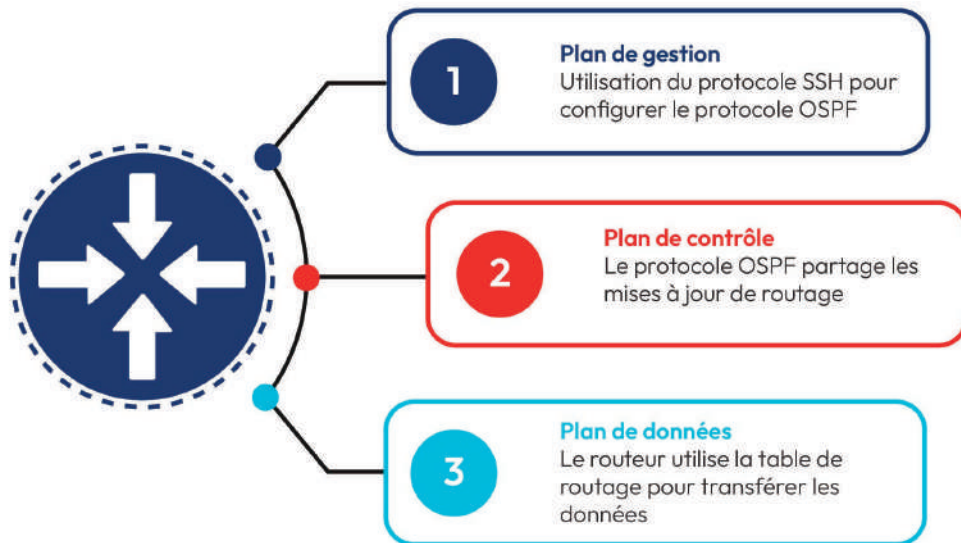


21

L'AUTOMATISATION DES RÉSEAUX

21.1. Introduction à la mise en réseau basée sur un contrôleur :

21.1.1. Plan de gestion, plan de contrôle et plan de données :



Le **plan de données** fait référence aux tâches qu'un périphérique réseau effectue pour transférer un message.

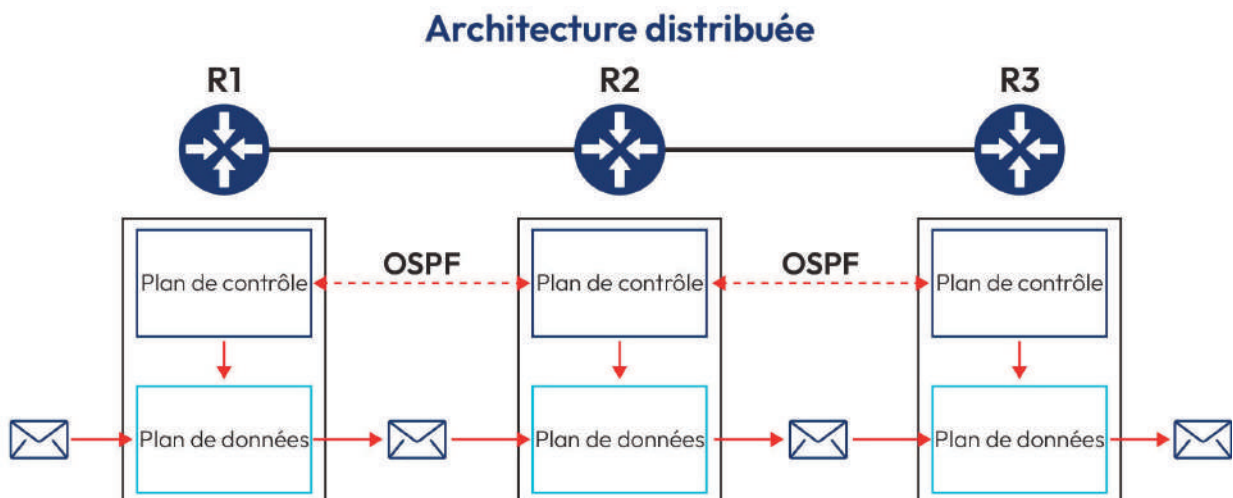
- ➞ Encapsulation et décapsulage des paquets.
- ➞ Recherche de la correspondance dans la table MAC d'un commutateur.
- ➞ Recherche de la correspondance dans une table de routage.
- ➞ Cryptage des données et ajout d'un nouvel en-tête IP (VPN).
- ➞ Modification de l'adresse IP source ou de destination (NAT).
- ➞ Ignorance d'un message en raison d'un filtre (ACL).

Le **plan de contrôle** fait référence à toute action qui contrôle le plan de données :

- ➞ Les protocoles de routage (OSPF, EIGRP, RIP, BGP, etc.).
- ➞ Le protocole ARP (pour IPv4).
- ➞ Le protocole NDP (Pour IPv6).
- ➞ Apprentissage des adresses MAC au niveau des commutateurs.
- ➞ LACP et PAGP.
- ➞ STP, etc.

Le **plan de gestion** comprend des protocoles qui permettent aux ingénieurs réseau de gérer les périphériques : SNMP, Telnet, SSH, FTP, Secure FTP, SYSLOG, etc.

21.1.2. Architecture réseaux traditionnelle distribuée :

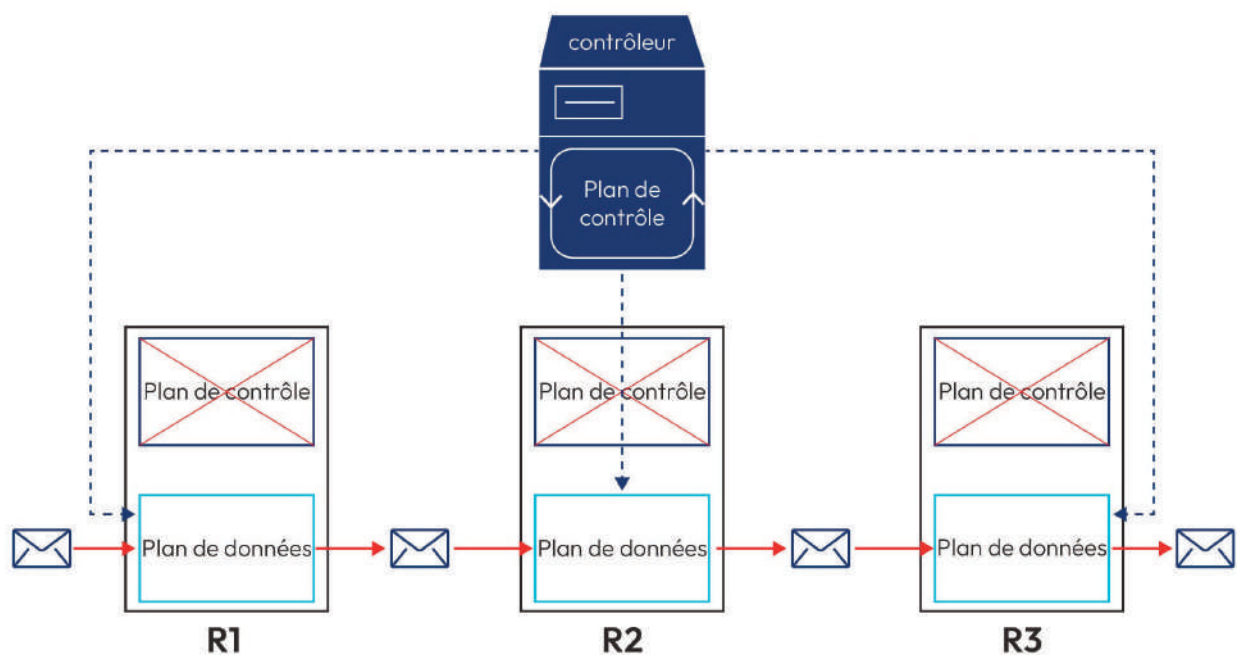


Les périphériques des réseaux traditionnels utilisent à la fois un plan de données distribué et un plan de contrôle distribué.

Ainsi, les périphériques réseau sont surchargés par les actions du plan de contrôle (Mises à jour OSPF, trames BDPDU de STP, ARP, etc.).

21.1.3. Architecture réseau basée sur un contrôleur :

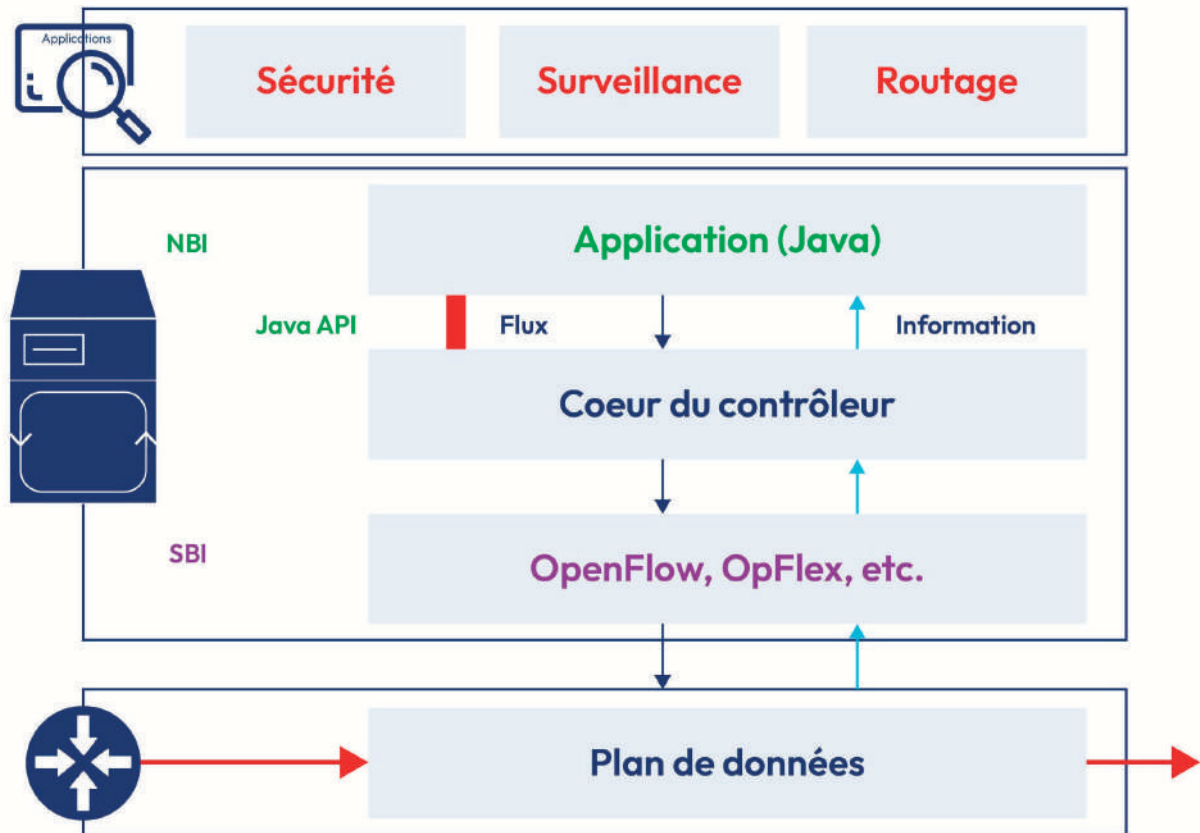
DÉFINITION :



Le contrôleur SDN centralise le plan de contrôle des équipements réseau.

Ainsi, les équipements réseau sont seulement responsables sur les opérations de transfert des données qui se trouvent dans leurs tables de transfert (Table de routage, table MAC, etc.).

ÉLÉMENTS DE L'ARCHITECTURE BASÉE SUR UN CONTRÔLEUR :



Collecte des informations

Le contrôleur recueille des informations :

- ➡ La liste de tous les équipements du réseau.
- ➡ Les capacités de chaque équipement réseau.
- ➡ Les interfaces/ports de chaque équipement.
- ➡ L'état actuel de chaque interface/port.
- ➡ La topologie du réseau.
- ➡ Configuration des équipements de la topologie.

Ajout des actions aux tables de transfert (**Flux**)

Le contrôleur ajoute l'action spécifique dans les tables de transfert des équipements réseau :

- ➞ Table de routage pour les routeurs.
- ➞ Table MAC pour les commutateurs, etc.

L'interface Northbound NBI:

L'utilisateur utilise des **applications** pour communiquer avec le contrôleur à l'aide de l'interface Northbound **NBI** en utilisant des **API** (**A**pplication **P**rogramming **I**nterface).

- ➞ Application de routage.
- ➞ Application de supervision.
- ➞ Application pour la gestion des réseaux sans fil.
- ➞ Application de sécurité.
- ➞ Application pour les réseaux WAN.
- ➞ Application pour la BIGDATA, etc.

L'interface Southbound SBI

Une interface Southbound SBI est une interface qui permet au contrôleur de communiquer avec le programme du périphérique réseau en utilisant :

- ➞ **OpenFlow** (Open Networking Foundation)
- ➞ **OpFlex** de Cisco
- ➞ **CLI** : Telnet, SSH, SNMP et Netconf





21.1.4. Comparaison entre l'architecture distribuée et l'architecture SDN :

	RÉSEAU TRADITIONNEL	RÉSEAU SDN
Budget	Plus de budget	Moins de budgets
Erreurs de configuration	Plus d'erreurs	Moins d'erreurs
Performances	Plus performances	Moins de performances
Automatisation	Pas d'automatisation	Réseau automatisé
Compétences de l'équipe réseau	Moins de compétences	Plus de compétences
Points de défaillance	Un équipement en panne n'influence pas les autres généralement	Si le contrôleur SDN tombe en panne, tout le réseau sera hors service

21.1.5. Les solutions SDN :

OPENFLOW :

OpenFlow est une solution souvent centralisée.

Source	Open Source	
Interface SBI	OpenFlow, netconf, etc.	
Organisation	ONF	
Contrôleur	OpenDayLight Controller	

CISCO ACI :

Cisco ACI est une solution partiellement centralisée.

Interface SBI	Opflex	OpFlex
Organisation	Cisco	
Contrôleur	APIC	APIC

APIC-EM :

APIC-EM est une solution Cisco avec un plan de contrôle non centralisé.

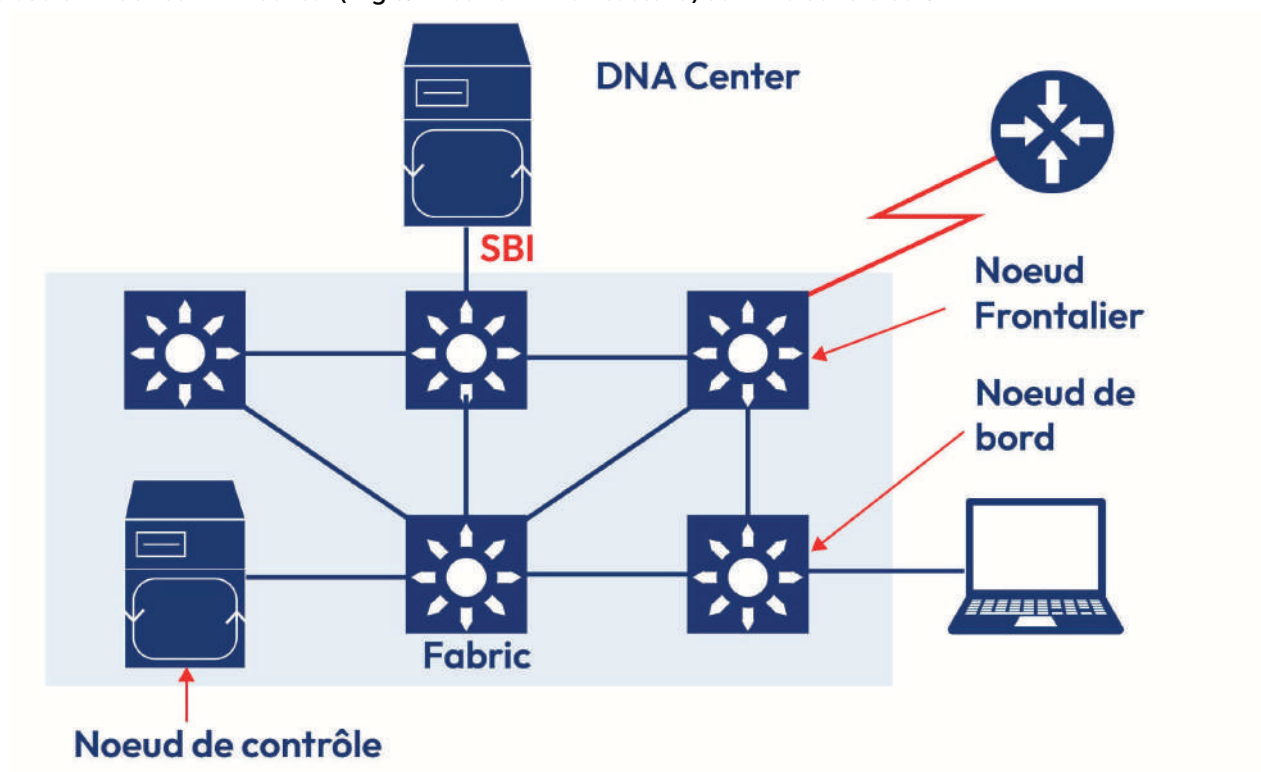
Interface SBI	CLI, SNMP	CLI,SNMP
Organisation	Cisco	
Contrôleur	APIC-EM	APIC-EM

21.2. SD-ACCESS et DNA Center :

21.2.1. Le réseau en tissu (Fabric)

Cisco SDA est une solution qui permet l'automatisation complète du réseau LAN ou WLAN.

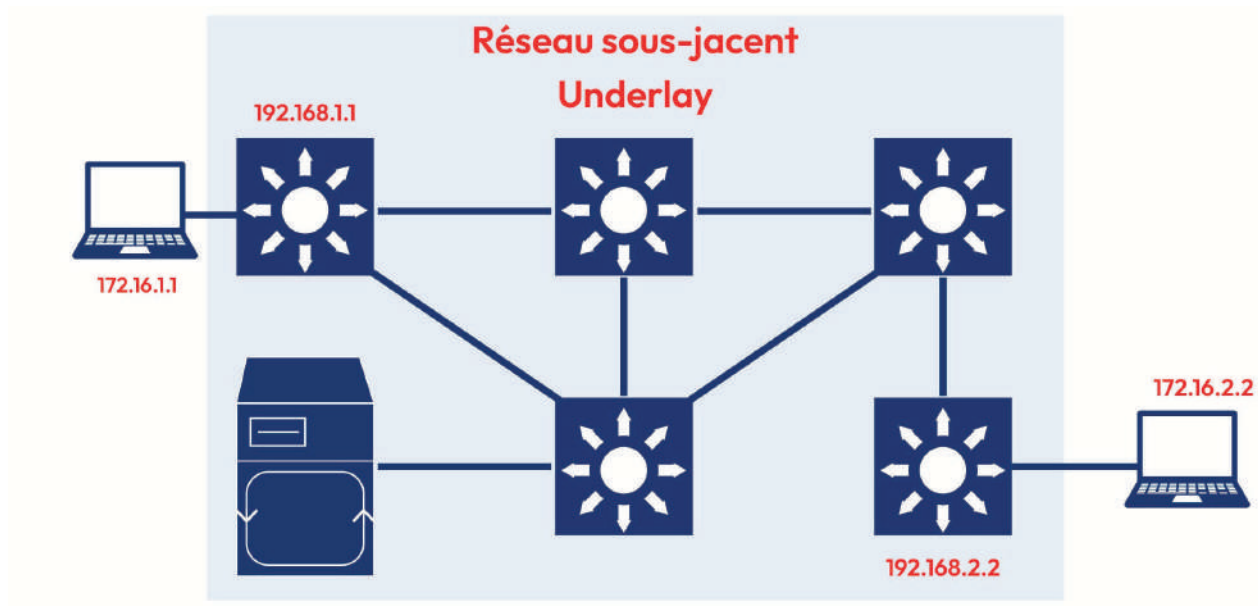
Cisco SDA utilise **DNA** center (**D**igital **N**etwork **A**rchitecture) comme contrôleur.



L'interface Southbound SBI du contrôleur contient :

- ➡ Réseau sous-jacent (Underlay)
- ➡ Réseau superposé (Overlay)
- ➡ Réseau en tissu (Fabric)

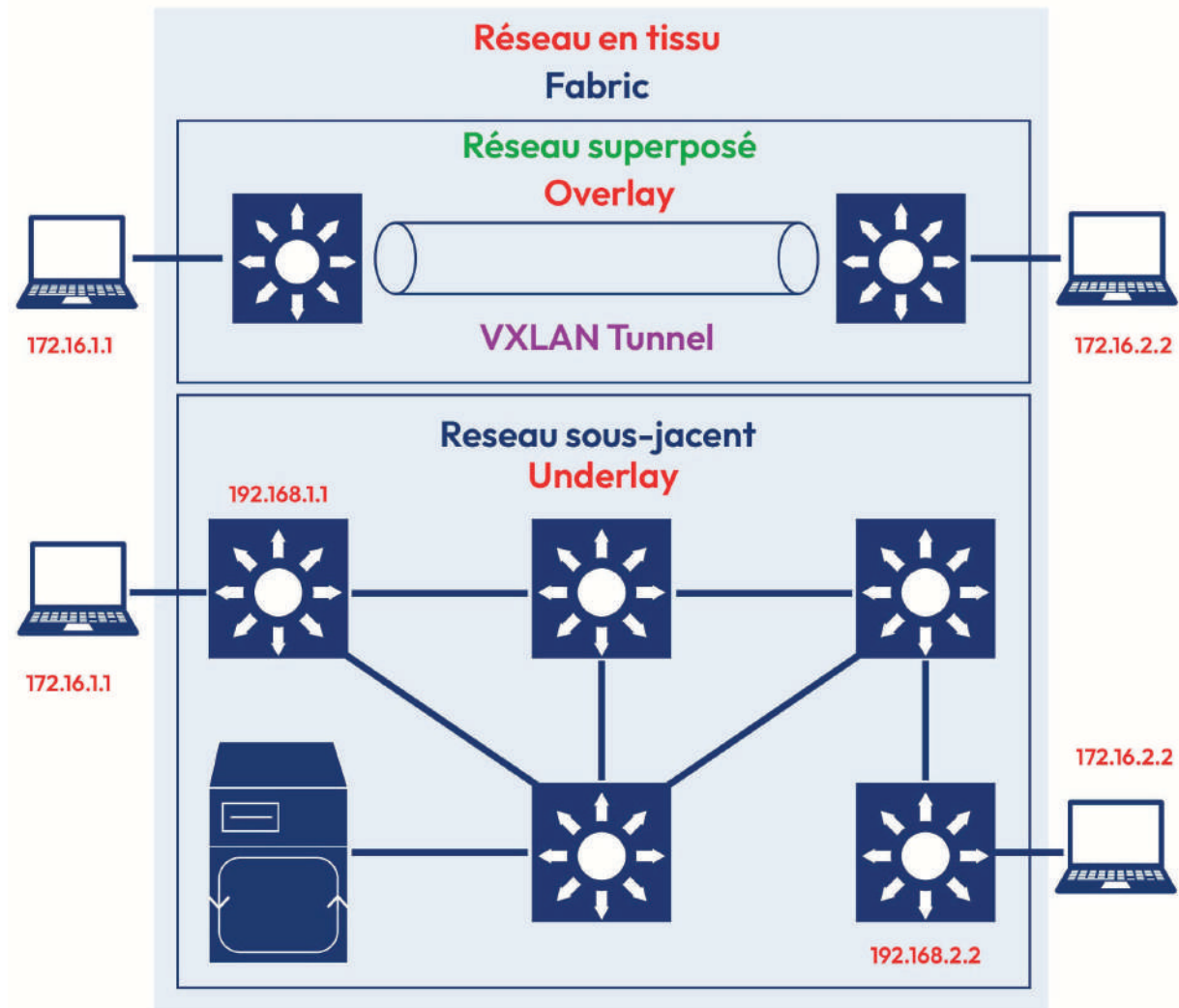
Le réseau sous-jacent (Underlay) est le réseau d'équipements et de connexions à tous les nœuds de la structure.



Le réseau superposé (Overlay) est un réseau logique et virtualisé construit sur la sous-couche physique.

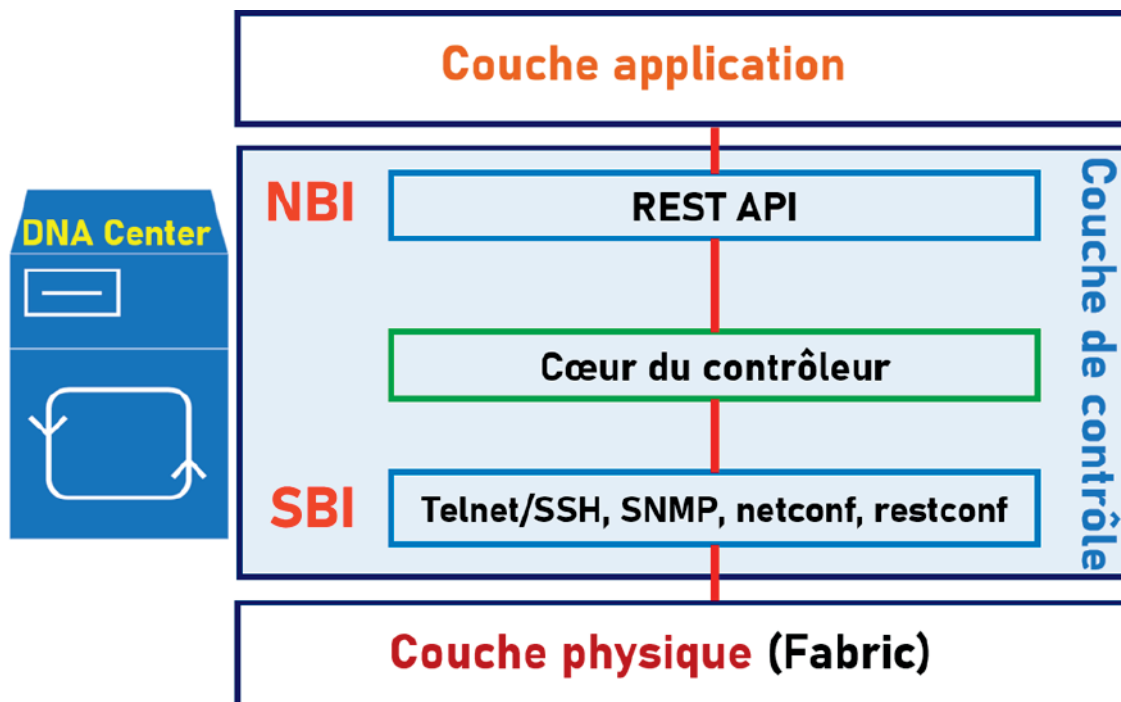


Le réseau en tissu (Fabric) est la combinaison des deux réseaux : Underlay et Overlay.



21.2.2. DNA Center :

- ➔ DNA Center est **le contrôleur** d'un réseau qui utilise **SDA**
 - ➔ DNA Center est **une plate-forme de gestion de réseau** pour les périphériques réseaux traditionnels
 - ➔ DNA Center est **une application logicielle** fournie par Cisco préinstallé sur un équipement Cisco
- DNA Center



21.2.3. Les avantages SDA :

- ➔ **Automatisation** : Elle facilite le déploiement simplifié de nouveaux périphériques réseau, ainsi qu'une gestion cohérente de l'approvisionnement de la configuration réseau LAN et WLAN.
- ➔ **Politique** : SDA utilise une politique basée sur les groupes.
- ➔ **Assurance** : SDA utilise des informations contextuelles pour une résolution rapide de problèmes.
- ➔ **Intégration** : SDA implémente des interfaces ouvertes et programmables pour l'intégration avec des solutions tierces.



21.3. Les formats de données et les API :

21.3.1. Les formats de données :

JSON (JavaScript Object Notation), **YAML** (Yet Another Markup Language), **XML** (eXtensible Markup Language) sont des formats de données :

- ➔ Lisibles par l'être humain.
- ➔ Utilisés pour stocker, transférer et lire des données par des applications.



FORMAT JSON :

Exemple 1 :

Structure hiérarchique et valeurs imbriquées :

- ➔ La figure présente un objet JSON {}
- ➔ L'objet JSON {} contient une donnée sous forme de paires Clé/Valeur
 - La clé = "site"
 - La valeur est un autre objet JSON {}
- ➔ L'objet JSON {} contient deux données sous forme de paires Clé/Valeur :
 - **Donnée 1 :**
 - La clé = "domaine"
 - La valeur = "Formip.com"
 - **Donnée 2 :**
 - La clé = "Admins"
 - La valeur est une liste de deux objets séparés par une virgule [{}, {}]
- ➔ Les deux objets {} contiennent quatre données :

```
{
  "Site": {
    "domaine": "Formip.com",
    "Admins": [
      {
        "Nom": "Martin",
        "Prénom": "Jules",
        "Age": 30,
        "Adresse": {
          "pays": "France",
          "ville": "Paris"
        }
      },
      {
        "Nom": "Soulages",
        "Prénom": "Damien",
        "Age": 35,
        "Adresse": {
          "pays": "France",
          "ville": "Marseille"
        }
      }
    ]
  }
}
```

DONNÉES		VALEURS	
		Objet 1	Objet 2
Clés	Nom	Martin	Soulages
	Prénom	Jules	Damien
	Age	30	35
	Adresse	Objet {}	Objet {}

Les valeurs doivent être un type de données JSON :

Chaîne : "Damien", par exemple

Nombre : 35 par exemple

Tableau : [5,3] par exemple

Booléen : true ou false

Objet : {"Pays": "France", "Ville": "Marseille"}

Exemple 2 :

La sortie de la configuration d'une interface d'un routeur :

```
GigabitEthernet0/0 is up, line protocol is up (connected)
Description: Interface LAN1
Internet address is 172.16.0.2/24
```

La représentation de la configuration en **JSON** :

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet0/0",
    "description": "Interface LAN1",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

FORMAT XML :

Exemple 1 :

```
<?xml version="1.0" encoding="UTF-8" ?>
<Site>
  <domaine>Formip.com</domaine>
  <Admins>
    <Nom>Martin</Nom>
    <prénom>Jules</prénom>
    <Age>30</Age>
    <Adresse>
      <pays>France</pays>
      <ville>Paris</ville>
    </Adresse>
  </Admins>
  <Admins>
    <Nom>Soulages</Nom>
    <prénom>Damien</prénom>
    <Age>35</Age>
    <Adresse>
      <pays>France</pays>
      <ville>Marseille</ville>
    </Adresse>
  </Admins>
</Site>
```

Exemple 2 :

La sortie de la configuration d'une interface d'un routeur :

```
GigabitEthernet0/0 is up, line protocol is up (connected)
Description: Interface LAN1
Internet address is 172.16.0.2/24
```

La représentation de la configuration en XML :

```
<?xml version="1.0" encoding="UTF-8" ?>
<ietf-interfaces:interface>
  <name>GigabitEthernet0/0</name>
  <description>Interface LAN1</description>
  <enabled>true</enabled>
  <ietf-ip:ipv4>
    <address>
      <ip>172.16.0.2</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ietf-ip:ipv4>
</ietf-interfaces:interface>
```

XML utilise des balises <Tag> Valeur </Tag>

FORMAT YAML :

Exemple 1 :

```
Site:
  domaine: Formip.com
Admins:
  - Nom : Martin
    Prénom : Jules
    Age : 30
    Adresse :
      pays : France
      ville : Paris
  - Nom : Soulages
    Prénom : Damien
    Age : 35
    Adresse :
      pays : France
      ville : Marseille
```

- ➔ YAML utilise l'indentation pour la hiérarchisation.
- ➔ YAML utilise les tirets (-) pour les tableaux

Exemple 2 :

La sortie de la configuration d'une interface d'un routeur :

```
GigabitEthernet0/0 is up, line protocol is up (connected)
Description: Interface LAN1
Internet address is 172.16.0.2/24
```

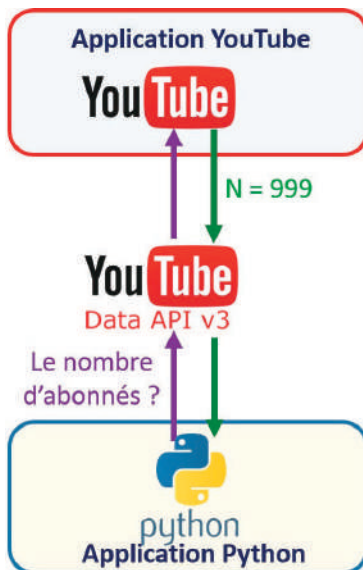
La représentation de la configuration en YAML :

```
ietf-interfaces:interface:
  name: GigabitEthernet0/0
  description: Interface LAN1
  enabled: true
  ietf-ip:ipv4:
    address:
      - ip: 172.16.0.2
      netmask: 255.255.255.0
```

21.3.2. Les API :

DÉFINITION DE L'API :

L'**API** (**A**pplication **P**rogramming **I**nterface) est une interface qui permet aux applications de communiquer entre elles.



Dans l'exemple, l'**application Python** peut demander des informations à l'**application YouTube** via l'API de YouTube (YouTube API v3).

ACCÈS AUX API :

Selon la possibilité d'accès aux API, il existe 3 types :

API	DESCRIPTION
API ouverte/publique	C'est une API accessible au public.
API interne/privée	C'est une API utilisée au sein d'une organisation.
API partenaire	C'est une API entre une entreprise et ses partenaires.

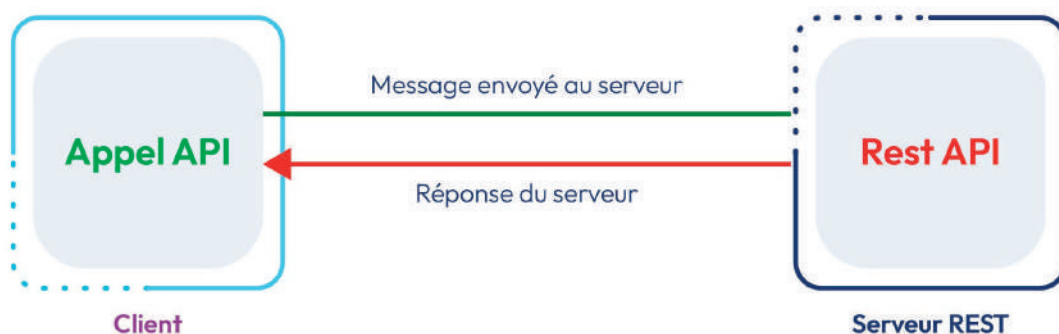
TYPES D'API DE SERVICE WEB :

CARACTÉRISTIQUES	SOAP	REST	XML-RPC	JSON-RPC
Format de données	XML	JSON, XML, YAML et autres	XML	JSON
Avantages	Bien établi	Formatage flexible et le plus largement utilisé	Bien établi, simplicité	Simplicité

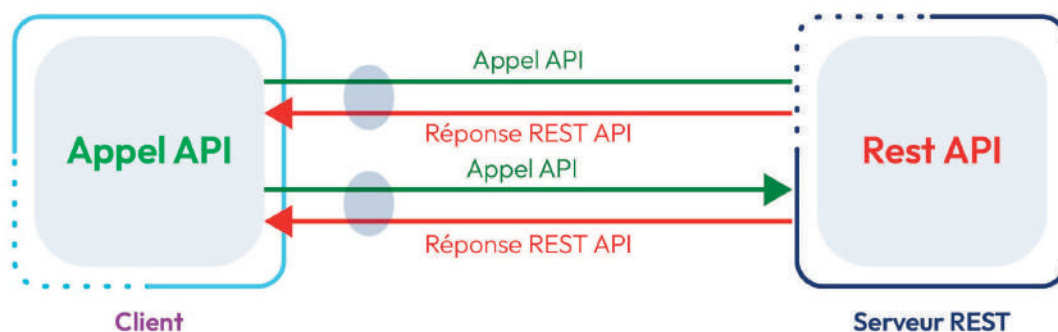
REST API :

Fonctionnalités de REST API :

REST API utilise l'architecture Client/serveur :

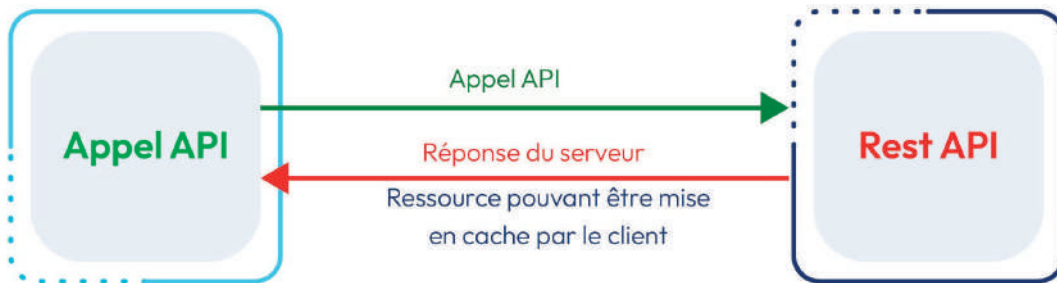


REST API utilise des opérations sans état :



Chaque appel et réponse d'API n'utilise aucun autre historique pris en compte lors du traitement de la demande.

Les clients peuvent mettre en cache les réponses pour améliorer les performances :

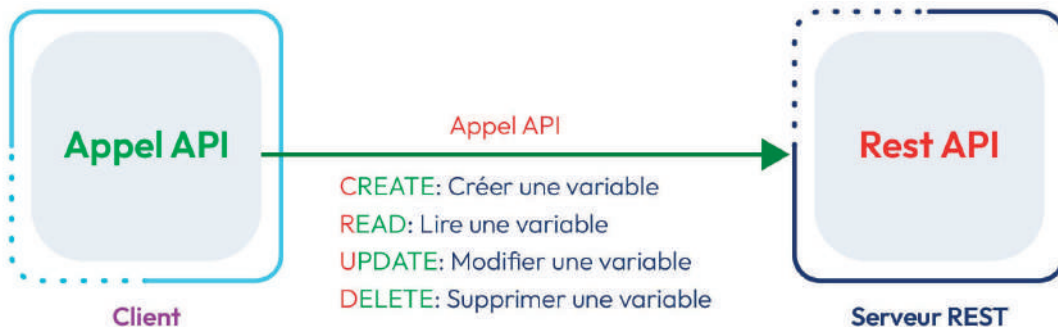


Implémentation de REST API :

REST API est implémentée à l'aide de http et elle se base sur les aspects suivants pour la collection des ressources :

- ➔ Opérations prises en charge (POST, GET, etc.)
- ➔ Identificateur de la ressource uniforme (URI)
- ➔ Format de données (JSON, YAML, XML ou autre)

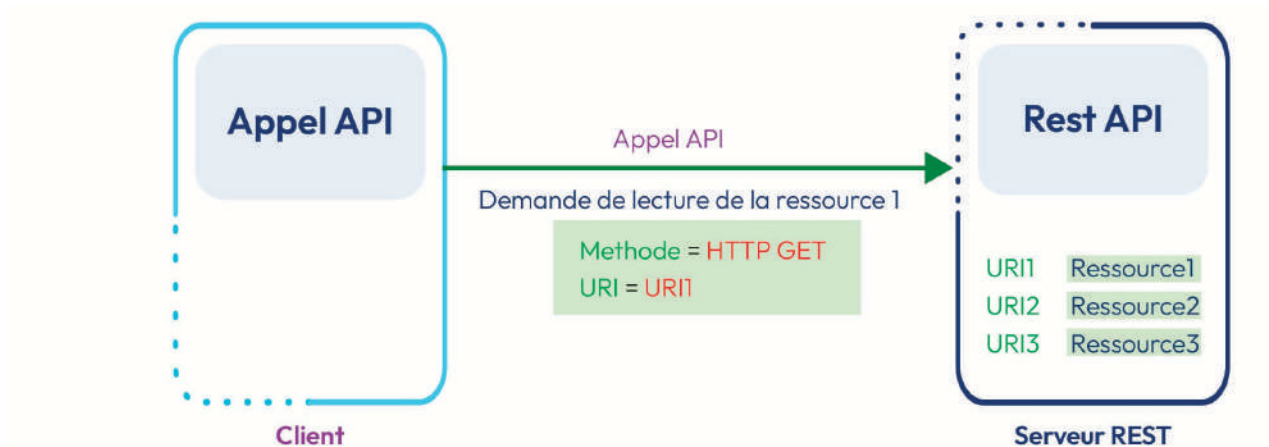
Les opérations REST API (CRUD):



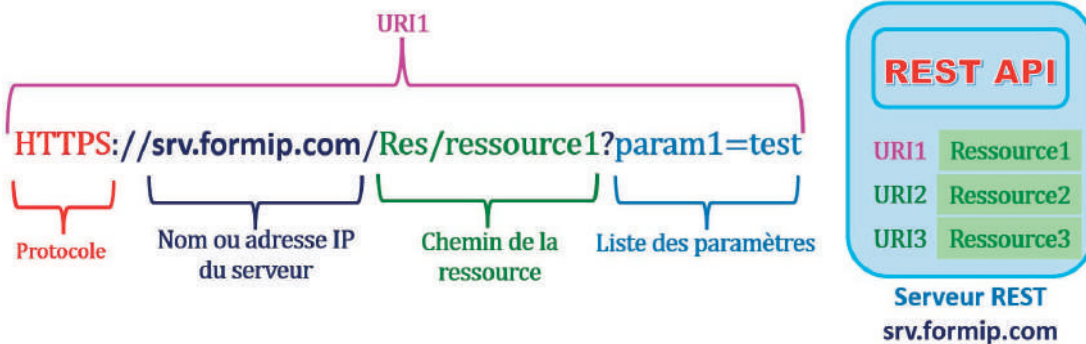
La comparaison entre les opérations CRUD et les verbes REST API :

ACTION	TERME CRUD	VERBE REST HTTP
Créer de nouvelles variables	CREATE	POST
Lire la valeur des variables	READ	GET
Modifier les valeurs des variables	UPDATE	PATCH, PUT
Supprimer des variables	DELETE	DELETE

Les identificateurs de ressources URI, URN et URL :

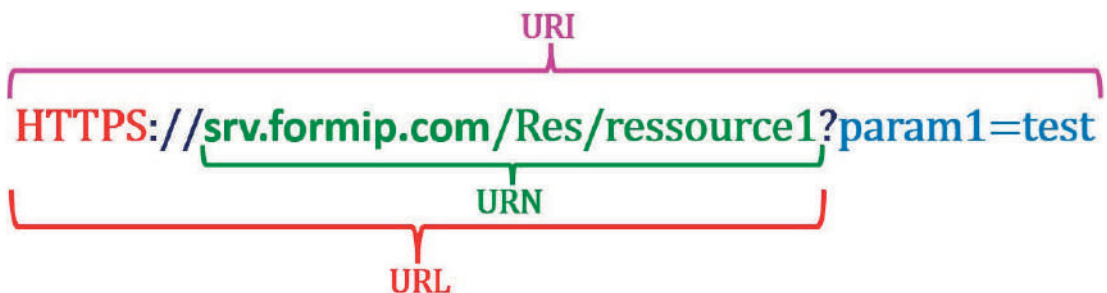


Un **URI** (**U**niform **R**esource **I**dentifier) est une chaîne de caractères qui identifie une ressource spécifique.

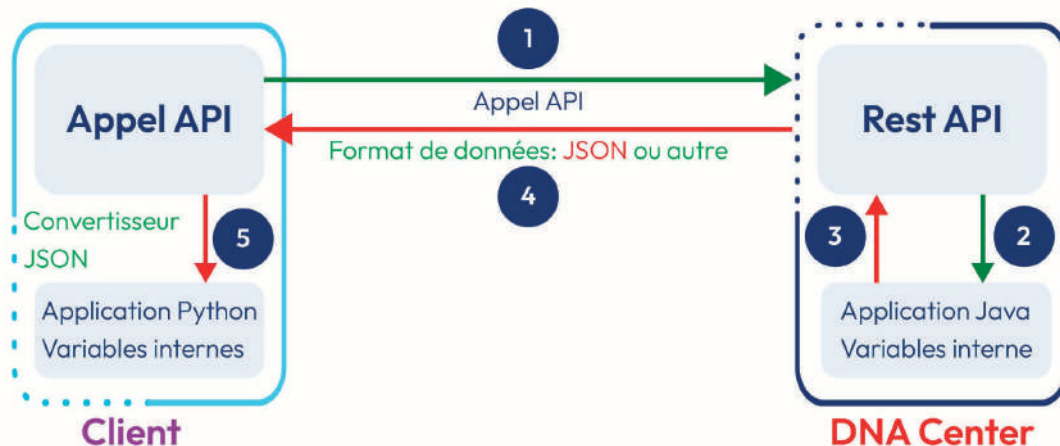


L'URI a deux spécifications :

- ➔ URN (**U**niform **R**esource **N**ame) : Elle identifie la ressource sans faire référence au protocole (Page Web, Image, document, etc.)
- ➔ URL (**U**niform **R**esource **L**ocator) : Elle identifie la ressource en faisant référence au protocole (http, https, ftp, etc.)



Format des données :



Le client envoie un appel API.

L'interface REST API envoie la demande à l'application du DNA Center.

L'application répond à la demande.

L'interface REST API envoie la réponse sous forme de données **JSON**, par exemple.

Le client convertit les données JSON et les envoie à l'application interne du client.

21.4. Les outils de gestion de la configuration :

21.4.1. Définition :

Les outils de gestion de la configuration utilisent l'interface REST API pour automatiser les tâches de configuration de plusieurs périphériques.



21.4.2. Comparaison des outils de gestion :

CARACTÉRISTIQUE	ANSIBLE	CHEF	PUPPET	SALTSTACK
LANGAGE DE PROGRAMMATION	Python + YAML	Ruby	Ruby	Python
AVEC OU SANS AGENT	Sans	Avec	Les deux	Les deux
GESTION DES PÉRIPHÉRIQUES	Tout périphérique peut être un contrôleur	Chef Master	Puppet Master	Salt Master
NOM DE L'ENSEMBLE D'INSTRUCTIONS	Playbook	Cookbook	Manifest	Pillar