ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Scuola di Scienze
Corso di Laurea Triennale in Informatica

RASSEGNA SU SOFTWARE-DEFINED NETWORKING E OPENFLOW

Relatore

prof. Fabio Panzieri

Candidato

Enrico Marchelletta

II sessione
Anno Accademico 2014/2015

INDICE

INTRODUZIONE

La “rete delle reti”, oggi conosciuta come Internet, è il frutto dello sviluppo tecnologico e dei sistemi di telecomunicazione che nell'ultimo mezzo secolo hanno avuto un'impressionante evoluzione.

Agli inizi degli anni settanta venne sviluppata la prima rete di calcolatori Arpanet[1] con lo scopo di creare un nuovo sistema di comunicazione che favorisse lo scambio di informazioni a distanza. A partire da quel momento, questa struttura embrionale iniziò la sua inarrestabile crescita e in pochi decenni divenne quella che oggi è considerata la prima rete mondiale e il più grande mezzo di comunicazione di massa che la storia dell'uomo abbia mai conosciuto.

Questa rapida evoluzione ha portato con sé un'incessante richiesta da parte del mercato e degli utenti e nonostante i servizi offerti dalla rete internet siano cresciuti esponenzialmente, diventando sempre più numerosi e complessi, così come le tecnologie utilizzate dall'infrastruttura fisica, non è mai cambiato il paradigma su cui essa si basa, che è rimasto in sostanza invariato dalla sua nascita a oggi.

La capacità delle rete di rinnovarsi ha avuto negli ultimi anni un notevole rallentamento, un fenomeno che è stato definito come “ossificazione di internet”[2]. Oggi è diventato estremamente difficile introdurre una vera innovazione nel campo delle reti, sia per quanto riguarda la sua infrastruttura fisica sia per quanto riguarda i protocolli che utilizza.
L’esplosione delle applicazioni utilizzate in tempo reale ha portato i programmatori delle reti a dover riprogettare i protocolli tradizionali al fine di garantire adeguate prestazioni rispetto alle odierne esigenze.

Una soluzione a questi problemi potrebbe trovarsi in quello che oggi è stato definito come “Software-Defined Networking” (SDN) : un paradigma emergente nel campo

delle reti di calcolatori che consente di controllare, tramite un software centralizzato a livello logico, il comportamento dell'intera rete. La promessa di questa nuova tecnologia è quella di semplificare notevolmente la gestione della rete e dare uno slancio all'innovazione e all'evoluzione in questo campo.

Questa tesi è una rassegna sul tema del Software-Defined Networking e sul protocollo OpenFlow ovvero il fattore che ne abiliterebbe l'adozione nel prossimo futuro.
La struttura di questo scritto è composta da un primo capitolo in cui vengono esposti i limiti della rete attuale, viene definito il termine SDN e descritta l'architettura di base. Nel secondo, viene esaminato il protocollo OpenFlow attraverso una sintesi delle specifiche rilasciate dalla Open Network Foundation[3] con una descrizione generale del suo funzionamento. Nell'ultimo capitolo vengono elencate una serie di applicazioni sviluppate in ambito accademico e infine vengono esposte alcune problematiche riscontrate

Capitolo 1

Software-Defined Networking

Negli ultimi anni il settore delle telecomunicazioni e dell' IT (Information Technology) si è modellato sulla base del progresso tecnologico: la crescita esponenziale dei dispositivi mobili, dei servizi cloud e della virtualizzazione server, hanno cambiato il modo con cui gli host comunicano tra loro[4]. La rete attuale, formata da una serie di switch organizzati in una struttura gerarchica, riflette il tradizionale modello client-server non più adatto al dinamismo odierno ed alla flessibilità richiesta dai grandi data center, dalle università e dagli operatori. Con l'avvento dei data center, in un contesto in cui gli utenti finali sono al contempo fruitori e produttori di contenuti, c'è stato un cambiamento della direzione del traffico, per questo motivo i ricercatori stanno studiando soluzioni che permettano di transitare da un modello di tipo nord-sud ad uno di tipo est-ovest.
Con l'avvento dei servizi cloud e dell'uso della virtualizzazione nei sistemi di calcolo, la gestione delle infrastrutture di rete diventa sempre più complessa, richiedendo maggiore dinamicità e automazione.

Queste ed altre considerazioni, che verranno trattate fra poco, hanno portato al concepimento di un nuovo concetto di rete, quello di “rete programmabile”. L'idea di migrare verso reti definite via software viene incontro alle odierne esigenze e promette di portare grandi innovazioni in campo.

In questo capitolo partiremo dall'analisi dei limiti della architettura odierna, per poi passare all'esposizione di questo nuovo approccio architetturale, definendo il Software-Defined Networking, risalendo alle sue origini per poi arrivare a descriverne l' architettura ed elencarne i vantaggi.

1.1 Limiti della architettura odierna

Quando fu progettata l'infrastruttura di rete non era possibile prevedere con precisione il modo in cui si sarebbe evoluta la società e con essa le tecnologie attualmente disponibili. Lo sviluppo ottenuto fino ad oggi in questo campo ha raggiunto un limite difficile da oltrepassare rendendo complicato il lavoro di ristrutturazione dell' infrastruttura fisica, dei protocolli e delle relative performance.

Di seguito sono riassunte alcune delle limitazioni principali dell'odierna architettura di rete[4]:

Complessità. Il funzionamento delle reti di calcolatori è regolato da un insieme di protocolli progettati per collegare in modo affidabile host su differenti distanze, collegamenti fisici e topologie. Nel corso degli ultimi decenni i produttori degli apparati di networking hanno sviluppato protocolli per sfruttare al meglio le capacità della rete. Tali protocolli tendono ad essere sviluppati separatamente, ognuno assolve il proprio compito risolvendo uno specifico problema talvolta senza nessuna fondamentale astrazione comune. Questa moltitudine di protocolli differenti ha portato al limite principale delle reti odierne: la complessità.
A causa di questa complessità, le reti di oggi sono relativamente statiche e non sono in grado di adattarsi dinamicamente alle mutevoli esigenze delle richieste del traffico, delle applicazioni e degli utenti.

Politiche incoerenti. Per attuare una politica estesa a tutta la rete, bisogna configurare migliaia di dispositivi e meccanismi. Per esempio, ogni volta che una nuova macchina virtuale viene attivata, può richiedere ore, in alcuni casi giorni, per configurare le ACLs (Access Control List) su tutta la rete. La complessità delle reti di oggi rende molto difficile applicare un insieme coerente di accesso, sicurezza, QoS (Quality of Service) e altre politiche ad un numero sempre maggiore di utenti mobili, e lascia le aziende in uno stato di vulnerabilità rispetto a violazioni di sicurezza con ovvie conseguenze negative .

Scalabilità. Le esigenze dei Data Centers e dei Mega-operatori della rete, come Google, Yahoo!, Facebook, crescono molto rapidamente, ma le attuali tecnologie di rete faticano a rispondere tempestivamente ed efficientemente ad esse. Del resto, la rete diventa enormemente più complessa con l'aggiunta di centinaia o migliaia di dispositivi di rete e, considerando il dinamismo e l'imprevedibilità degli schemi di traffico, un buon grado di scalabilità non può essere raggiunto con una configurazione manuale.

Dipendenza dai fornitori. Gli operatori e le imprese cercano di implementare nuove funzionalità e servizi per rispondere rapidamente alle mutevoli esigenze aziendali o alle richieste degli utenti. Tuttavia, la loro capacità di rispondere a queste esigenze è ostacolata dai cicli di prodotti, che possono durare tre anni o più. La mancanza di interfacce aperte e standard, limita la capacità degli operatori di adattare la rete ai loro ambienti di lavoro. Questa mancata corrispondenza tra esigenze di mercato e capacità della rete ad innovarsi, ha portato questo settore ad un punto di svolta.

1.2 Definizione

Nel paragrafo precedente abbiamo analizzato i limiti delle architetture odierne, ora cerchiamo di definire questo nuovo approccio architetturale. Il termine SDN è divenuto un parola chiave nell'ambito delle reti e racchiude in sé molti aspetti. Non è facile dare una definizione esatta del termine poiché esso racchiude una serie di concetti e pratiche in continua evoluzione.

Il Software-Defined Networking è una proposta per la ristrutturazione della tradizionale architettura di rete. Un aspetto fondamentale per la realizzazione di questo nuovo paradigma è la separazione del piano di controllo (control plane) dal piano dei dati (data plane), solitamente legati ad unico dispositivo di rete, in cui il controllo è affidato ad un software, il controller appunto, attraverso il quale si cerca di realizzare il concetto di "rete programmabile".
Il piano di controllo si occupa principalmente della gestione del traffico dei pacchetti, attraverso l'aggiornamento continuo delle informazioni presenti nella tabella di instradamento (routing table). Il gestore dell'instradamento scambia informazioni circa la topologia della rete con altri router e definisce una routing table basandosi su protocolli come, RIP (Routing Information Protocol), OSPF (Open Shortest Path Forwarding), o BGP (Border Gateway Protocol). Il piano dei dati esegue le operazioni di inoltro in una modalità più rapida rispetto al piano di controllo. È responsabile del parsing degli header dei pacchetti e gestisce funzionalità come QoS, filtraggio, incapsulamento e accodamento dei pacchetti con delle predefinite politiche da applicare.

Un secondo aspetto essenziale risiede nello sviluppo di un protocollo di comunicazione standardizzato che permetta la comunicazione tra il piano di controllo e il piano di inoltro. In questa tesi è stato studiato il protocollo OpenFlow ovvero l' interfaccia che viene utilizzata dal piano di controllo per "programmare" il piano dei dati. Nel prossimo capitolo verrà approfondito e descritto il funzionamento di tale

elemento.

1.3 Architettura

L'architettura di rete SDN prevede tre livelli, aventi distinte funzionalità, in mezzo ai quali sono utilizzate le interfacce che permettono a ogni livello di comunicare con un altro.

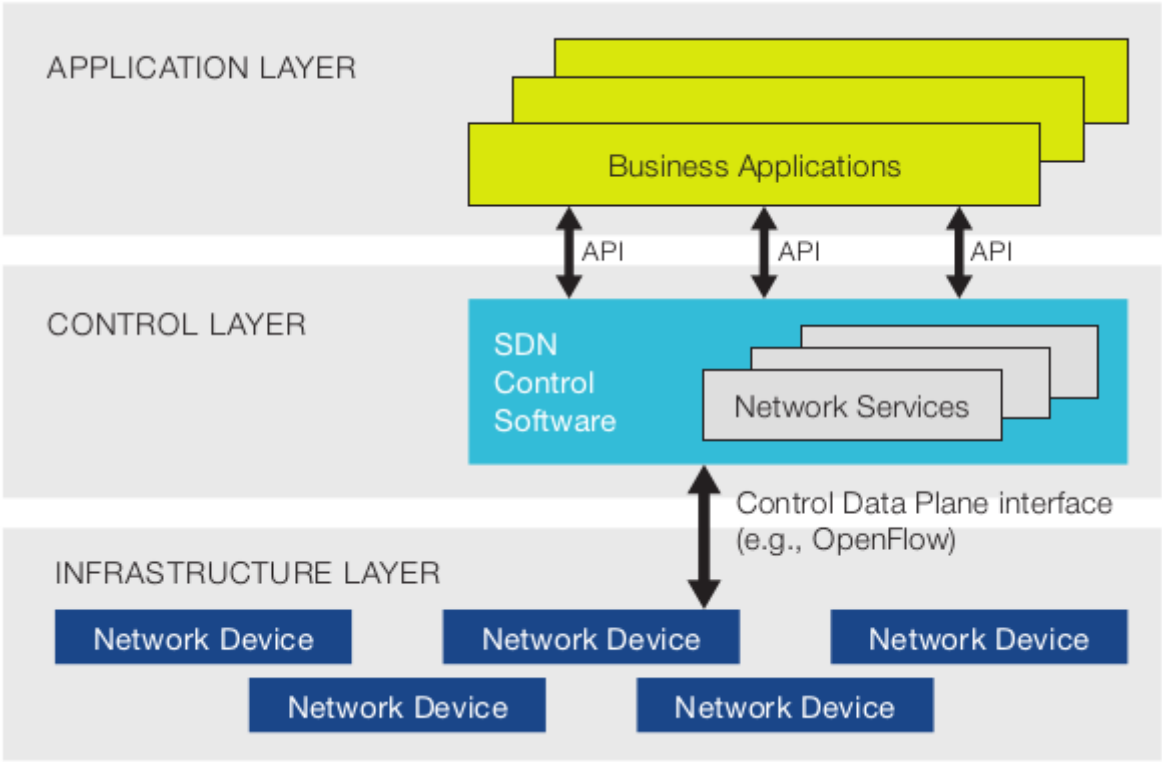

Fig. 1: Architettura SDN
Fonte : ONF

Infrastructure Layer. È il livello più basso, rappresenta l'infrastruttura di rete che comprende i dispositivi fisici che si occupano dell'inoltro dei dati attraverso la rete. A differenza della rete tradizionale questi apparati sono semplici elementi di inoltro dati dove il controllo logico e le decisioni di instradamento sono spostate in un controller

separato, il “cervello della rete”, situato sul piano di controllo.

Southbound Interface. Permette la comunicazione tra elementi di controllo e quelli di inoltro, rappresenta infatti uno strumento essenziale per ottenere la separazione tra le due differenti funzionalità.
Tale interfaccia può essere paragonata all'insieme di istruzioni di un processore, cioè la successione di quelle variabili primitive che possono essere visibili agli sviluppatori del software. Essa viene implementata sia sul lato del controller che sul lato dei dispositivi e instaura così un canale di comunicazione end-to-end sicuro grazie ai convenzionali protocolli crittografici come SSL (Secure Sockets Layer), o TLS (Transport Layer Security).
Dunque questa interfaccia rappresenta un punto critico dell'architettura SDN, in quanto non solo permette al controller di gestire in maniera dinamica e semplificata il traffico di una rete, ma ne incrementa l'efficienza in termini di traffico.
Nel prossimo capitolo viene approfondito il protocollo OpenFlow poiché è il più diffuso e maggiormente sviluppato, ma non l'unico disponibile. Ci sono infatti altre proposte come ForCES, POF, OpenState, OpFlex e altre ancora.

Control Layer. In questo livello troviamo il controller ovvero il fulcro dell’intera architettura SDN; centralizza gran parte dell’intelligenza della rete e svolge le funzioni per ottenere interoperabilità tra i dispositivi (fisici o virtuali) dell'infrastructure layer, dei quali nasconde le specificità, e l’application layer; inoltre mantiene una visione globale della rete. In particolare esso è fisicamente disaccoppiato dal layer di trasporto, al contrario dell’architettura attuale, dove il controllo è confinato nei dispositivi di rete, limitando l’accesso a software esterni e quindi ostacolando sviluppi specifici a supporto degli operatori e degli utilizzatori di rete.
Il controller è direttamente programmabile infatti risulta aperto agli operatori e agli utilizzatori della rete, consentendo di definire la rete tramite software, da cui appunto

deriva il nome Software-Defined Networking.

Northbound Interface. Rappresenta l'interfaccia di programmazione indispensabile per la comunicazione tra controller SDN e i software applicativi di controllo. È necessaria un'interazione multilaterale tra le due parti, in quanto determinate funzioni o servizi potrebbero dover acquisire informazioni riguardo i livelli inferiori della struttura di rete oltre quelle riguardanti la normale gestione delle risorse e del traffico. Mentre per l'implementazione standard delle interfacce inferiori, situate tra controller e switch, OpenFlow risulta il protocollo prevalente, per quanto riguarda le NBI attualmente non esiste un modello di API (Application Program Interface) dominante. Questa mancanza rappresenta un punto debole dell'attuale sviluppo dell'architettura SDN, infatti, pensando a un controller come un "sistema operativo di rete", rimane necessaria la presenza di un'interfaccia che permetta l'astrazione logica dell'hardware sottostante così da non rendere essenziale, per chi sviluppa applicazioni software, la conoscenza dettagliata dell'implementazione di un controller.
Open Networking Foundation ha assegnato ad un team di lavoro il compito della standardizzazione di un'interfaccia che ha l'obiettivo di definire l'interfaccia esposta dallo strato di controllo verso le applicazioni.

Application Layer. A questo livello operano tutte quelle applicazioni di rete che sfruttano il livello di controllo per implementare funzionalità che saranno tradotte in comandi da eseguire sul piano dati, che poi detteranno il comportamento dei dispositivi di inoltro.
Le applicazioni per reti software-defined possono essere distribuite su qualsiasi ambiente tradizionale di rete, dalla rete casalinga a quella aziendale fino ad arrivare ai grossi ambienti come i data center.
Tali applicazioni possono svolgere le classiche funzioni di routing, bilanciamento del carico, tolleranza agli errori e applicazione di politiche di sicurezza, ma anche esplorare nuovi approcci, come ad esempio riduzione del consumo di energia. Altri

esempi includono funzionalità per la gestione della mobilità in reti wireless o per sfruttare al meglio le reti ottiche.
La varietà di queste applicazioni dovrebbe essere una delle forze principali per la promozione di SDN.

Nel ultimo capitolo sono descritte alcune di queste applicazioni facendo riferimento costante ad articoli scientifici dai quali sono state tratte.

1.4 Le origini

L'architettura internet è sempre stata un argomento di ricerca ed esplorazione fin dalla sua nascita. Ci sono stati diversi programmi governativi e progetti di ricerca per rendere migliore la rete e, una volta garantita la connettività tra gli elaboratori arbitrariamente distribuiti su tutta la rete, l'attenzione si è concentrata sulla sua programmabilità. Possiamo dividere la storia della rete programmabile in tre fasi[5]: la prima riguarda le *Active Networks.* Durante la metà degli anni novanta la Defense Advanced Research Projects Agency (DARPA) negli Stati Uniti lanciò un programma su active networking che permettesse all'utente e agli operatori di rete di "programmare" gli elementi di rete a livello del data plane. Successivamente, anche in Europa venne lanciato il Future Active IP networks (FAIN) con scopi molto simili.
La seconda fase ebbe inizio dieci anni più tardi, quando i ricercatori proposero *Forwarding and Control Element Separation (ForCES)* un protocollo che realizzasse la programmabilità della rete a livello di flusso. Purtroppo nonostante la proposta fosse molto allettante, essa non trovò un riscontro sul piano reale.
In quegli stessi anni, siamo intorno al 2005, la U.S. National Science Foundation lanciò un programma chiamato Global Enviroment For Networks Innovations (GENI) [6] per la creazione di un'infrastruttura su la larga scala che aveva come obiettivo la sperimentazione su reti di calcolatori e sistemi distribuiti. Analogamente in Europa nasceva Future Internet Research and Experimentation (FIRE).
La terza fase ebbe inizio nel 2007 con lo sviluppo del protocollo OpenFlow che ereditò il lavoro fatto con ForCES, SANE, Ethane e 4D, ed altri progetti. In particolare modo, Ethane dimostrò nel 2006 come l'approccio ad una logica centralizzata ottenuta attraverso la separazione del piano di controllo da quello d'inoltro poteva essere applicata con successo per la rete di un campus universitario[6]. Il piano di controllo veniva realizzato tramite un network OS, il quale costruiva e presentava una mappa logica di tutta la rete alle applicazioni sovrastanti.

Di seguito viene riproposto uno schema cronologico dell'evoluzione della rete

programmabile:

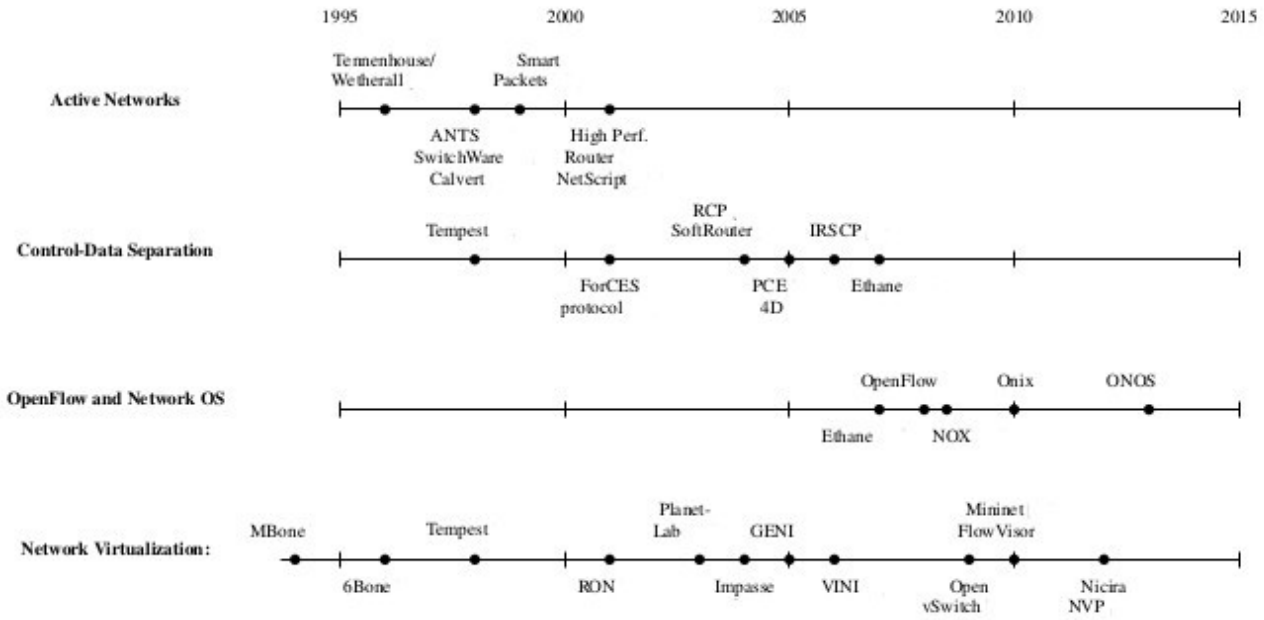

Fig. 2: Cronologia dell'evoluzione della rete programmabile

1.5 Vantaggi di una rete SDN

Il Software-Defined Networking centralizzerà e semplificherà le gestione dei nodi della rete. Tutto questo si tradurrà in una maggiore programmabilità e flessibilità e darà la possibilità di offrire una molteplicità di servizi di rete.

I principali vantaggi sono :

Maggiore controllo di rete. Il controllo centralizzato e disaccoppiato dai singoli dispositivi fornisce all'operatore una manovra precisa e granulare del traffico, come ad esempio la possibilità di applicare determinate regole per elementi specifici connessi alla rete. Ciò permette l'utilizzo di applicazioni "multi-tenant", cioè in grado di creare partizioni di istanze virtuali per ogni utente, pur mantenendo allo stesso tempo livelli elevati di sicurezza, controllo della congestione e gestione dinamica delle risorse

Sicurezza. La virtualizzazione ha reso la gestione della rete più impegnativa, infatti con le macchine virtuali è diventato più difficile applicare in modo coerente firewall e una politica di filtraggio dei contenuti.
Con il controller SDN si fornisce un punto di controllo centrale per la gestione della sicurezza, questo ha lo svantaggio di creare un unico punto di attacco, ma se viene implementato in modo sicuro e corretto è un ottimo modo per gestire la sicurezza. Inoltre grazie a SDN si rende possibile bloccare selettivamente il traffico malevolo lasciando inalterato il normale flusso.

Abbassamento dei Costi Operazionali. Un'amministrazione più efficiente, un miglioramento nell'utilizzo dei server, e una migliore virtualizzazione server porteranno alla riduzione dei costi operazionali. Anche se è ancora presto per dare una prova effettiva del risparmio che ci sarà, SDN attraverso un'amministrazione centralizzata e automatizzata abbatterà i costi richiesti dalle utilizzo di risorse umane.

Minore spesa per Hardware. La semplificazione dell'hardware di rete causerà una maggiore concorrenza nel mercato che porterà ad una riduzione dei costi dei dispositivi.
L'hardware esistente potrà essere riutilizzato e nuovi dispositivi potranno essere sviluppati con minori costi dal momento che limiteranno le loro funzioni all'inoltro dei pacchetti dal momento che le decisioni di instradamento saranno prese dal controller.

Consegna garantita dei contenuti l'architettura SDN fornisce un set di API che consentono di implementare servizi di rete comuni, come routing, multicast, security, access control, bandwidth management, traffic engineering, quality of
service, processor e storage optimization, energy usage, policy management.

Capitolo 2

Il protocollo OpenFlow

OpenFlow è la prima interfaccia aperta[7] per la comunicazione tra piano di controllo e piano di inoltro, ed è divenuto uno standard *de facto* nell'ambito del Software-Defined Networking. Il controller attraverso questo protocollo può accedere direttamente e manipolare il piano di inoltro dei dispositivi di rete, come switches e routers, siano essi fisici o virtuali. Il protocollo OpenFlow è considerato il fattore abilitante per realizzare il Software-Defined Networking e più in generale verso i concetti di rete flessibile e programmabile.

Nasce dall'esigenza di spostare il controllo della rete dagli apparati di networking ad un software logicamente centralizzato al fine di rendere possibile ai ricercatori di testare idee innovative, come nuovi protocolli di routing, modelli di sicurezza, schemi di indirizzamento ed eventuali alternative a IP.

Il suo sviluppo ebbe inizio nel 2007 da una collaborazione tra la Stanford University e la University of Californa a Berkley, nel 2011 il progetto approdò alla ONF (Open Network Foundation), un'organizzazione fondata da Deutsche Telekom, Facebook, Google, Microsoft, Verizon, Yahoo!, attuale responsabile del processo di standardizzazione.

Le funzionalità fornite da OF includono analisi del traffico, controllo centralizzato della rete, aggiornamento dinamico delle regole di inoltro e astrazione del flusso. Alcune delle applicazioni basate su tale protocollo sono state proposte per facilitare la configurazione di rete, per semplificarne la gestione, per aggiungere nuove feature di sicurezza e per la virtualizzazione di rete nei data centers.

OF offre grandi opportunità all'innovazione nel campo del networking ma al contempo deve far fronte a delle problematiche relative a compatibilità, scalabilità, stabilità e sicurezza, ed è proprio su questi punti che si concentrano le ricerche che guideranno lo sviluppo di questa tecnologia.

In questo capitolo verrà esposta una sintesi delle specifiche del protocollo, si descriverà il suo funzionamento e verranno portati esempi al fine di rendere più facile la comprensione del suo utilizzo.

2.1 Specifiche di OpenFlow

Le specifiche considerate in questa tesi sono le ultime rilasciate dall'ONF: OpenFlowSwitch Specification, versione 1.5.0, dicembre 2014 [8]

L'interfaccia realizzata da OpenFlow si colloca al livello più basso di astrazione previsto dall'architettura SDN (vedi fig.1). Uno degli aspetti fondamentali, che sta alla base dell'attività di specifica avviata da OpenFlow, consiste nella definizione di un modello standard dell'hardware di forwarding dei pacchetti che costituisce il nucleo dei diversi dispositivi di networking.

OpenFlow fornisce un protocollo basato sul concetto di flusso (flow) ovvero una sequenza unidirezionale di pacchetti che attraversano un nodo entro un determinato intervallo temporale e aventi caratteristiche comuni con sorgente e destinazione fissate. Per esempio, un flusso può essere una connessione TCP (Transmission Control Protocol), o tutti i pacchetti che corrispondono a un determinato indirizzo MAC (Media Access Controll addres) o IP (Internet Protocol) di origine, o tutti i pacchetti con lo stesso ID WLAN (Wireless Local Area Network), o tutti i pacchetti della stessa porta fisica di uno switch.
Per identificare il traffico di rete utilizza delle predefinite regole di corrispondenza (match rules) che permettono di programmare la tabella di flusso (flow table**)** di svariati switch. In questo modo, un ricercatore può partizionare il traffico di rete in flussi di produzione e ricerca, controllando i flussi e scegliendo quali strade i pacchetti dovranno seguire e quale trattamento dovranno ricevere.
Normalmente l'implementazione delle tabelle di classificazione dei pacchetti, che gestiscono la definizione dei flussi e le relative regole di inoltro è una caratteristica proprietaria del particolare apparato di networking. Per superare questo modello, OpenFlow mira ad individuare, specificare e rendere accessibili attraverso il protocollo, un insieme di funzioni supportate dalla maggior parte dei router o switch commerciali.[9]

L’obiettivo principale consiste quindi nel definire un modello astratto dell’elemento che esegue il forwarding dei pacchetti, rendendolo programmabile attraverso un’interfaccia aperta e standard.

Attualmente, gli switch conformi ad OpenFlow (OpenFlow-compliant) si dividono in due tipi: *OpenFlow-only* e *OpenFlow-hybrid*.

OpenFlow-only switch supporta solamente operazioni su OpenFlow, in questi switch tutti i pacchetti seguono una OpenFlow pipeline e non possono essere elaborati diversamente.

OpenFlow-hybrid switch supporta sia operazioni OpenFlow che normali operazioni di commutazione ethernet poiché dotati di un meccanismo di classificazione per la gestione di entrambe le pipeline.

Un OpenFlow Switch è composto principalmente da un insieme di porte di ingresso, attraverso le quali i pacchetti vengono ricevuti per essere processati attraverso la pipeline costituita a sua volta da un insieme di flow table connesse tra loro, ed inoltrati tramite una porta di uscita. La group table è un gruppo di voci che consente la rappresentazione di metodi di invio alternativi. La comunicazione tra swtich e il processo di controller remoto, il cosidetto controller, avviene su di un canale di controllo (Control Channel).

In basso, una rappresentazione delle componenti appena elencate seguita da una descrizione dei dettagli di funzionamento.

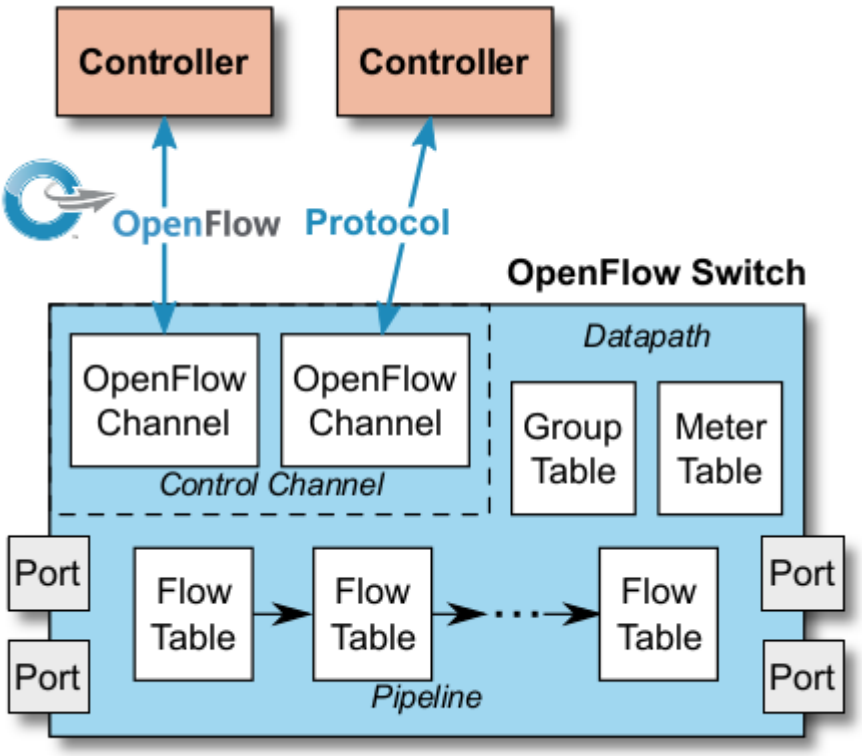

Fig. 3: Componenti si uno switch OpenFlow

2.1.1 Porte di OpenFlow

Un switch OpenFlow-compliant deve supportare tre tipi di porte: porte fisiche (physical ports), porte logiche (logical ports) e porte riservate (reserved ports).

Le porte fisiche corrispondono alle interfacce fisiche dello switch. In alcuni casi possono essere virtualizzate attraverso l'hardware di commutazione, rappresentando di conseguenza solo una parte virtuale dell'interfaccia di quest'ultimo.
Le porte logiche sono porte definite dallo switch che non corrispondono direttamente a un'interfaccia hardware di esso. Le porte logiche sono astrazioni a livello superiore che possono essere definite mediante metodi diversi da OF, come le interfacce di loopback, link aggregation o tunneling, e possono comprendere l'incapsulamento di pacchetti e il port mapping su una o più porte fisiche.

L’elaborazione eseguita da tali porte deve essere dipendente all’elaborazione OF a livello di implementazione, in modo che le porte logiche possano essere trattate come porte fisiche. L’unica differenza tra porte fisiche e logiche sta in una voce aggiuntiva chiamata “Tunnel-ID”, presente su un pacchetto processato a livello logico e contenente l’indirizzo sia della porta logica sia della porta fisica associata;

Le Porte Riservate definiscono azioni di inoltro generiche come l'inoltro di pacchetti verso un controller, il flooding, o l’inoltro tradizionale senza metodi OpenFlow come fosse uno switch tradizionale

- `ALL`: rappresenta tutte le porte che lo switch può usare per inoltrare un determinato pacchetto, può essere utilizzata solo come porta d’uscita. Una copia del pacchetto verrà inviata su tutte le porte standard OF, ad eccezione della porta d’ingresso di quest’ultimo e delle porte configurate ad hoc;
- `CONTROLLER`: rappresenta il canale di controllo con i controller OF e può essere usata sia come porta d’ingresso che d’uscita. Nel primo caso il pacchetto viene identificato come “ricevuto dal controller”, mentre nel secondo caso viene incapsulato e inviato mediante il protocollo OpenFlow;
- `TABLE`: utilizzabile solo come porta in uscita, conduce il pacchetto verso l’inizio della fase di ricerca nelle tabelle di flusso;
- `IN_PORT`: invia il pacchetto attraverso la sua porta di ingresso;
- `ANY`: viene utilizzata nel caso particolare in cui un’operazione di OpenFlow può essere eseguita su una porta qualsiasi;
- `LOCAL`: permette ad elementi remoti l’interazione con lo switch median
- `FLOOD` : rappresenta il flooding dei pacchetti effettuato utilizzando la tradizionale pipeline dello switch non basata su OpenFlow

2.1.2 Elaborzione attraverso pipeline

Uno switch compatibile con OpenFlow contiene una o più flow tables, ognuna delle quali contiene una lista di flow entries. La pipeline di elaborazione di OpenFlow definisce il modo in cui i pacchetti interagiscono con queste flow tables.

Le flow tables di uno switch-OF sono sequenzialmente numerate partendo da zero.
Il processo di confronto tra un dato pacchetto e l'insieme delle flow entries delle tabelle di flusso avverrà esclusivamente in ordine crescente, pertanto inizierà sempre dalle voci presenti nella tabella numero 0.
L'esecuzione della pipeline avviene in due fasi: elaborazione in ingresso (ingress processing) ed elaborazione in uscita (egress processing). La separazione delle due fasi è stabilita dalla prima egress table, tutte le tabelle con un numero più basso della prima egress table devono essere utilizzate come ingress table e nessuna tabella con un numero più alto o uguale alla prima egress table può essere usata come una ingress table. (vedi Fig. 4)

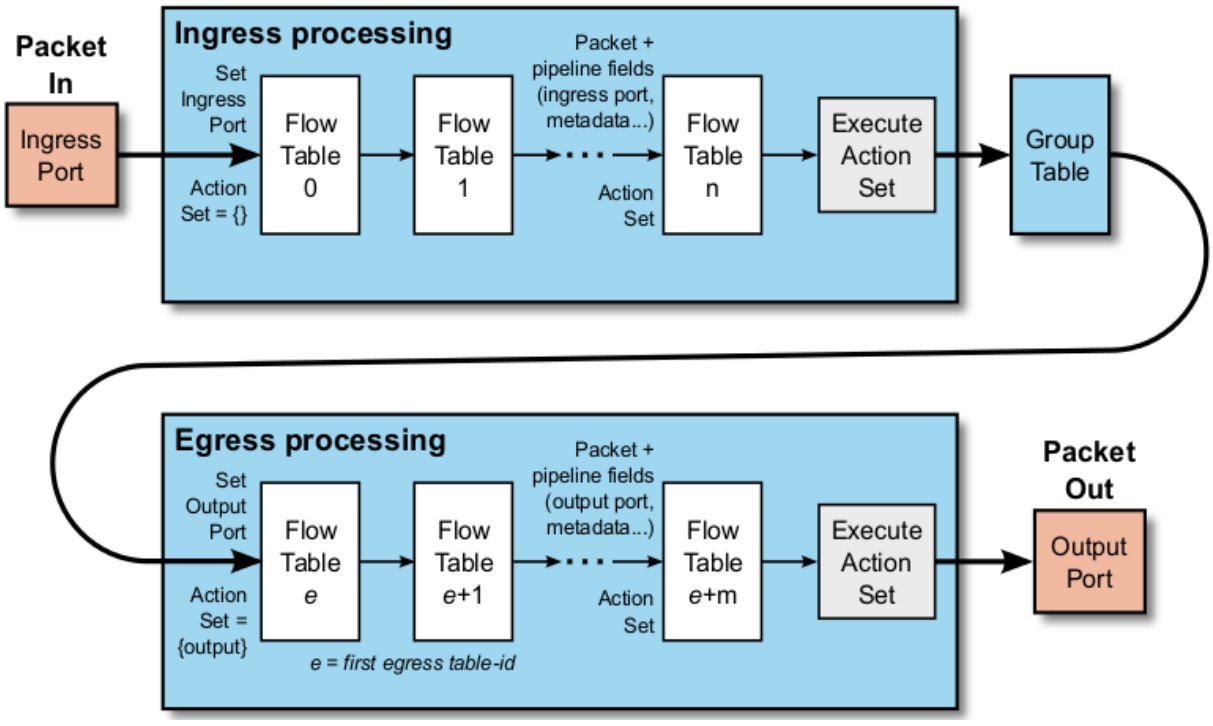

Fig. 4: OpenFlow pipeline

L'elaborazione attraverso pipeline comincia sempre dalla prima flow table: il pacchetto viene confrontato con le flow entries della flow table 0. Altre ingress tables potrebbero essere usate in base alle informazioni ottenute in uscita dalla prima tabella.
Se nessuna tabella di uscita valida è configurata, il pacchetto verrà spedito verso la porta in uscita, e nella maggior parte dei casi il pacchetto è inoltrato al di fuori dello swtich. Se una tabella d'uscita valida è configurata, il pacchetto viene confrontato con le flow entries della flow table e altre egress table possono essere usate in base all'uscita in quella flow table.

Se un pacchetto non trova nessuna corrispondenza con una flow entry della flow table verrà associato alla table-miss. Il comportamento in caso di mancanza di corrispondenza dipende dalla configurazione della table-miss. Le istruzioni incluse nella flow entry della table-miss possono specificare in modo flessibile come elaborare i pacchetti che non hanno trovato corrispondenze, possono essere scartati, o inoltrati ad altre tabelle o inviati ai controllers.
Se la flow entry nella table-miss non è presente, il pacchetto viene automaticamente scartato.

Ogni flow table nello switch contiene un insieme di flow entries, ognuna di queste è composta da campi di corrispondeza (match fields), priorità (priority), contatori (counters), un insieme di instruzioni (set of instructions), scandenze (timeouts), cookie , flags, da applicare ai pacchetti corrispondenti (matching packets). (Fig.)

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Fig. 5: Struttura di una Flow Entry in una Flow Table

- `Match Fields`: permettono di trovare la corrispondenza. Includono : porta d'ingresso, intestazione (header) del pacchetto, eventuali campi aggiunti nell'elaborazione di pipeline come metadati specificati da flow tables precedenti

- `Priority:` stabilisce la priorità di una flow entry
- `Counters:` vengono aggiornati quando viene trovata una corrispondenza
- `Instructions:` istruzioni da seguire
- `Timeouts:` stabilisce le scadenze di un flusso
- `Cookie:` potrebbero essere utilizzati dal controller per estrarre informazioni relative a statistiche, modifiche o richieste di cancellazioni del flusso
- `Flags:` alterano il modo in cui vengono gestiti i flussi

Le Flow entries associano i pacchetti in ordine di priorità infatti viene utilizzata la prima entry corrispondente per ogni flow table. Se esite una entry corrispondente, l'istruzione associata viene eseguita. Se non corrisponde a nessuna entry nella flow table, l'uscita dipende dalla configurazione della flow entry nella table-miss: per esempio, il pacchetto può essere inoltrato ai controller sul canale openflow, può essere scartato o potrebbe continuare sulla flow table successiva.

Una group table è formata da un gruppo di voci. Quando una flow entry punta ad un gruppo rende possibile ulteriori metodi di inoltro.

Qui viene rappresentata una voce di gruppo e per ogni campo viene spiegata la sua funzione

Group Identifier	Group Type	Counters	Action Buckets

Fig. 6: Struttura di una voce di gruppo

- `Group Identifier :` identifica univocamente il gruppo
- `Group Type:` stabilisce la semantica da applicare
- `Counters:` conta il numero di elaborazioni effettuate dal gruppo

- `Action Buckets`: lista ordinata di buckets di azioni.

OpenFlow Channel e Control Channel

Attraverso il canale OpenFlow il controller configura e gestisce lo switch, riceve eventi da esso e spedisce i pacchetti al di fuori dello switch. Il canale di controllo supporta un solo canale per un singolo controller ma può anche supportare più canali nel caso di ci fossero più controller.

Le comunicazioni possono avvenire attraverso tre tipi di messaggi : controller-to-switch, asincorono, e simmetrico, ognuno dei quali supporta più sottotipi.

Controller-to-Switch è inizializzato dal controller e potrebbe non richiedere una risposta dallo switch. Viene utilizzato per ottenere informazioni circa l'identità e le funzionalità di base di uno switch, per settare delle configurazioni, per modificarne lo stato, aggiungendo o elimando le voci nelle flow tables, per leggerne lo stato e per l'inoltro dei pacchetti

I messaggi asincroni vengono spediti da uno switch per notificare l'arrivo di un pacchetto o un cambiamento di stato. Sono inviati nei casi in cui i pacchetti non hanno trovato nessuna corrispondenza con le flow-entries e quindi vengono inoltrati al controller, oppure vengono utilizzati per la notifica di particolari eventi, come la rimozione di una flow entry da una flow table, il cambiamento di una porta, a seguito della perdita di un collegamento, per monitorare lo stato delle flow tables.

I messaggi sincroni vengono spediti da ambedue le direzioni senza alcuna sollecitazione. Sono scambiati per inizializzare la connessione, per verificare che la connessione sia attiva o per misura latenza e larghezza di banda, per notificare errori come il fallimento di una richiesta inizializzata dal controller, per fornire un modo per sperimentare e offrire funzionalità aggiuntive.

2.2 Funzionalità

Le architetture basate su OpenFlow permettono un controllo centralizzato del network, analisi del traffico, aggiornamento dinamico delle regole di inoltro e astrazione del flusso. In questa sezione descriviamo queste funzionalità[7] e mostriamo alcuni esempi che illustrano come possono essere sfruttate.

Controllo centralizzato del network.

Una funzionalità importante di una rete OpenFlow deriva dal fatto che il controllore ha una conoscenza astratta di tutta la rete sottostante. Diversi switch possono essere collegati ad un solo controllore il quale permette di prendere decisioni in maniera centralizzata. Invece di avere diversi dispositivi di rete con una conoscenza limitata del network, un controllore ha una vasta conoscenza della rete questo rappresenta un grande vantaggio nella gestione della rete.

Gli switch diventano semplici macchine che inoltrano e scartano pacchetti secondo le regole definite dal controllore.

Un altro esempio di questa funzionalità è dato dal recupero dei collegamenti falliti. In un network tradizionale, quando un collegamento fallisce le strade vengono corrette su ogni switch fino a quando vengono trovate nuove strade. In un network OpenFlow un controllore centralizzato può trovare nuovi percorsi in modo più veloce e semplice.

Analisi del traffico basate sul software (Software-based traffic analysis)

Le analisi del traffico basate su software offrono una buona possibilità d'innovazione in quanto è possibile migliorare le funzionalità di uno switch usando qualsiasi tecnica basata su software.

Ad esempio, proprio sull'analisi del traffico sono basati diversi metodi di rilevamento di attacchi di Denial of Service (Ddos).

Tale metodo si basa sul recupero dei dati di traffico ad intervalli periodici e

sull'utilizzo di mappe auto-organizzate per classificare il traffico come normale o maligno. Dato che le analisi di traffico sono fatte dal software, ci sono molte più possibilità di utilizzare configurazioni avanzate per svolgere tali analisi.
Un'altra applicazione di questa funzionalità è la convalida dell'indirizzo sorgente di ogni nuovo flow. Quando uno switch inoltra un pacchetto al controllore perché non corrisponde con nessuna voce nella flow table, il controllore può oppure no validare se la sorgente dell'indirizzo corrisponde ad un flow valido.

L'aggiornamento automatico delle regole di inoltro

Un'altra funzionalità è quella di permettere l'aggiornamento automatico delle regole di inoltro. Tutti i tipi di modifiche alla topologia di rete possono essere attuate in tempo reale, effettuate in base alle decisioni prese dal controllore. Nessun intervento manuale è richiesto. Ciò è possibile perché il controllore può modificare le voci della flow table in ogni istante.
In questo modo se al controllore viene notificato un errore di collegamento diventa possibile correggerlo senza richiedere alcuna azione all'amministratore di rete. Inoltre, per ottimizzare la gestione del traffico, il controllore può automaticamente assegnare larghezza di banda a seconda del carico di traffico.
Infatti, un'applicazione di questa funzionalità è il bilanciamento del carico. Il controllore può valutare il carico di rete di diversi severs e cambiare dinamicamente le regole di inoltro per assicurare che il carico sia bilanciato correttamente.

2.3 Esempi

Si cercherà ora di far comprendere alcune delle funzionalità appena introdotte attraverso alcuni esempi.

Esempio 1

Come semplice esempio[10] si consideri il caso in cui si voglia testare un nuovo protocollo di instradamento in una topologia di rete composta da end-host e da switch OpenFlow. L'esempio verrà eseguito all'interno di un PC desktop che funge da controller, in modo tale da evitare di interferire nella rete. Ogni volta che un nuovo flusso applicativo viene generato, il protocollo che si vuole testare sceglie un percorso attraverso una serie di switch OpenFlow e aggiunge una flow entry in ognuno lungo il percorso. Per il traffico in entrata nella rete OpenFlow si definisce un flusso in modo che tutto il traffico entrante nello switch passi attraverso la porta connessa al PC e si aggiunge una entry con l'azione incapsula e spedisci tutti i pacchetti al controller. Quando tutti i pacchetti arrivano al controller, il protocollo sceglie un percorso e aggiunge una nuova flow entry per ogni switch lungo tutta la traiettoria. Quando i pacchetti successivi giungono nello switch, vengono processati velocemente attraverso le flow table (essendo già presenti le regole per gestirli).
Ci sono domande legittime da porsi circa le prestazioni, l'affidabilità e la scalabilità di un controller che aggiunge e rimuove in modo dinamico flussi per osservare come un esperimento procede:

- Puo un controller centralizzato essere abbastanza veloce per elaborare
nuovi flussi e programmare gli switch?

- Cosa succede quando un controller si guasta?

Per rispondere a queste domande, in letteratura si è testata una semplice topologia che ha utilizzato dei semplici switch ed un controller centrale. I risultati preliminari

mostrano che un controller come questo è in grado di elaborare oltre 10.000 nuovi flussi al secondo, sufficienti per un grande campus universitario.
Naturalmente, il tasso con il quale i nuovi flussi possono essere elaborati dipende dalla complessità della computazione richiesta dagli esperimenti del ricercatore. Se ci spostassimo in una rete di più larga scala, nella quale il protocollo che si vuole testare viene testato in una rete utilizzata da molte altre persone, sarebbe desiderabile che la rete avesse due caratteristiche aggiuntive:

1. Pacchetti che appartengono a utenti dovrebbero essere instradati tramite un protocollo standard di instradamento eseguito nello switch o nel router;

2. Colui che gestisce la rete dovrebbe essere in grado di aggiungere voci di flusso per il suo traffico, in modo tale da consentirgli di controllare i flussi e la stabilità della rete.

La prima caratteristica si ottiene dall'abilitazione degli switch OpenFlow; è possibile installare al loro interno delle regole che permettano la gestione di determinati pacchetti appartenenti ad un singolo utente. La seconda invece dipende dal controller; esso dovrebbe essere visto come una piattaforma che permette ai ricercatori di implementare i loro esperimenti. Le restrizioni della seconda caratteristica possono essere ottenute con un appropriato uso dei permessi, limitando il potere degli altri utenti sul controllo delle entry delle flow table.

Esempio 2: VLAN

OpenFlow può facilmente fornire un aiuto agli utenti con la propria rete isolata, proprio come fanno le VLAN2 . L'approccio più semplice è dichiarare un insieme di flussi che specifica le porte accessibili dal traffico su un determinato ID VLAN. Il traffico identificato come proveniente da un singolo utente (ad esempio, provenienti da porte specifiche o indirizzi MAC) viene etichettato dagli switch (tramite un'azione)

con l'ID VLAN appropriato. Quindi quando l'utente genera del traffico, questo traffico ha l'ID della VLAN alla quale l'utente appartiene. Un approccio più dinamico potrebbe utilizzare un controller per gestire l'autenticazione degli utenti e utilizzare la conoscenza delle posizioni degli utenti per la codifica del traffico in fase di esecuzione.

Esempio 3: Mobile wireless VOIP

Per questo esempio si consideri un esperimento di un nuovo meccanismo di chiamata tramite smartphone attraverso la tecnologia Wi-Fi. I client stabiliscono una nuova connessione tramite una rete di switch OpenFlow. Un controller è implementato per tenere traccia della posizione dei clienti, delle connessioni e del re-instradamento, riprogrammando le tabelle di flusso, in base a come gli utenti si muovono attraverso la rete, consentendo il trasferimento da un punto di accesso all'altro. Se un client che sta eseguendo una chiamata cambia punto d'accesso, il controller se ne accorge e riconfigura le flow table in modo da consentire la continuazione della chiamata a fronte di questo cambio.

Capitolo 3

Applicazioni SDN/OpenFLow

3.1 Aree di utilizzo

Gestione di rete

Molte applicazioni basate su SDN e OpenFlow sono state sviluppate per garantire la disponibilità di rete attraverso metodi di bilanciamento del carico e di tolleranza agli errori.

Il bilanciamento del carico è una tecnica largamente usata per distribuire il carico di lavoro tra due o più nodi della rete. La tolleranza agli errori invece si riferisce alla proprietà di un sistema di non subire avarie (cioè interruzioni di servizio) e di continuare a funzionare anche in caso di guasti.

Bilanciamento del carico: per ottenere questo risultato è stato proposto Plug-n-Serve[11], un sistema di bilanciamento del carico per reti non strutturate con lo scopo di minimizzare i tempi di risposta.
Il Plug-n-Serve controller, implementando un algoritmo di ottimizzazione chiamato LOBUS (Load-Balancing Over Unstructured Networks), tiene conto della congestione della rete e analizzando il carico dei server in tempo reale decide dove sia opportuno dirigere il traffico al fine di bilanciare al meglio il traffico su tutti i nodi disponibili. Usando questa soluzione, è anche possibile aggiungere nuovi server al cluster. Il software infatti è in grado di rilevarli e aggiungerli nel sistema di bilanciamento del carico in maniera dinamica. Una versione migliorata di Plug-n-Serve è Aster*x[12] utilizzato anche all'interno GENI ma su una scala molto più grande di quella impiegata nell'utilizzo di Plug-n-Serve.

Tolleranza agli errori: un altro studio ha esplorato la tolleranza agli errori ed eventuali guasti. In questo articolo[13], gli autori spiegano come il controller sia in grado di cambiare dinamicamente le regole di routing quando viene rilevato un errore di collegamento.
Gli autori sostengono che le reti di grossa portata devono essere in grado di recuperare i collegamenti in meno di 50 ms. Gli esperimenti hanno avuto esiti positivi, ciò nonostante è emerso che la dipendenza dal controller centralizzato rende l'obiettivo di rimanere sotto la soglia dei 50 ms impegnativo da raggiungere.

Ricapitolando, il bilanciamento del carico viene eseguito grazie alla capacità del controller di modificare le regole di inoltro mentre negli studi sulla tolleranza agli errori, dopo che un errore si è verificato, si riescono a trovare nuove strade per mantenere i collegamenti attivi grazie alla capacità di ottenere una visione globale della topologia di rete. Tradizionalmente, questo tipo di recupero è realizzato dagli switch che non hanno questa funzionalità, per questo motivo risulta più difficile questa operazione.

Data Center

I data center si sono evoluti ad un ritmo impressionante negli ultimi anni, cercando sempre di soddisfare le richieste del mercato in rapida evoluzione. Un'attenta gestione del traffico e l'applicazione delle policy è fondamentale quando si opera su una cosi vasta scala, specialmente quando l'interruzione del servizio o ulteriori delay possono portare alla perdita della produttività con conseguenti perdite di profitto.

Un fattore molto importante da considerare in tale contesto è il consumo di energia, che ha un costo per niente irrisorio nei data-center di grandi dimensioni. In un articolo[13] vengono mostrare tecniche per il miglioramento dei server, in particolare sul sistema di raffreddamento, che costituisce il 70% del totale consumo energetico,

attraverso una migliore gestione dell'hardware e del software.

I ricercatori della Stanford University hanno proposto ElasticTree[14], un power manager che utilizza SDN per trovare il sottoinsieme di nodi di rete con i più bassi consumi in modo tale che soddisfino le condizioni di traffico in un determinato momento e spegnendo gli switch che non sono necessari. Come risultato, essi mostrano un risparmio energetico tra 25-62% in differenti condizioni di traffico. Si può immaginare che questi risparmi possano essere ulteriormente aumentati se utilizzati in parallelo con la gestione dei server e la virtualizzazione; una possibilità è Honeyguide [15] un approccio per l'ottimizzazione energetica che utilizza la migrazione di macchine virtuale per aumentare il numero di macchine e switch che possono essere arrestati.

Tuttavia, non tutte le soluzioni SDN possono essere appropriate alle reti ad alte performance. Mentre la gestione semplificata del traffico e la visibilità globale della rete sono utili, tutto questo influisce sulla scalabilità e riduzione delle prestazioni. Alcuni ricercatori[16] credono che OpenFlow tenda ad accoppiare eccessivamente la logica di controllo con una visibilità completa, quando in realtà solo i flussi "significativi" dovrebbero essere gestiti; questo approcci può portare a colli di bottiglia, durante la comunicazione tra piano di controllo e piano dati sono aggiunti delay non necessari e allo stesso tempo gli switch sono sovraccaricati di migliaia di voci nella tabella di flusso.
Il loro framework, DevoFlow, propone alcune semplici modifiche di progettazione per mantenere i flussi nel piano dati quanto più possibile, pur mantenendo una sufficiente visibilità della rete e una gestione dei flussi efficace. Questo si ottiene spostando la responsabilità della maggior parte dei flussi di nuovo sugli switch e aggiungendo più efficienti meccanismi di raccolta delle statistiche, attraverso i quali i flussi "significativi" (ad esempio quelli a lunga durata e con elevato throughput) sono identificati e gestiti dal controllore. In una simulazione di bilanciamento del carico, la loro soluzione aveva 10-53 volte meno voci nella tabella di flusso e 10-42 volte meno

messaggi di controllo in media su OpenFlow.

Un esempio pratico di un'applicazione reale del concetto e dell'architettura SDN nel contesto dei data center è stata presentata da Google. La società ha presentato all'Open Networking Summit un' implementazione su larga scala di una rete basata su SDN che collega i suoi data center. Il lavoro in [17] presenta in dettaglio la progettazione, attuazione, e la valutazione di B4, una WAN che collega i data-center di Google in tutto il mondo. Questo lavoro descrive una delle più grandi implementazioni SDN. La motivazione di tale progetto era la necessità di personalizzare l'instradamento e la gestione del traffico e insieme alla considerazione che il livello di scalabilità, fault tolerance, l'efficienza dei costi e il controllo richiesto, non poteva essere raggiunta mediante un tradizionale architettura WAN. Quindi, fu proposta una soluzione personalizzata con un'architettura basata su SDN/OpenFlow. Dopo tre anni in produzione, B4 ha dimostrato di essere efficiente, infatti molti dei collegamenti instaurati tra i nodi di differenti data-center si avvicinano al 100% di utilizzo.

Inoltre, l'esperienza riportata nel lavoro ha sottolineato l'importanza di ridurre i fenomeni come colli di bottiglia risultanti dalla comunicazione tra piano di controllo e piano dei dati e l'overhead dovuto alla programmazione in hardware.

Reti Mobili

Numerosi sforzi sono stati fatti per ottenere connettività ad infrastrutture di rete ad accesso wireless, dal momento che la richiesta di connessioni di questo tipo è in costante aumento.

Il progetto OpenRoads[18], ambisce a realizzare un sistema in cui gli utenti possano muoversi liberamente e senza interruzioni attraverso differenti infrastrutture wireless che possono essere gestite da vari fornitori di servizi.

OpenRoads fornisce il controllo di rete attraverso due protocolli: per la gestione del percorso dati si utilizza OpenFlow, invece per il controllo delle configurazioni del dispositivo viene utilizzato SNMP (Simple Network Management Protocol).

Attraverso la combinazione di questi due protocolli e grazie a metodi di astrazione si è

facilitata la gestione di dispositivi di rete eterogenei, ovvero tecnologie wireless come WiFi e WiMAX.
Il lavoro realizzato ha fornito l'ispirazione per il successivo lavoro [85] che cerca di soddisfare i requisiti e le sfide specifiche dell'implementazione di una rete cellulare software-defined.

Un altro interessante progetto è OpenRadio [19]. L'idea chiave consiste nella separazione del piano di elaborazione e il piano dati al fine di implementare un'interfaccia modulare in grado di operare su sottoinsiemi di traffico utilizzando diversi protocolli come come WiFi, WiMAX, LTE 3GPP-Advanced, ecc. Sulla base di quest'idea, un operatore può esprimere regole di decisione e corrispondenti azioni, che vengono associate da moduli del piano di elaborazione.
Inoltre OpenRadio mira a fornire una infrastruttura per aggiornare stazioni base di sistemi wireless via software. Attualmente, i dispositivi vengono raccolti periodicamente per essere aggiornati manualmente rendendo questa operazione onerosa in termini di costi, per tale motivo gli aggiornamenti automatizzati via software risultano più adeguati.

Infine citiamo il progetto Odino[20] un prototipo di architettura SDN che semplifica la gestione dei client in una WLAN in una rete resa programmabile attraverso punti di accesso virtuale.

Sicurezza

Diversi metodi per il rilevamento di attacchi DDoS (Denial of Service) sono stati studiati[21], [22], [23]. Nel primo di questi approcci è stata proposta un'architettura di rete orientata ai contenuti. Si fonda sulla creazione di flussi in base all'identità del client e il tipo di contenuto richiesto. Un attacco DDoS viene rilevato quando il server, che fornisce un determinato tipo di contenuto, riceve più richieste del previsto, seguendo un predefinito range di richieste accettabili.

Nel secondo è stato proposto un metodo che analizzi la frequenza di traffico. Se la soglia viene superata, il controller ritiene che un attacco DDoS si stia verificando e inizia a scartare pacchetti.
Nell'ultimo metodo analizzato vengono raccolte le informazioni sul traffico e attraverso l'utilizzo di mappe auto organizzate viene classificato il traffico come normale o malevolo.

Rimanendo nell'ambito della sicurezza viene proposto VAVE[24], un'architettura basata su OpenFlow progettata per convalidare l'indirizzo di tutti i pacchetti in entrata. Quando uno switch riceve un pacchetto che non corrisponde a nessuna regola, il pacchetto viene inviato al controller e l'indirizzo di origine viene convalidato. Se viene rilevato un tentativo di spoofing, una nuova regola viene creata per bloccare il traffico.

Inoltre sono state studiate tecniche di isolamento del traffico[25]. Gli autori che propongono tale sistema sostengono che le attuali tecniche di isolamento del traffico, quali VLAN, aumentano la complessità di configurazione della rete. Essi propongono la creazione di slices di rete a un livello superiore. Secondo la loro opinione, un linguaggio di programmazione di rete dovrebbe essere in grado di creare questa slice per isolare il traffico. In questo modo, le slice sono definite ad alto livello per fare in modo di aggiungere automaticamente delle regole di inoltro agli switch.

Quindi, anche quando si parla di sicurezza, notiamo quanto i ricercatori si basino sulla capacità di elaborazione dei dati nel controller dove qualche tipo di intelligenza è aggiunta allo switch attraverso il controller.

2.4 Problematiche

Lo sviluppo della tecnologia basata su SDN e OpenFlow deve fronte ad una serie di sfide che devono essere prese in considerazione, quali: sicurezza, disponibilità, scalabilità, affidabilità, costi e compatibilità[7].

Sicurezza

Una delle principali sfide che deve affrontare la tecnologia realizzata attraverso OpenFlow è la dipendenza dal controllore. Il controllore diventa un componente cruciale e dunque rappresenta un bersaglio allettante per un eventuale attacco. Le misure di sicurezza devono garantire l'utilizzabilità del controllore e al tempo stesso devono proteggerlo da eventuali intrusi.

Anche il canale tra il controllore e gli switches può essere molto vulnerabile. Secondo le specifiche OpenFlow, il Transport Layer Security (TLS) può essere utilizzato per rendere sicura la comunicazione. Questo ulteriore livello di sicurezza non è richiesto quindi la comunicazione tra controllore e switches può avvenire *in chiaro*. L'utilizzo o meno del TLS dipende da come è stato progettato il network specifico.

Anche la flow table è un elemento soggetto a dei rischi di sicurezza, benché nessuna vulnerabilità sia stata ancora resa nota. È possibile gestire una flow table da due diversi controllori: uno di essi è hardware in produzione, mentre l'altro è solo sperimentale. Dato che quest'ultimo sarà soggetto a controlli di sicurezza più bassi, è importante accertarsi che l'integrità della tabella flow persista e che un aggiornamento dannoso, proveniente da un altro controllore, non manometta altre flow entries.

Ciò nonostante, un controllore logicamente centralizzato può avere dei vantaggi in termini di sicurezza. In un network distribuito molte vulnerabilità devono essere indirizzate a differenti protocolli ed a differenti dispositivi. Avere un controllore di software al di fuori del data plane può semplificare il modo in cui la sicurezza viene applicata.

Scalabilità

Il controllore può diventare un collo di bottiglia. Se troppi pacchetti vengono inoltrati al controllore si possono veridicare forti cali di prestazione. Una rete ben progettata deve garantire che la maggior parte del traffico sia gestita dagli switch, senza dover ricorrere continuamente all'inoltro dei dati al controllore. È anche importante saper valutare se il controllore diventerà un punto di strozzatura con l'aumento del numero dei nodi.

Le architetture basate su OpenFlow devono affrontare due importanti sfide di scalabilità: la grandezza limitata della flow table ed i vincoli imposti dall'hardware. Primo, il numero di voci che una flow table può contenere è limitato. Risulta ancora una sfida gestire un grande numero di flows utilizzando uno switch OpenFlow-compliant. Anche la gestione dei pacchetti a livello di control plane è lenta. Perciò, il controllo del traffico end-to-end è difficile da realizzare se molti e differenti flows devono essere gestiti. Secondo, ci sono limitazioni hardware circa la velocità con la quale i flows possono essere aggiunti. Per queste due ragioni non è ancora chiaro se OpenFlow potrà essere usato per controllare il nucleo di un network esteso, infatti attualmente OpenFlow è utilizzato solo ai margini della rete.

Affidabilità

La dipendenza dal controllore crea anche problemi di affidabilità. Un esempio lo si è visto nel paragrafo precedente: in un network distribuito il recupero di un percorso caduto può risultare un processo lungo.

Una proposta multipath (multi-strada) per OpenFlow risponde alla necessità di recuperare velocemente gli errori. Questa proposta include un supporto di re-indirizzamento veloce dove i flow backup possono essere utilizzati. Se lo switch rileva che una specifica porta ha perso connettività, allora viene utilizzato il flow backup. Questo è un modo proattivo di trattare errori di collegamento ed ha il vantaggio di non aver bisogno di contattare immediatamente il controllore quando si ha l'errore.

CAPEX /OPEX

E' stato discusso se OpenFlow può ridurre le spese capitali ed operazionali (CAPEX e OPEX) di un'organizzazione.

Coloro che adottano OpenFlow affermano che spostando la complessità del controllo verso un software renderà i problemi di network più facili da risolvere e probabilmente si avranno degli effetti positivi in termini di spesa in quanto questo ridurrebbe il CAPEX. Detto ciò, bisogna tener presente che OpenFlow ha anche dei limiti ed un hardware avanzato è ancora richiesto per gestire un network. Per ora non sembra verosimile uno scenario in cui gli attuali apparati di rete vengano semplificati a tal punto da fungere solo come elementi di inoltro.

In più bisogna garantire la disponibilità di rete attraverso un piano di controllo accuratamente studiato, infatti è importante che il controllore rimanga raggiungibile anche nel caso di errori nel piano di controllo. Questo raggiungimento può aumentare i costi di uno sviluppo in tal senso.

Un simile problema si verifica anche per l'OPEX. Di certo OpenFlow può essere utilizzato per ridurre il numero di attività di configurazione basate sulle risorse umane che sono dispendiose in termini di tempo e soggette ad errori e questo ridurebbe l'OPEX. D'altro canto, spostare la complessità del network al piano di controllo del software richiede lavoro. Gli amministratori di progetto, gli sviluppatori del software, i testers, i debuggers ed altri costi sono esempi delle spese che possono incorrere in un progetto basato su OpenFlow.

Compatibilità

Un'altra importante sfida per OpenFlow è che i sistemi operativi di rete supportano solamente alcune versioni della specifica OpenFlow. Attualmente la maggior parte di loro non supporta le caratteristiche della specifica nella versione più nuova. La sfida è quella dunque di aggiornare sia la specifica OpenFlow che il software dei controller.

Questo problema di compatibilità ricorre anche nei dispositivi di network, i cui software devono essere aggiornati per soddisfare i requisiti delle nuove specifiche OpenFlow.

CONCLUSIONI

In questa tesi è stato esaminato il tema del Software-Defined Networking ed è stato approfondito il protocollo OpenFlow con lo scopo di fare maggiore chiarezza su questi argomenti e dare una valutazione complessiva sulla base degli articoli trovati nella letteratura disponibile.

Si è dato uno sguardo alla storia, all'architettura, sono stati presentati esempi e sono state descritte alcune delle applicazioni sviluppate intorno a tale paradigma.
Durante lo studio di tali argomenti sono emersi i vantaggi che porterebbe l'adozione di tale tecnologia e come questi attraggano gli interessi sia dell'industria che della ricerca. La prima sta supportando il suo sviluppo poiché crede che porterà ad una facilitazione nella gestione della rete rendendola più semplice e personalizzabile. Inoltre si prevede che tale tecnologia porterà ad una riduzione delle spese operative e maggiori ricavi grazie all'offerta di nuovi servizi.
Anche la comunità dei ricercatori è favorevole al suo sviluppo poiché considera SDN un'opportunità per dare un contributo all'innovazione sperimentando soluzioni che prima erano impossibili da testare.

Tuttavia rimangono delle problematiche ancora da risolvere. Sopratutto, bisogna tener presente che la maggior parte delle applicazioni realizzate per SDN/OpenFlow hanno preso in considerazione reti molto limitate con un numero ridotto di switch e solo pochi studi hanno dimostrato il loro funzionamento su reti più estese. Per cui una delle maggiori sfide che dovrà affrontare sarà sicuramente quella di dimostrare il suo funzionamento su reti WAN (Wide Area Network).

Sicuramente lo sviluppo del Software-Defined Networking e del protocollo OpenFlow rappresenta un tentativo concreto di reagire a questo momento di stasi per l'innovazione nel campo delle reti, d'altra parte bisognerà attendere la sua completa maturazione affinché possa imporsi realmente e permettere di fare un passo avanti verso la rete del futuro.

BIBLIOGRAFIA

[1] Arpanet - https://it.wikipedia.org/wiki/ARPANET

[2] M. Mendonca: A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, IEEE Communications Surveys & Tutorials, 2013, pp. 1-2

[3] Open Networking Foundation - https://www.opennetworking.org

[4] Open Networking Foundation: Software-Defined Networking: The New Norm for Networks, ONF White Paper, 2012

[5] Nick Feamster, Jennifer Rexford, Ellen Zegura, The Road to SDN: An Intellectual History of Programmabl Networks

[6] GENI. Campus OpenFlow topology, 2011. http://groups.geni.net/geni/wiki/OpenFlow/CampusTopology.

[7] A. Lara, A. Kolasani, B. Ramamurthy: Network Innovation using OpenFlow: A Survey, IEEE Communications Surveys & Tutorials, 2013.

[8] Open Networking Foundation: OpenFlow Switch Specification, ONF, 2014. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

[9] Antonio Manzalini, Vinicio Vercellone and Mario Ullio, “Software Defined Networking: sfide e opportunit`a per le reti del futuro”, Notiziario tecnico Telecom Italia, No. 1, pp. 30-43, 2013.

[10] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner, OpenFlow: Enabling Innovation in Campus Networks March 14, 2008

[11] Nikhil Handigol, Srinivasan Seetharaman Mario Flajslik, Nick McKeown, Ramesh Johari - Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow

[12] N. Handigol, S. Seetharaman, M. Flajslik, A. Gember, N. McKeown, G. Parulkar, A. Akella, N. Feamster, R. Clark, A. Krishnamurthy,V. Brajkovic, and T. Anderson, “Aster*x: Load-Balancing Web Traffic over Wide-Area Networks,” 2011

[13] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, “Software defined networking: Meeting carrier grade requirements,” in 18th IEEE Workshop on Local Metropolitan Area Networks (LAN-MAN), 2011

[14] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, “ElasticTree: saving energy in data center networks,” 2010

[15] H. Shirayanagi, H. Yamada, and K. Kono. Honeyguide: A vm migration-aware network topology for saving energy consumption in data center networks. In Computers and Communications (ISCC), 2012

[16] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yala-gandula, Puneet Sharma, and Sujata Banerjee. Devoflow: scaling flow management for high-performance networks. SIGCOMM Comput.

[17] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. pages 3–14. ACM, 2013.

[18] K.K. Yap, M. Kobayashi, R. Sherwood, T.Y. Huang, M. Chan, N. Handigol, and N. McKeown. Openroads:

Empowering research in mobile networks. , 2010.

[19] M. Bansal, J. Mehlman, S. Katti, and P. Levis. Openradio: a programmable wireless dataplane. In Proceedings of the first workshopon Hot topics in software defined networks, pages 109–114. ACM, 2012.

[20]L. Suresh: Towards Programmable Enterprise WLANs with Odin, Proceedings of the first workshop on Hot topics in software defined networks, 2012

[21]J. Suh, H. Choi, W. Yoon, T. You, T. Kwon, and Y. Choi, "Implementation of a Content-oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure," in European NetFPGA Developer Workshop, 2010

[22] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in 2010 IEEE 35th Conference on Local Computer Networks (LCN), October 2010.

[23] Y. Chu, M. Tseng, Y. Chen, Y. Chou, and Y. Chen, "A novel design for future on-demand service and security," in 12th IEEE International Conference on Communication Technology (ICCT), 2010.

[24]G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," in 19th IEEE International Conferenceon Network Protocols (ICNP), 2011

[25] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation:a slice abstraction for software-defined networks," in Proc. FirstWorkshop on Hot Topics in Software Defined Networks (HotSDN),2012