SDN

Software Defined Network

- Reti basate sul software

INTRO

La rete IP, soprattutto quando l'utilizzo diventa intenso (come al giorno d'oggi) spesso non è sufficiente. Essa offre soluzioni non al livello delle richieste
Pensiamo ad esempio all'instradamento e a tutti i problemi che possono derivare

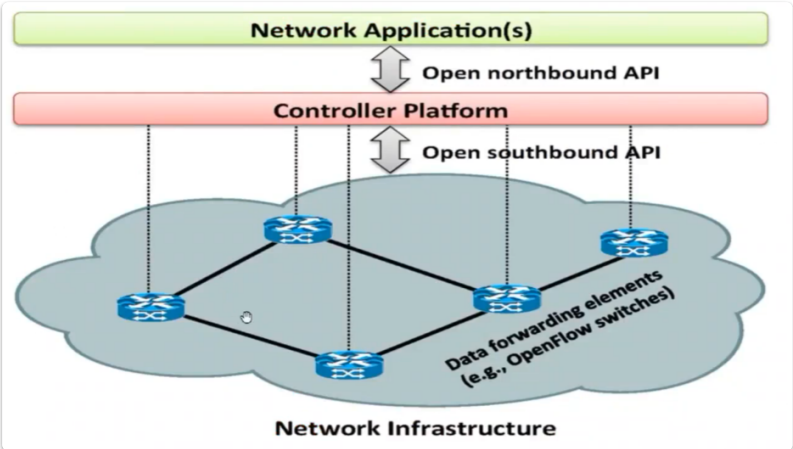
Per avere un sistema più flessibile, meno complesso e che possa avere una certa intelligenza interna per reagire a problemi esterni sono state pensate reti più avanzate
Esse si caratterizzano per gestire in modo separato le parti di:
- Hardware (router, switch) e relative politiche di inoltro
- Controllo (intelligenza per inoltrare in modo opportuno)

Le **SDN** ha rotto la filosofia "verticale" dei *layers* della rete, creando un nuovo paradigma:
Esso introduce astrazioni e modellizzazioni della rete che facilitano la gestione della rete stessa

Fin ora abbiamo visto tecniche di instradamento classiche, come distance vector e link state che avevano come obiettivi quello di popolare un DB locale del router su cui salvare tutte le informazioni utili per la successiva scelta ottimale per l'inoltro di una informazione da una certa sorgente a un destinatario
Nota: il DB locale può essere anche riempito "manualmente", cioè un tecnico addetto può individuare i percorsi che ritiene migliori e implementarli - tuttavia questa procedura è lenta e poco flessibile

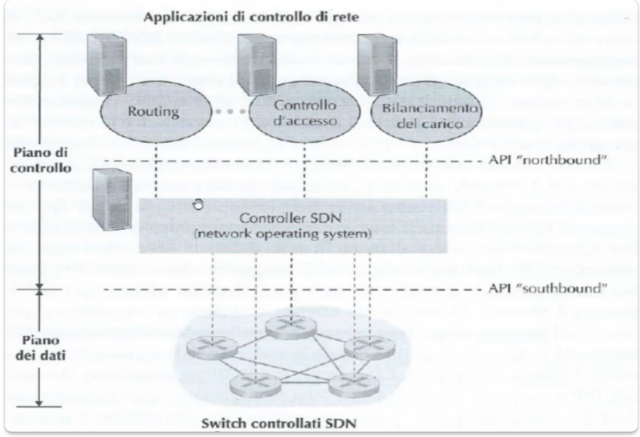
APPROCCIO SDN

Come detto le SDN considera separati per l'utilizzo la parte Hardware da quella di Controllo (intelligenza). Quest'ultima parte viene intesa in primo luogo in modo "logicamente centralizzata" - ovvero abbiamo una parte di controllo (Cloud) paragonabile come al cervello della rete
Per capire meglio la divisione delle parti:



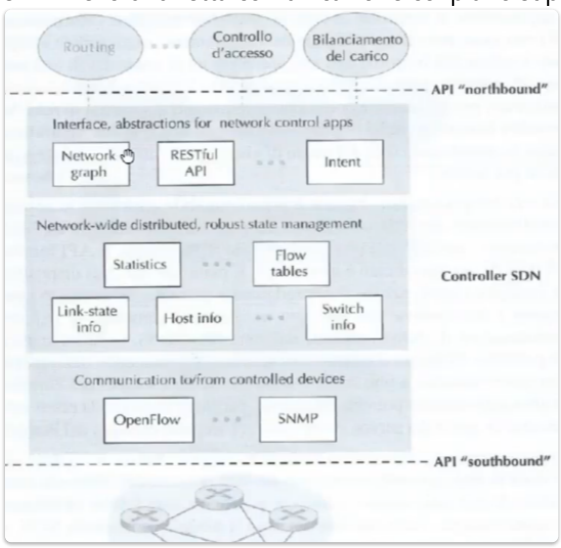
- Network Infrastructure: dove risiedono i dispositivi fisici (Hardware)
  - Con la differenza che a cose normali ogni router ha la propria parte intelligente, cioè il proprio software che si impegna alla fase di calcolo. In questo caso invece il router è *programmabile* e quindi adattabile alle esigenze - attraverso i comandi della parte intelligente della rete, ovvero il Controller Platform
- Controller Platform: come detto è la parte intelligente della rete, idealmente remota (Cloud) che si interfaccia con il livello sottostante mediante una apposita interfaccia detta *southbound* (inferiore) - grazie ad applicazioni software che permettono di trasferire i comandi. Il piano di controllo interagisce a sua volta con quello delle applicazioni
- Network Application(s): Piano applicazioni - interfacciato con il livello di controllo attraverso l'interfaccia *northbound* (superiore). Ogni applicazione comanda il piano inferiore di controllo chiedendo specifiche azioni che sono quindi "trasdotte" (convertite, adattate) per i dispositivi dentro la Network Infrastructure.
  - Ad esempio, può richiedere l'implementazione di politiche di gestione del flusso oppure gestire l'attivazione degli algoritmi di routing

Diagramma alternativo :

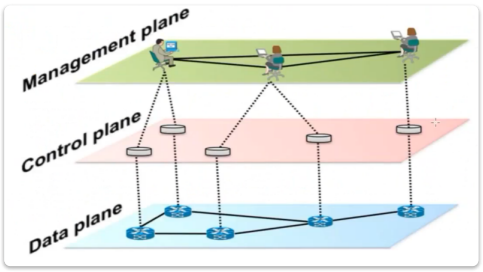


Nel dettaglio, il controller è costituito da 3 livelli:

- **Comunicazione** - per poter trasferire i comandi al piano inferiore (Hardware/Dati) $\rightarrow$ programmare i router e acquisire da essi le informazioni per poter attivare le successive azioni di controllo
- **Memorie**: memorie dove vengono costituiti i dati e gli algoritmi che saranno utilizzati dal piano applicazione
- **Astrazione**: Vengono definiti i vari modelli - livello di diretta comunicazione col piano superiore



In maniera più generale:



INOLTRO GENERALIZZATO SDN

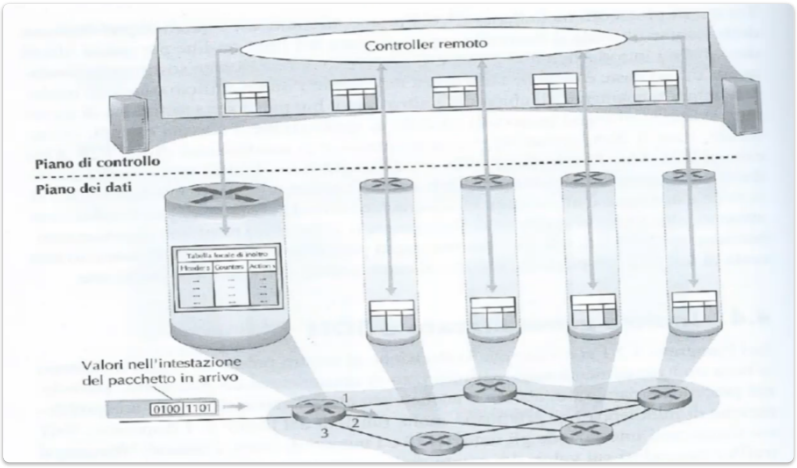
- Sappiamo che gli algoritmi di routing, come l'MPLS, consentono funzionalità aggiuntive tra cui ad esempio distinguere il tipo di traffico da inoltrare, in relazione a specifiche richieste di servizio (attraverso l'uso di etichette). L'utilizzo di MPLS è tuttavia un $middleware$, cioè un livello intermedio che non può essere implementato direttamente al livello IP (causa poca capienza testata datagrammi IP), situato tra il livello collegamento e il livello IP $\rightarrow$ questo porta a fare qualcosa di "fuori logica" rispetto a quanto sappiamo coi livelli, principalmente perché l'instradamento avviene utilizzando informazioni di livello inferiore
- Viceversa il NAT permette l'instradamento andando a leggere i numeri di porta del livello superiore
- Anche il firewall

> *L'SDN consente invece di attuare tutte queste (e altre) possibilità in modo unificato: senza usare cioè protocolli specifici al caso, ma con **applicazioni generali** che vengono adattate secondo le richieste di servizio* 😃

- Le operazioni effettuate nel routing tradizionale sono due:
  - Ricerca indirizzo IP di destinazione (analizzando ed estraendo la maschera di sottorete nel datagramma) $\rightarrow \boxed{match}$
  - Inoltro (invio) effettivo del pacchetto $\rightarrow \boxed{action}$

> *Nel caso di SDN queste operazioni sono **integrate**: un $match$ corrisponde a una $action$*
> *Avviene cioè tutta sullo stesso livello e sullo stesso "programma" di gestione*

ALGORITMO DI ROUTING IN SDN



- Nel routing tradizionale, tutte le azioni rimanevano confinate nel piano Infrastructure - cioè dei dispositivi fisici, che si scambiavano le tabelle per ricercare la soluzione ottima per il percorso sorgente-destinazione (basandosi su un certo algoritmo)
  - Realizzazione distribuita: tutti i router collaborano per le tabelle

> *Nelle SDN gli algoritmi rimangono gli stessi, ma ciascun dispositivo non comunica più con i propri vicini rimanendo a livello infrastructure, ma viceversa comunica con un cloud (controller) remoto (quindi di livello superiore) - dove appunto gli algoritmi risiedono nel piano applicazione: essi eseguono i calcoli da fare e poi vengono quindi trasferite ai router le tabelle di instradamento trovate → realizzazione centralizzata*

- Nel routing tradizionale, ci si basa sull'indirizzo di destinazione per progettare i relativi percorsi sorgente-destinazione per effettuare appunto il routing

> *Invece in SDN è previsto una generalizzazione dell'operazione di routing: ci si basa non più solo sull'indirizzo di destinazione, ma ci sono più funzionalità per aumentare le prestazioni - il routing si effettua analizzando una stringa (successione) di campi*

OPEN-FLOW

Standard interpretabile sia come un *linguaggio di programmazione* (perché serve per costruire le tabella di routing) ma anche come un *protocollo di comunicazione* (perché permette il trasferimento i risultati delle tabelle di routing dal controller (Cloud) alla rete)

- Utilizza TCP porta 6653
- Prevede i seguenti messaggi di comando rivolti al controller
  - Configurazione: programmare tabella dei router
  - Modifica Stato: modificare le tabelle a seconda delle esigenze
  - Leggi Stato: leggere le tabelle
  - Invia Pacchetto: permette l'invio di un pacchetto passando da una certa porta
  - Tabella Rimossa: informa il controller che è stata rimossa una tabella
  - Stato Porta: informa il controller che è cambiato lo stato di una specifica porta

PACKET-SWITCH

- Non si parla più di switch o router ma di $packet\ switch$ (commutatore di pacchetto) - essi devono seguire le azioni che sono contenute nella tabella dei flussi ($flow\ table$) che risiede in OpenFlow

NO RISCONTRO

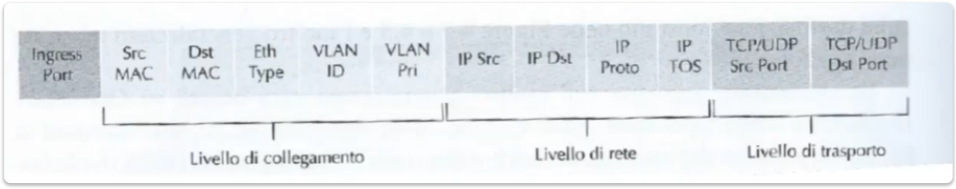
Se un pacchetto non trova riscontro nella relativa tabella dei flussi, allora esso viene inviato a una entità che effettua un controllo più approfondito - tale entità può essere il Controller remoto o un dispositivo preposto più specifico

CONTATORI

- Un qualcosa in più negli switch
  Sono dei dispositivi preposti a monitorare le richieste di inotro dei datagrammi su *determinate* rotte - controllano il traffico della rete
- Le informazioni che raccolgono vengono trasferite al Controller che le elabora e definisce e aggiorna le decisioni in base allo stato della rete
  - *L'insieme delle azioni* costituisce appunto il modo con cui il dispositivo opererà - sono decisi dal Controller

MATCH

Con la logica OpenFlow, l'operazione di $match$ consiste nella analisi dei seguenti campi:



- Non solo il campo sorgente e destinazione (IP Src e IP Dst) ed eventualmente le testate IP (IP Proto e IP TOS) ma utilizza **anche** informazioni *provenienti da altri livelli* (cross-layer):
  - Collegamento (schede di rete MAC, tipo di rete Ethernet etc...)
  - Trasporto (un po' come avevamo già visto col NAT)
  - Porta di ingresso

> *Un $packet\ switch$ ha funzionalità superiori: può funzionare da switch di livello due oppure anche di livello tre o quattro*

- Attualmente OpenFlow ha esteso i campi esaminabili da 12 (in figura, che già sembravano tante) a 41
  - Q: Ne risente la velocità di trasmissione? A: No, perché le velocità son sempre più alte
  - Non se ne aggiungono altre proprio per non risentirne in velocità: $fai\ una\ cosa\ ma\ falla\ bene$

WILDCARD

Si può abbinare a più indirizzi di rete una stessa azione specificando un gruppo di indirizzi in maniera apposita, utilizzando degli asterischi.
Ad esempio:

$$128.119.*.*$$

Trova corrispondenza con tutti i datagrammi che iniziano (primi 16 bit) con 128.119

AZIONI PREVISTE IN OpenFlow

INOLTRO

- Ripetere il datagramma sulla porta di uscita che si ritiene migliore per quella richiesta
- Può essere programmato da un Controller remoto ed eventualmente modificato a seconda dei contesti (per esempio per evitare congestioni)

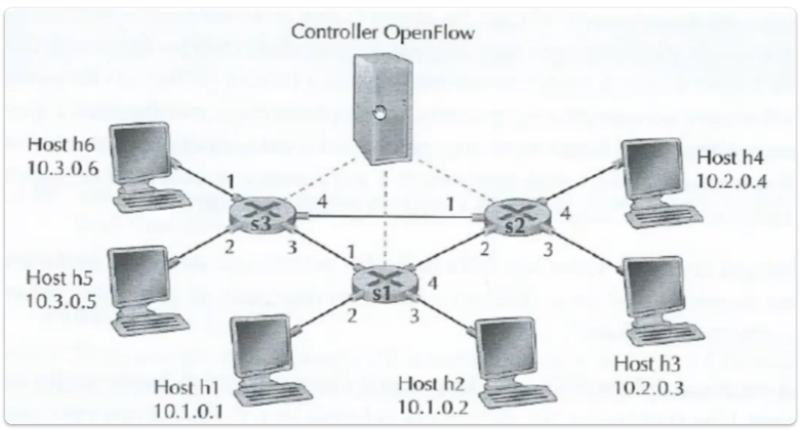
DROPPING (SCARTO)

- Possibilità di eliminare un pacchetto che non trova corrispondenza nelle tabelle

MODIFICA DEI CAMPI

- Possibilità di modificare la testata dei vari datagrammi (ad esempio per effettuare la frammentazione)

ESEMPI MATCH-ACTION IN OpenFlow



ESEMPIO 1

- Il controller è gestito da OpenFlow. Con esso si relaziona con i client
- I client si accordano con OpenFlow

Abbiamo 6 host che sono connessi a 3 router

- Vediamo come sono scritte le tabelle $action$ in relazione a determinate richieste

> *$H5$ e $H6$ devono inviare pacchetti a $H3$ e $H4$*

- i pacchetti arrivano a $S3$
  - Supponiamo che essi non possano seguire la strada diretta $S3 \rightarrow S2$ ma devono passare da $S1$
- Abbiamo due indirizzi di rete che mirano a una stessa azione: ad esempio i pacchetti generati da $H5$ e $H6$ possono arrivare a $H4$

Il router $S1$ ha la seguente tabella $action$:

s1 Flow Table (Example 1)

Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)
...	...

- L'interfaccia di ingresso è la 1 (vedi figura)
- La sorgente è la rete ".3" (generica)
- La destinazione è la rete ".2" (generici) $\rightarrow$ cioè deve arrivare a $S2$
  - Buono mantenere generico perché devo leggere meno campi

A $S2$ invece si associa la seguente tabella

s2 Flow Table (Example 1)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

- Si nota come il router finale "toglie" la wildcard e specifica l'indirizzo preciso di destinazione, a seconda della richiesta

ESEMPIO 2: LOAD BALANCING

Si vogliono inoltrare verso la stessa rete pacchetti provenienti da host della stessa rete *facendoli fare percorsi diversi*

- Ad esempio: $h3 \Rightarrow S2 \rightarrow S1$ e $h4 \Rightarrow S2 \rightarrow S3 \rightarrow S1$ tutti con destinazione $S1 \approx 10.1.*.*$
  Questo non era permesso con il tradizionale inoltro IP!

s2 Flow Table (Example 2)

Match	Action
Ingress port = 3; IP Dst = 10.1.*.*	Forward(2)
Ingress port = 4; IP Dst = 10.1.*.*	Forward(1)

ESEMPIO 3: FIREWALLING

Come terzo esempio si consideri uno scenario di firewall in cui s2 desideri ricevere su qualunque sua interfaccia il traffico inviato da host attaccati a s3.

s2 Flow Table (Example 3)

Match	Action
IP Src = 10.3.*.* IP Dst = 10.2.0.3	Forward(3)
IP Src = 10.3.*.* IP Dst = 10.2.0.4	Forward(4)

Se non ci fossero altre occorrenze nella tabella dei flussi in s2, solo il traffico da 10.3.*.* verrebbe inoltrato agli host attaccati a s2

CONSIDERAZIONI su SDN

L'introduzione "massiccia" di un software nell'ambito della rete ha portato nuovi scenari, permettendo la creazione di tecnologie come $Edge\ Computing$, $Fog\ Computing$ e in generale l'$Industria\ 4.0$ (automatizzazione processi industriali)

- Ha portato a parlare di $ecosistema$ di dispositivi

LINK STATE IN SDN

- Ci chiediamo come viene implementato l'algoritmo di $Dijkstra$ (link-state) in SDN

Supponiamo che l'algoritmo sia già stato costruito sulla base delle informazioni che i vari router hanno raccolto

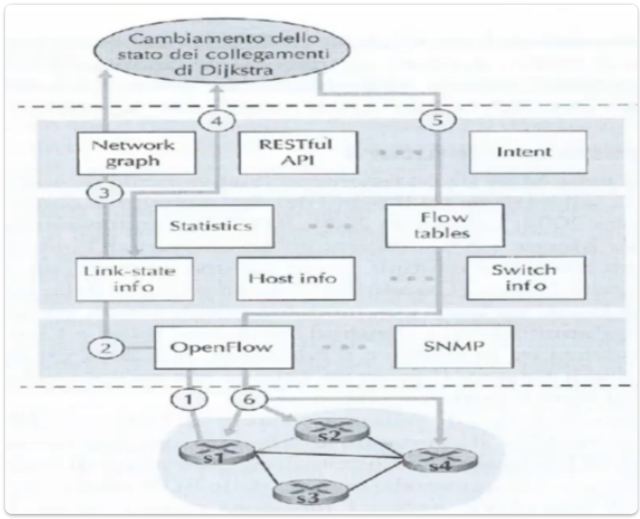
- Vengono quindi comunicate ai vari router le tabelle di instradamento

Ipotizziamo che accada un *evento imprevisto*: il link $S1$ - $S2$ si interrompe

$$S1 \not\leftrightarrow S2$$

- Nella modalità classica, $S1$ doveva notificare a tutti i router della rete questa situazione - a quel punto allora i router dovevano aggiornare le proprie tabelle
  - Dopo un certo tempo (e vari possibili problemi) tutti i router avrebbero capito il problema

> *In SDN invece, ogni router comunica l'eventuale cambio di stato di un collegamento direttamente con il suo controller. In questo modo la riprogrammazione delle tabelle è più veloce ed efficiente*
> *In particolare: il router si interfaccia con OpenFlow, che capisce la richiesta e la trasferisce al dispositivo del "livello" controller superiore che controlla lo stato dei collegamenti. A sua volta si interfaccia con la parte di rete effettiva che definisce i percorsi per arrivare fino all'algoritmo. In esso si inseriscono le modifiche che si vuole e lui procede quindi a definire la soluzione (interfacciandosi eventualmente più volte con il Link-State info (4)). Una volta trovata la soluzione definitiva vengono aggiornate le tabelle di flusso che saranno comunicate a OverFlow che procede a inviarle ai router che sono influenzati dai cambiamenti apportati (ad esempio viene detto a $S1$ e $S2$ che dovranno mandare i loro pacchetti passando da $S4$ [...])*



VIRTUALIZZZIONE FUNZIONI DI RETE

- È la funzionalità principale e più influente di SDN

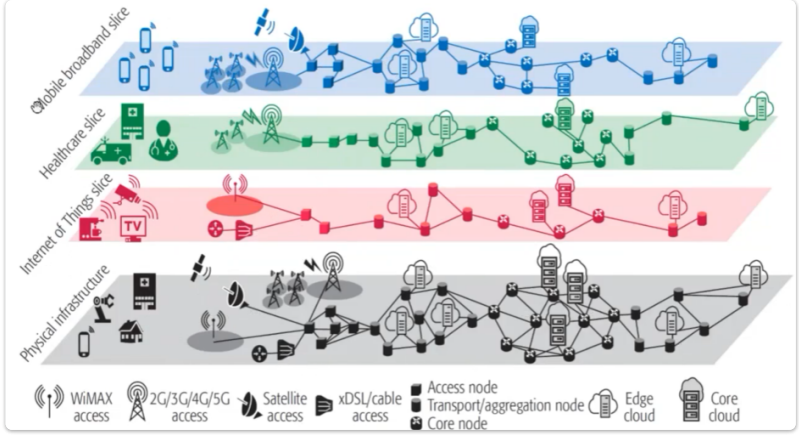
Le funzioni di rete sono virtualizzate (non risiedono più in determinati postazioni legati ad hardware specifici), ovvero possono essere utilizzare in più modalità e su hardware diversi

- Utilizzano $virtual\ machines$ che risiedono nei dispositivi
  - Maggiore flessibilità 😃
  - Si possono far alloggiare più applicazioni SW (anche S.O.) su uno stesso HW, partizionando quest'ultimo
    - Se le funzioni di rete sono complesse, si costruisce una *catena* di virtual machine che condividono tra loro le funzioni
    - 😃: HW indipendente dal SW che gira $\rightarrow$ ambiente **OPEN** (no HW o SW proprietario)

NETWORD SLICING

La virtualizzazione ha introdotto il paradigma $network\ slicing$

- Suddividere la rete in più porzioni (virtuali) a seconda del servizio che si vuole offrire
  - In figura tutti i dispositivi risiedono nella parte fisica (in nero), ma logicamente sono suddivisi in livelli a seconda del servizio
  - Buona suddivisione anche per la privacy
  - Suddivisione più economica perché è logica non è fisica



- Tutto si adatta bene anche a modifiche/richieste particolari
  - Se vogliamo per un periodo dare più spazio a uno specifico ambiente (ad esempio $healthcare$), è sufficiente *riprogrammare* (quindi lato SW!) il livello relativo
    - Gli algoritmi AI riescono ad analizzare/predire eventuali cambiamenti
- ORCHESTRAZIONE: Saper gestire ad esempio un alto numero di reti di varia natura in modo orinato e coerente
  - Nello $slicing\ network$ è il mezzo che permette quanto già detto, ovvero riuscie a tenere in ordine e coordinati i vari livelli logici in cui la rete (fisica) è divisa