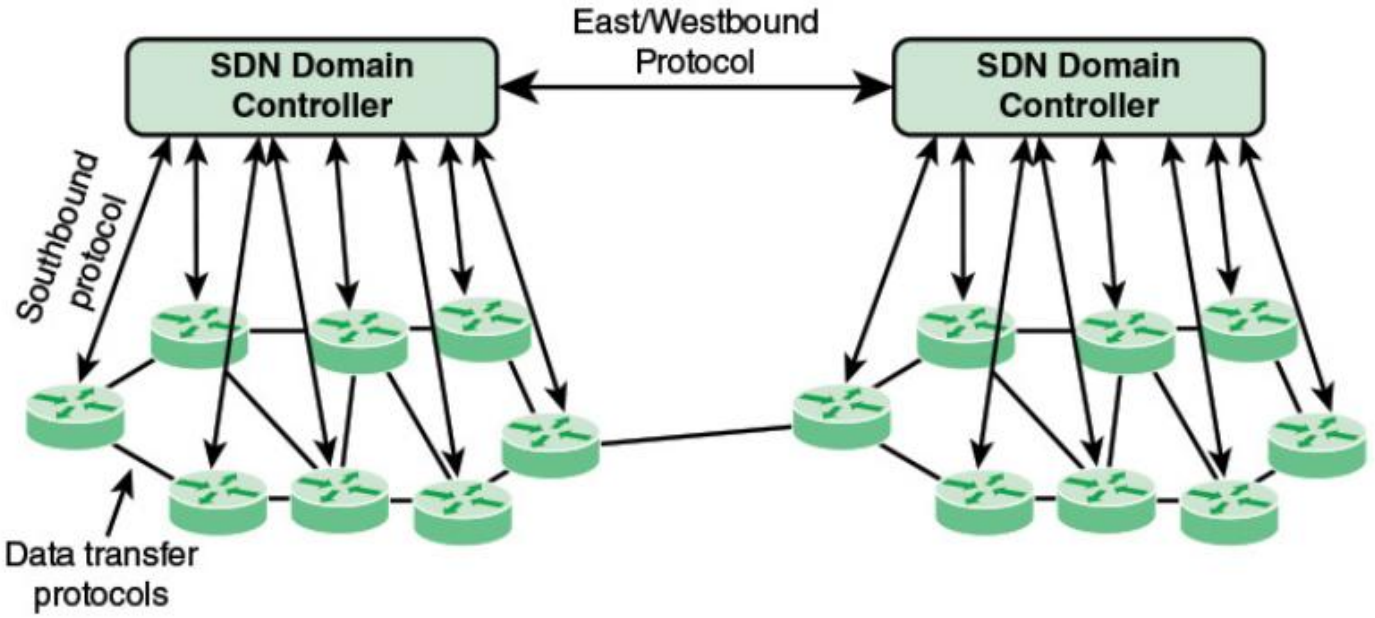


❑ Contrôleurs SDN distribués

Dans un grand réseau d'entreprise, le déploiement d'un seul contrôleur pour gérer tous les équipements du réseau s'avérerait peu maniable ou indésirable. Un scénario plus probable est que l'opérateur d'une grande entreprise ou d'un réseau d'opérateurs divise le réseau en un certain nombre de domaines SDN ne se chevauchant pas, également appelés îlots SDN gérés par des contrôleurs distribués.

Les fonctions associées à l'interface east/westbound pour une architecture distribuée comprennent la maintenance d'une base de données de la topologie et des paramètres du réseau, et des fonctions de surveillance/notification.

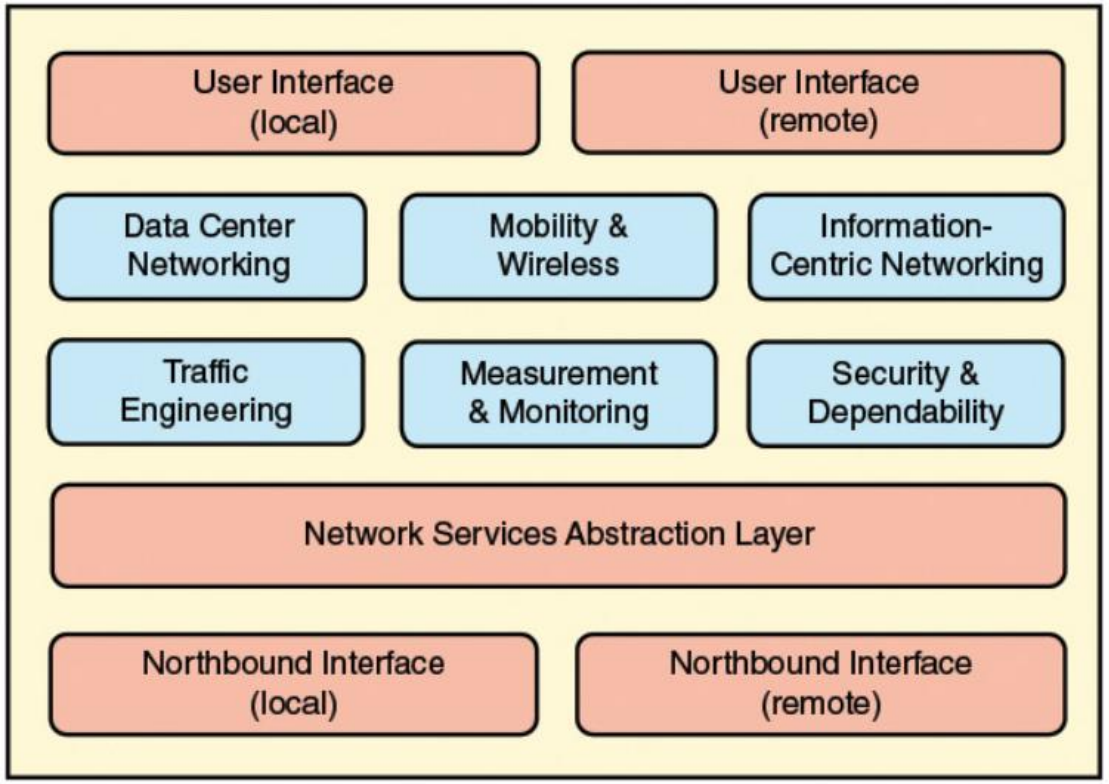
- Scalabilité
- Fiabilité
- Confidentialité
- Déploiement incrémentiel



❑ Plan d'application

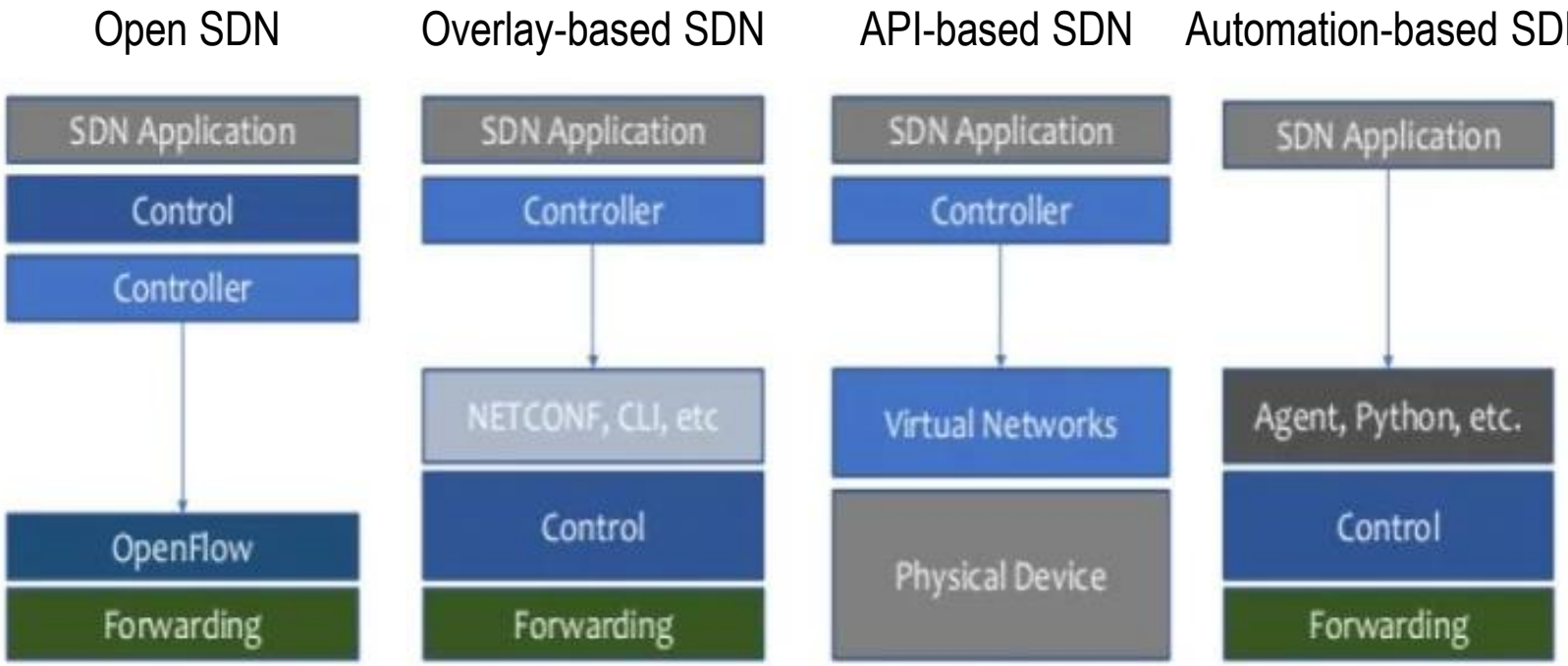
Le plan d'application contient des applications et des services qui définissent, surveillent et contrôlent les ressources et le comportement du réseau. Ces applications interagissent avec le plan de contrôle par l'intermédiaire d'interfaces de contrôle d'application.

Il existe de nombreuses applications réseau qui peuvent être implémentées pour un SDN.



❑ SDN : évolution

Le concept de SDN a évolué depuis son invention initiale et a depuis connu plusieurs implémentations en fonction de la manière dont la couche contrôleur est connectée aux dispositifs SDN.



❑ SDN : évolution

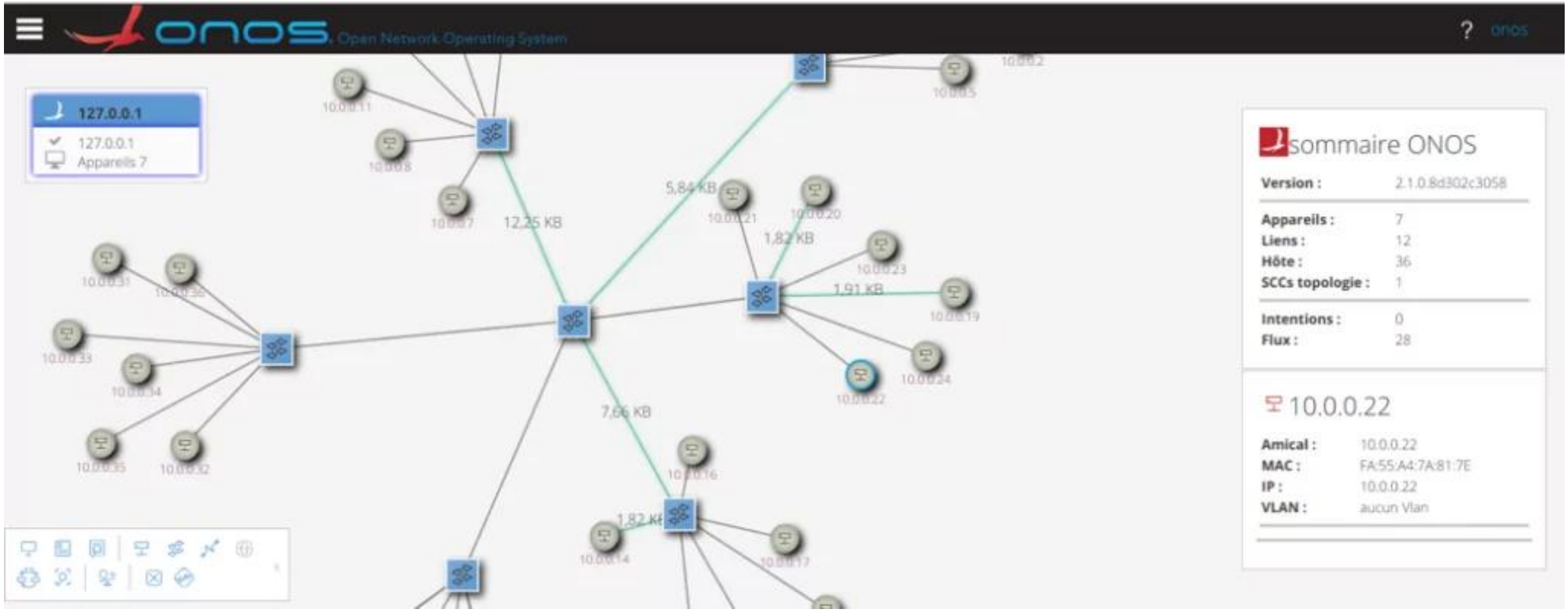
- Open SDN : a un plan de contrôle centralisé et utilise OpenFlow pour son API sud. Il nécessite des stratégies de déploiement évolutives et hybrides. Exemple d'implémentation*: ONOS et BigSwitch*.
- Overlay-based SDN : fonctionne sur un réseau overlay et représente une solution pratique pour résoudre les problèmes de connectivité des datacenters, mais ne traite pas les réseaux physiques sous-jacents. Exemple d'implémentation: *Juniper Contrail, NEC VTN et NSX (VMWare)*.
- API-based SDN (ou unopened SDN): l'accent est mis ici sur la programmation réseau. Il utilise NetConf / OpFlex / SSH pour les API sud afin de permettre la programmation des nœuds de réseau, y compris l'infrastructure matérielle existante. Exemple d'implémentation: *OpenDaylight, APIC-DC, EM et Tail-f (NSO)*.
- Automation-based SDN (or hybrid SDN): couvre les équipements de réseau compatibles avec le SDN ainsi que les équipements existants. L'un de ses objectifs est de réduire le coût des composants réseau. Il utilise des outils d'automatisation (agents, Python, etc.) et des composants génériques supportant plusieurs types de systèmes d'exploitation et tirant parti de l'ONIE (Open Network Install Environment).

❑ Contrôleurs SDN : implémentation

Alors qu'il existe de nombreux contrôleurs SDN, nous comparerons les contrôleurs SDN les plus populaires dans l'industrie et le monde universitaire, notamment :

❑ ONOS

ONOS (Open Network Operating System) est conçu pour être distribué, stable et évolutif, en mettant l'accent sur les réseaux de fournisseurs de services.



❑ ONOS

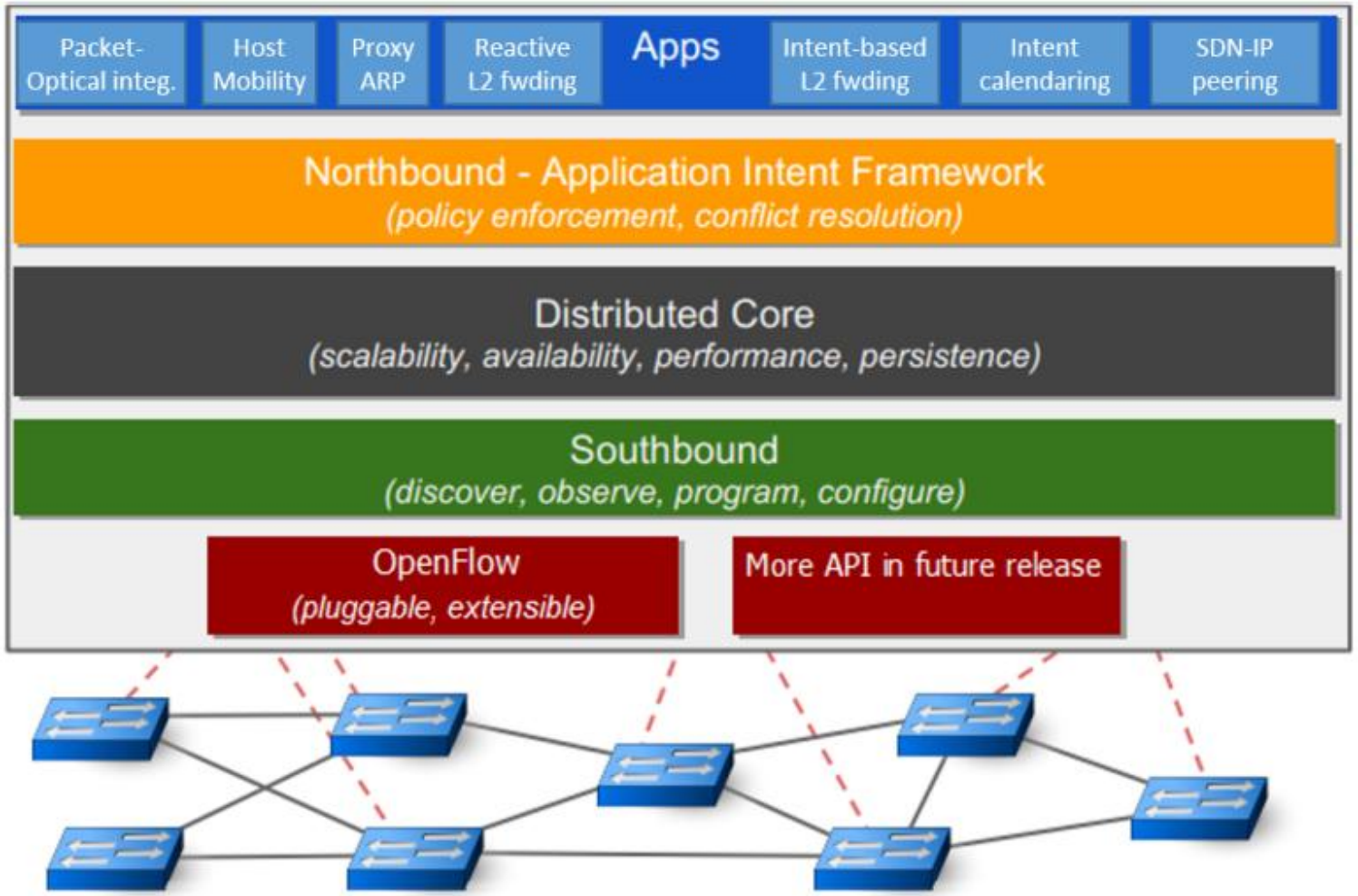
- Interfaces
  - Southbound : OpenFlow, P4, NI ETCONIF, TL1, SNMP, BGR, RESTCONF et PCER
  - Northbound : API gRPC et RESTful
  - GUI: application Web, fournissant une interface visuelle au contrôleur
- Scalabilité
  - ONOS est conçu spécifiquement pour évoluer horizontalement en termes de performances et de géo-redondance dans de petites régions.
- Résilience et tolérance aux pannes
  - Assure la tolérance aux pannes dans le système. En cas de défaillance du nœud maître, un nouveau leader sera sélectionné (se focalise sur la cohérence à terme) pour prendre le contrôle du réseau.

❑ ONOS

- Architecture

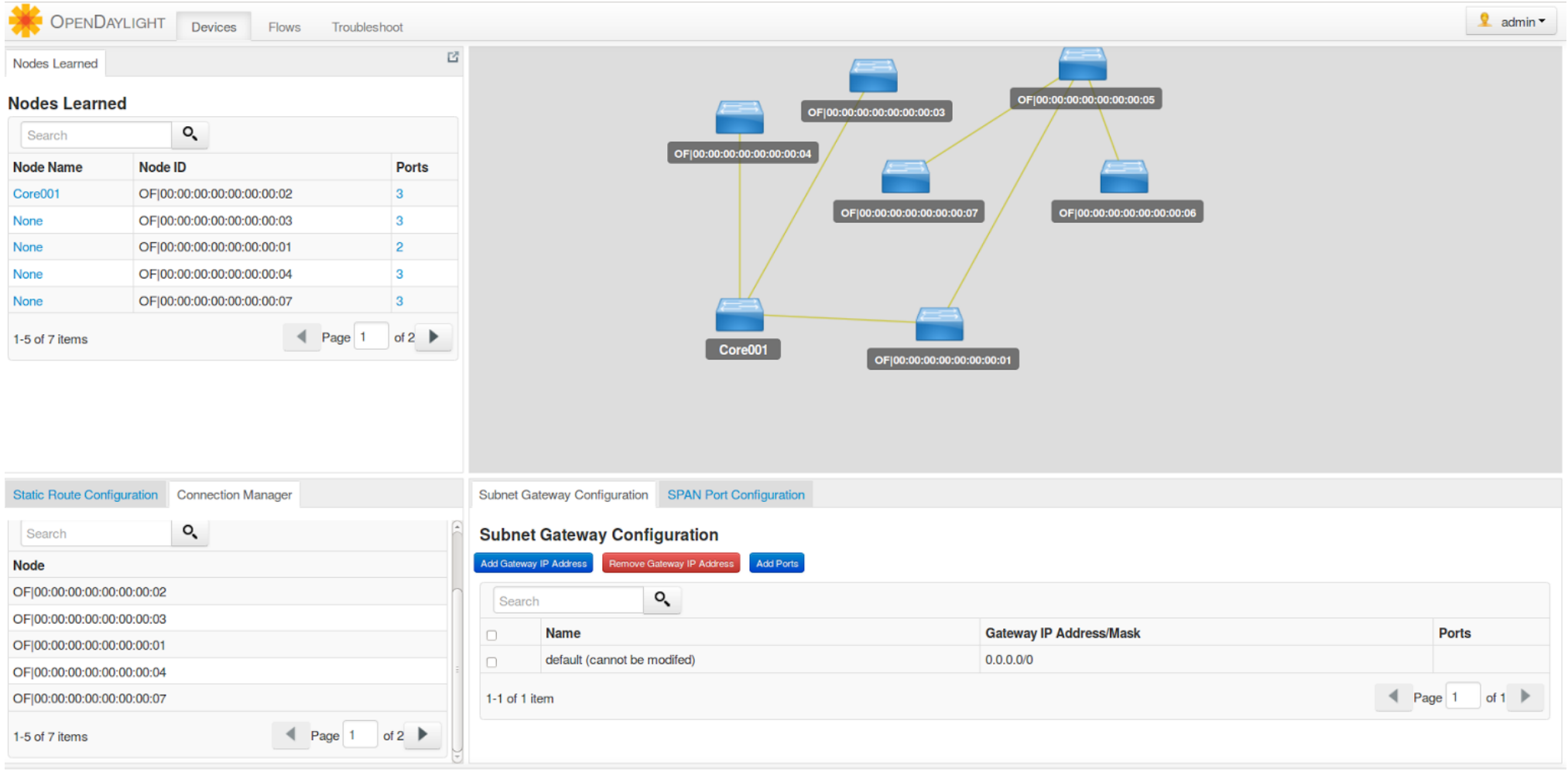
ONOS est conçu comme une architecture à trois niveaux:

- Niveau 1 : comprend les modules liés aux protocoles qui communiquent avec les dispositifs du réseau (Southbound).
- Niveau 2 : comprend le cœur d'ONOS et fournit l'état du réseau sans dépendre d'un protocole particulier.
- Niveau 3 : comprend des applications qui utilisent les informations d'état fournies par le niveau 2.



❑ OpenDayLight

OpenDayLight (ODL) est une plateforme modulaire et ouverte qui permet la personnalisation et l'automatisation du réseau, quelle que soit sa taille. ODL se concentre davantage sur les espaces d'intégration SD-LAN et Cloud.



❑ OpenDayLight

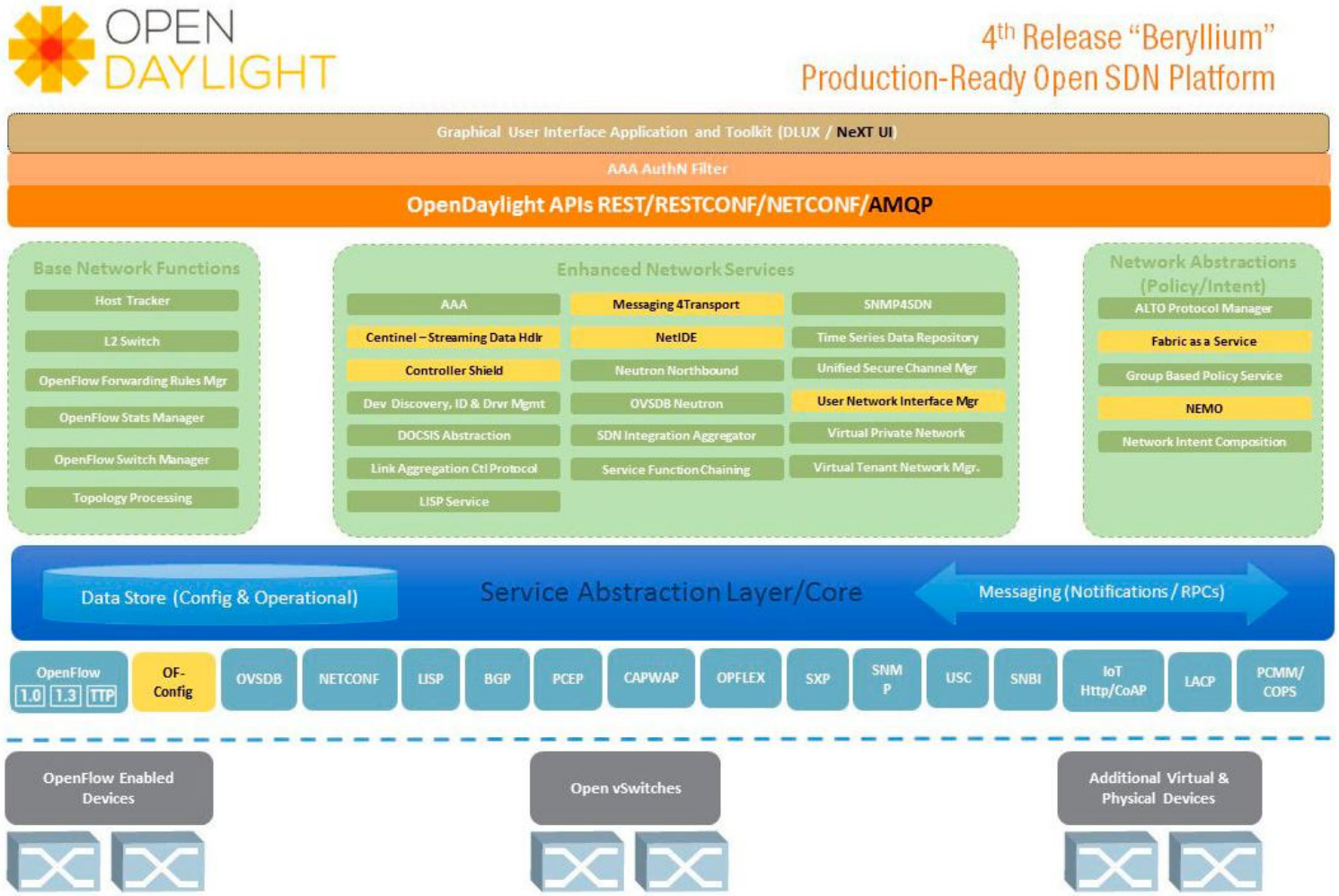
- Interfaces
  - Southbound : OpenFlow, P4, NETCONF, SNMP, BGP, RESTCONF et PCEP
  - Northbound : API gRPC et RESTful
  - GUI: application Web, fournissant une interface visuelle au contrôleur
- Scalabilité
  - ODL prend en charge les déploiements multisites pour une portée géographique, les performances des applications et la tolérance aux pannes.
- Résilience et tolérance aux pannes
  - Mécanisme similaire à ONOS. Mais pour la sélection du leader, L'ODL se focalise sur la haute disponibilité.

❑ OpenDayLight

- Architecture

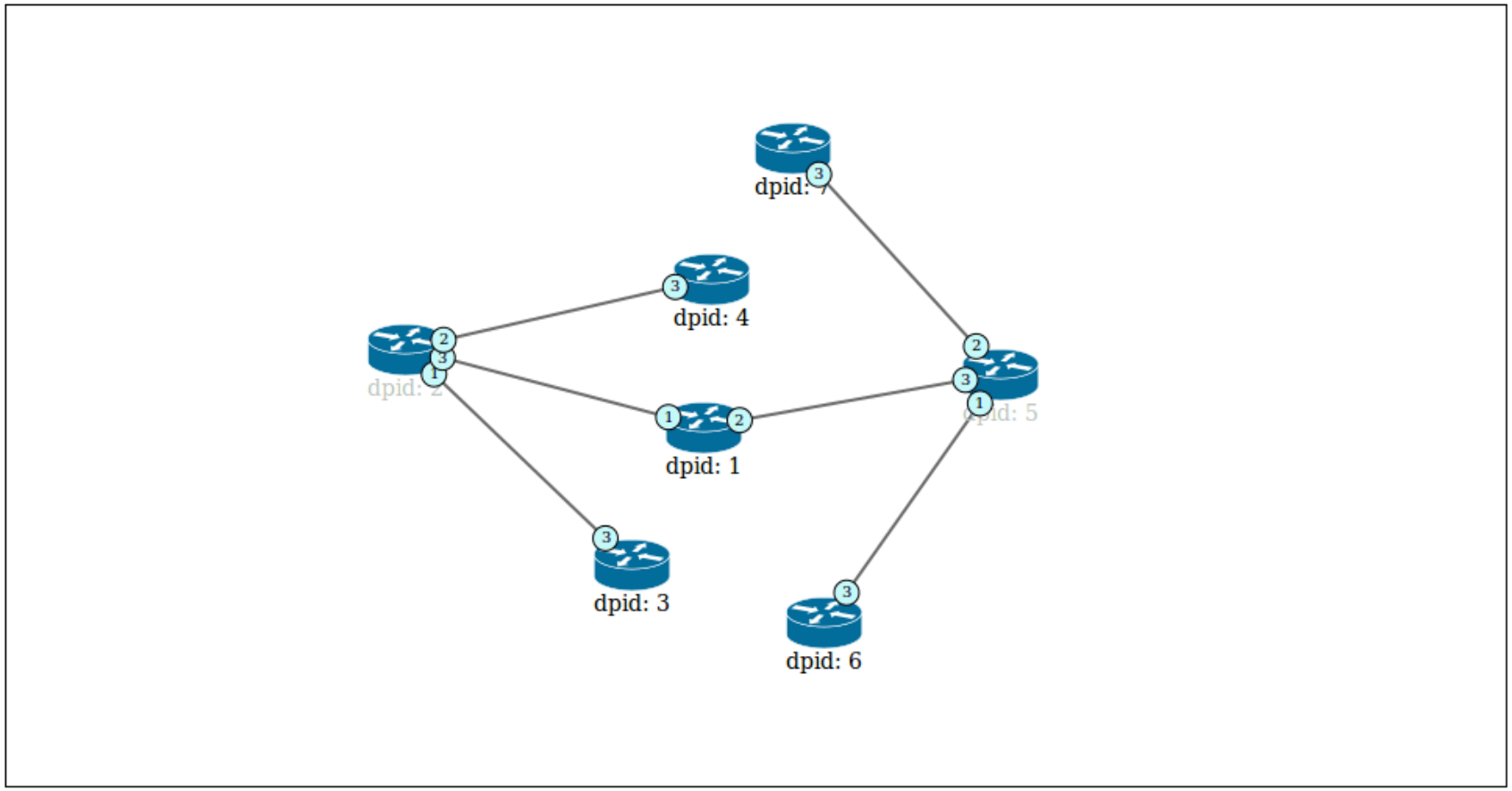
ODL se compose de 3 couches :

- Southbound plugins :pour communiquer avec les périphériques du réseau.
- Core Services: utilisé comme couche d'abstraction de services les autres composants à communiquer avec le contrôleur.
- Northbound interfaces : permet aux opérateurs d'appliquer des politiques de haut niveau aux dispositifs du réseau.



❑ Ryu

Ryu est un contrôleur SDN très différent. Ryu est plutôt un toolbox, avec laquelle la fonctionnalité de contrôleur SDN peut être implémentée. C'est un framework de composants SDN.



❑ Ryu

- Interfaces
  - Southbound : OpenFlow, NETCONF, OF-Config et un support partiel à P4
  - Northbound : API RESTful
  - GUI: application Web fournissant une interface basique pour visualiser la topologie réseau
- Scalabilité
  - Ryu n'a pas de capacité inhérente de mise en cluster et nécessite des outils externes pour partager l'état du réseau et permettre le basculement entre les membres du cluster.
- Résilience et tolérance aux pannes
  - Ryu n'a pas de mécanisme de clustering intégré, mais s'appuie sur des outils externes pour maintenir la disponibilité.

❑ Ryu

- Architecture

Ryu se compose de ces éléments :

- Southbound interface : permet la communication des commutateurs et contrôleurs SDN.
- SDN Framework : prend en charge des applications limitées (découverte de topologie, learning switch…) et des bibliothèques.
- External applications : peut déployer des politiques de réseau sur des plans de données via des API northbound bien définies telles que REST.