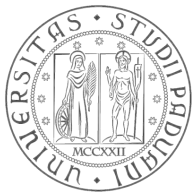

Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Studio e Progettazione di un'esperienza didattica su OpenFlow

Laureando

Stefano Zattara

Relatore

Prof. Andrea Zanella

Anno Accademico 2013/2014

A tutte le persone che mi hanno spinto ad arrivare fin qui...
ed a tutte quelle che credevano non ci riuscissi.

Zattara Stefano

Abstract

Nelle attuali strutture di rete vengono utilizzati principalmente apparati switch e router.
Gli switch sono molto veloci ad eseguire operazioni elementari e relativamente economici, d'altra parte i router sono più costosi e mediamente più lenti ma possono eseguire praticamente qualunque operazione sui pacchetti in transito.

OpenFlow si prefissa lo scopo di uniformare la struttura di rete, togliendo a router e switch l'onere di decide come trattare i singoli flussi di dati sfruttando solamente la loro struttura per l'inoltro dei pacchetti, e centralizzando su uno o più apparati intelligenti detti "controller", tutta la logica di gestione della rete, delegando ad applicazioni esterne le regole da applicare ai singoli pacchetti, trasformando di fatto la rete in una S.D.N. (Software Define Network).

Scopo della tesi è illustrare il funzionamento ed i vantaggi di OpenFlow, realizzare un ambiente di prova che permetta la sperimentazione di OpenFlow con switch virtuali sfruttando il modulo Openvswitch del kernel Linux, e con router fisici compatibili con OpenFlow.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Attuali strutture di rete

Le aziende, in un'ottica di ottimizzazione dei costi, sfruttano da un decennio le grandi e molteplici potenzialità messe a disposizione dai computer e recentemente dalla rete in generale. Dapprima per rete si intendeva un computer connesso ad internet, con servizi offerti da un provider esterno; l'azienda era considerata una semplice fruitrice di tali servizi (si pensi alle prime caselle di posta elettronica ed i primi siti web). Con il progresso tecnologico sviluppatosi negli ultimi anni, ogni media/grossa azienda è potenzialmente diventata un piccolo provider: server di posta interni per la posta aziendale, sito internet, e-commerce, database, terminali remoti ecc.
L'aspetto networking sta diventando per le aziende sempre più cruciale, al punto che un potenziale problema di rete può paralizzare completamente l'azienda, con danni economici non indifferenti.

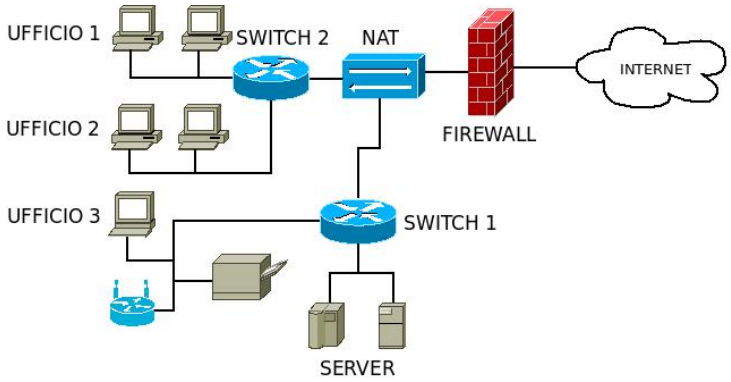

Figura 1.1: Schema di una generica Rete Aziendale

1.1 Tipologie ed apparati di rete

Mediamente in ogni azienda è presente almeno un router, uno o più switch e dei dispositivi intelligenti, siano essi palmari, tablet, personal computer, server o altro. Considerato il mercato in continua espansione, ogni grossa azienda di hardware di rete ha iniziato a costruire degli apparati più o meno sofisticati per il networking quali hub, switch, router, sistemi anti intrusione (Intrusion Prevention System o IPS), sistemi di identificazione degli accessi non autorizzati (Intrusion Detection System o IDS), Firewall. Ogni azienda ha però creato il proprio standard, integrato su hardware proprietario ed ha venduto queste "scatole nere" (black-box) mettendo a disposizione dell'utilizzatore delle interfacce per la configurazione senza però permetterli di sapere realmente come lavora l'hardware ed il software del dispositivo.

1.2 Configurazione, sicurezza, subnet

Un'azienda di grandi dimensioni, data la necessità di collegare numerosi dispositivi, richiederebbe l'utilizzo uno spazio di indirizzamento sufficientemente ampio, la cui gestione viene semplificata dal partizionamento in sottoreti. Questo genera un'intrinseca sicurezza da un lato, in quanto le sottoreti non sono in grado di comunicare direttamente tra loro, ma hanno bisogno di apparati intelligenti (router o switch di livello 3) per permettere il corretto instradamento dei pacchetti tra le sottoreti. In definitiva in ogni grossa azienda possiamo trovare un modello in miniatura della rete internet, composta da router, switch, server, firewall e con tutti i problemi ad essi collegati.

1.3 Innovazione nei servizi: La Virtualizzazione

Un'innovazione importante nell'ambito dei servizi è stata l'introduzione della virtualizzazione e della para-virtualizzazione. Questa innovazione ha permesso di creare reti fittizie, esistenti solo a livello logico. Le reti virtuali (virtual network), sono l'insieme dei nodi virtuali (virtual nodes; i nodi che compongono la rete, siano essi server, switch o router) e dai collegamenti virtuali (virtual links ovvero i collegamenti tra i vari nodi). La virtualizzazione consente di offrire contemporaneamente servizi diversi, divisi a livello logico ma ospitati fisicamente sullo stesso hardware. Tutta la configurazione viene fatta a livello utente, indipendentemente dall'hardware su cui si sta lavorando. Si pensi ad esempio a VmSphere come interfaccia di VmWare[1], a OpenVz[2] per la paravirtualizzazione su kernel Linux oppure a servizi esterni come quelli offerti da Amazon[3].
Questo sistema permette di avere un'astrazione tra il livello fisico (l'hardware su cui viene eseguito il processo) ed il livello utente, offrendo la possibilità di effettuare spostamenti di macchine virtuali, backup e modifiche della configurazione, in modo completamente indipendente dall'hardware su cui viene eseguito il processo.

1.4 Innovazione nei servizi: Il Cloud

Con lo sviluppo delle reti virtuali si è espanso anche il cosiddetto Cloud, ovvero un datastorage distribuito che permette agli utenti di caricare e scaricare files, appoggiandosi ad una serie di server sparsi per mondo. Alcuni software di cloud permettono di sincronizzare file su diversi dispositivi quali pc, laptop, cellulari ecc..., quasi in tempo reale in qualsiasi luogo ci si trovi. Come conseguenza si possono avere dei lunghi periodi in cui il carico del cloud è basso ed altri periodi di intensa attività dove viene richiesta molta banda internet ed altrettante risorse.

1.5 Problematiche dell'attuale architettura

La crescita tecnologica esponenziale a cui stiamo assistendo ha stimolato la definizione di standard, spesso proprietari, per la parte di sviluppo e la

[1] Si veda `http://www.vmware.com/it` per maggiori informazioni
[2] Si veda `http://openvz.org/Main_Page` per maggiori informazioni
[3] Si veda `https://aws.amazon.com/ec2/` per maggiori informazioni

gestione dell'hardware, di conseguenza ogni apparato hardware ha un suo linguaggio e una sua interfaccia di amministrazione. Raramente apparati di marche diverse sono "compatibili" tra loro. Questo significa che nel caso di un guasto ad un apparato, normalmente si ricorrerà alla sostituzione con un apparato uguale, anche se ormai tecnologicamente superato, perché questo significa accelerare i tempi di configurazione, conoscendo già l'interfaccia di amministrazione, oppure poter ripristinare un eventuale configurazione salvata in precedenza, ottimizzando i tempi di intervento.
Ipotizzando, invece, una sostituzione programmata di uno switch (ad esempio per aggiunta di porte o per cambio tecnologia), nel farlo è bene tener conto di tutti i processi che lo switch esegue e quindi configurare con i nuovi strumenti le funzionalità sul nuovo apparato (ad esempio ACLs[4], QoS[5], ecc.). Similmente nel corso di un ampliamento della struttura, si ricorrerà all'utilizzo di hardware conosciuto, anche se datato.

Per come è nato internet, il routing è stato pensato analizzando solamente la destinazione dei pacchetti in transito; i vari protocolli di routing, tra cui il RIP (Routing Information Protocoll), OSPF (Open Shortest Path First), BGP (Border Gateway Protocol) si sono evoluti in modo naturale per ottimizzare il percorso in cui instradare il pacchetto ed avere al loro interno tabelle di routing da consultare più snelle e performanti. Attualmente però il grosso problema è che spesso c'è necessità di prendere una decisione su dove instradare il pacchetto, non solo dalla destinazione, ma anche dall'ip sorgente o dalla porta di destinazione.
Si immagini, ad esempio, un'azienda con una linea wireless licenziata (con bassissima latenza ma poca banda) ed una normale linea xDSL. Molto probabilmente si vorrà utilizzare la connessione wireless per tutte le connessioni che hanno bisogno di poca latenza, ad esempio il VoIP, mentre per tutti gli altri servizi internet, l'altra linea.
Per fare ciò occorre creare un filtro che raccolga tutto il traffico con porta destinazione 5060 e di tipologia UDP e lo instradi su una rotta non di default. Questo processo di politiche di instradamento (routing policy) richiede molte risorse di calcolo dalla CPU, in quanto bisogna analizzare ogni singolo pacchetto che transita. Ci si trova però di fronte ad apparati di livello due[6]

[4]Liste di accesso (Access Control List) sono le regole del firewall

[5]Qualità del Servizi (Quality of Service) è un sistema per prioritizzare i pacchetti in modo che determinati pacchetti più "importanti" raggiungano sempre la destinazione nel tempo stabilito, indipendentemente dal carico del collegamento

[6]A livello due del modello ISO/OSI si trova il livello di DataLink, ovvero a livello MAC Address

molto veloci quali gli switch ed apparati di livello tre[7] più intelligenti ma lenti quali i router.

[7]A livello tre del modello ISO/OSI si trova il livello di Rete, ovvero a livello in cui si utilizzano gli indirizzi IP

Capitolo 2

L'astrazione del modello di rete: le S.D.N.

In questo capito si definiscono le Software Define Network o SDN[1, 2], il concetto di flusso (flow) ed i relativi vantaggi.

2.1 Software Define Network

Come è stato fatto con la virtualizzazione per le macchine fisiche (Virtual Machines o VM), con i collegamenti virtuali (Virtual Link o VL) e con gli apparati di rete (Virtual Nodes o VN), si è cercato di virtualizzare anche tutta la struttura di rete. Si è cercato di creare una struttura di controllo che sia universale, un protocollo condiviso per poi lasciare ai costruttori di hardware l'implementare nei loro apparati il protocollo in modo autonomo. Un'idea nata nel 2007 per cercare di uniformare una generica rete è quella del disaccoppiamento del piano di instradamento, tipicamente realizzato in hardware dal costruttore, da quello di controllo. Si pensava cioè di sfruttare la potenza intrinseca di uno switch nel manipolare i pacchetti, lasciando però ad un controller esterno il decidere come trattare ogni singolo flusso di dati (bloccarlo, modificarlo, instradarlo ad un altro gateway etc.) Il primo protocollo così sviluppato nel 2008 è stato OpenFlow e con il sistema operativo NOX nascono le prime Software Define Networks.

2.2 Il concetto di Flow

Openflow basa il suo principio di funzionamento sui cosiddetti flussi (flow). Per flow si intende letteralmente un flusso, ovvero una sequenza di pacchetti in transito da un generico dispositivo A ad un dispositivo B. Il flusso è unidirezionale; non è previsto cioè che eventuali regole che gestiscono il traffico da A a B gestiscano anche per il traffico di ritorno da B ad A.

2.3 La struttura

L'innovazione nelle SDN sta nel disaccoppiamento tra il controllo della rete e la parte fisica del dispositivo di rete. Tutti i dispositivi che compongono la rete (siano essi switch o router) vengono visti nello stesso modo da un dispositivo superiore, definito "controller". Ogni apparato connesso alla rete rende disponibile tramite un canale sicuro (tipicamente un collegamento in TLS[1]) l'accesso alle sue tabelle di flusso, tramite un protocollo; nel nostro caso openflow. Le tabelle di flusso sono delle tabelle, (realizzare di solito come CAM o TCAM[3]) che l'apparato consulta prima di instradare un pacchetto. Data la natura costruttiva di queste tabelle che si prestano bene a fare la ricerca di una stringa in una tabella (pattern matching), è facile realizzare

[1] `http://it.wikipedia.org/wiki/Transport_Layer_Security`

firewall, policy di routing, QoS e statistiche sui pacchetti eseguendole però alla velocità di switching dell'apparato. Viene lasciato poi al costruttore l'interfacciamento tra le tabelle di flusso e l'hardware del dispositivo consentendogli di implementare l'hardware secondo i suoi standard, in non si và ad intaccare la sua "black-box"[2].

Il **Piano di Controllo** (Control Plane) è formato dal Sistema Operativo di Rete e dalle applicazioni che vengono eseguite. Ogni switch può essere collegato a più controller ed i controller collegati tra loro, in modo da avere maggiore affidabilità della struttura. Le decisioni sull'instradamento vengono prese dalle applicazioni che comunicano con i vari switch tramite il Sistema Operativo di Rete, sfruttando delle API[3] messe a disposizione dallo stesso.

Il **Piano di Instradamento** (Forwarding Plane) è formato invece da tutti i dispositivi di livello fisico, ossia gli switch ed i router. I due piani vengono messi in comunicazione da un protocollo, openflow appunto.

2.4 Il Controller

Il controller in una SDN è il software che è in esecuzione su un dispositivo (normalmente un pc/server) che si occupa di gestire gli switch. É lo switch che si collega al controller la prima volta, dopodiché il controller può effettuare delle azioni di lettura e scrittura nelle tabelle di flusso dello switch. Il controller altro non è che l'interfaccia messa a disposizione per accedere agli switch; è costituito genericamente da un software basilare, che normalmente non prende nessuna decisione sui pacchetti ma mette a disposizione delle API per interfacciare gli switch a delle applicazioni esterne che prenderanno le decisioni su come trattare i pacchetti, e tramite il controller, comunicheranno l'istruzione al singolo switch incaricato di gestire il pacchetto.

[2] Con il termine "black-box" si intende un apparato di cui si conosce l'uscita dato un ingresso, ma si ignora completamente il metodo di funzionamento

[3] API: Interfaccia di programmazione per l'applicazione (Application Programming Interface) `http://it.wikipedia.org/wiki/Application_programming_interface`

Capitolo 3

OpenFlow switch

In questo capitolo si espongono le varie tipologie di switch utilizzabili con OpenFlow [4] e la sua struttura a livello di porte, pipeline e tabelle.

3.1 Tipologia di switch OpenFlow

Esistono due tipologie di switch openflow, gli switch esclusivamente openflow (*Dedicated OpenFlow Switch*) e gli switch compatibili con openflow (*OpenFlow-Enable Switch*).

3.1.1 Switch Dedicati OpenFlow (*Dedicated OpenFlow switches*)

Gli switch che supportano solamente openflow possono essere visti come una serie di percorsi per i dati tra le loro porte. Questo percorso viene impostato dal controller. Ogni switch deve poter eseguire:

- **Indirizzare un pacchetto su una determinata porta (o un gruppo di porte)**.
  Questa è la caratteristica fondamentale per poter mettere in comunicazione due o più dispositivi collegati. Nella maggior parte degli switch ci si aspetta che funzioni alla velocità di collegamento delle porte.

- **Incapsulare e instradare un pacchetto verso il controller**.
  Il pacchetto è consegnato attraverso il canale crittografato al controller. Tipicamente è quello che succede per ogni primo pacchetto di un nuovo flusso; lo switch invia il pacchetto al controller che lo analizza e decide come agire di conseguenza (aggiungere flusso, togliere flusso, ignorare il pacchetto o reinstradarlo verso un altro controller).

- **Scartare un pacchetto in modo silenzioso** (DROP).
  Questo può essere usato per motivi di sicurezza o per ridurre/eliminare problemi dovuti al traffico broadcast.

3.1.2 Switch Compatibili OpenFlow (*OpenFlow-enable switches*)

Questi sono gli switch che supportano openflow come "applicazione" da abilitare al loro interno. In questo caso, le caratteristiche degli switch sono le stesse degli switch dedicati openflow, con l'aggiunta di un quarto punto:

- **Instradare il pacchetto verso il normale processo di switching dello switch**.
  In questo modo, per alcuni pacchetti si può decidere che vengano trattati come se si usasse un normale switch.

3.2 Tipologia di porte OpenFlow

Le porte openflow sono le interfacce di rete che permettono il passaggio dei pacchetti attraverso l'apparato openflow. I pacchetti sono ricevuti su porte di ingresso (*ingress port*), processati ed indirizzati su porte di uscita (*output port*). Queste porte possono essere di 4 tipi:

3.2.1 Porte Fisiche

Le porte definite come Porte Fisiche (*Physical Port*) sono le porte corrispondenti alle interfacce hardware dello switch, indipendentemente esse siano ethernet, V35, SFP etc. In alcune applicazioni dove lo switch openflow è in realtà composto da diversi switch fisici, le porte fisiche possono corrispondere ad una sezione di porte hardware dello switch.

3.2.2 Porte Logiche

Le porte definite come Porte Logiche (*Logical Port*) sono le porte che non corrispondono direttamente ad interfacce dello switch. Le porte logiche sono un'astrazione ad alto livello di strutture dello switch non openflow (ad esempio il link aggregation, tunnel, interfacce loopback, bounding, failover etc...)

3.2.3 Porte Risevate

Le porte definite come Porte Riservate (*Reserved Port*) sono porte non fisiche a cui è associata un'azione speciale, ad esempio di inoltrare il pacchetto al controller, oppure passare il pacchetto al piano di switching tradizionale dello switch. Gli switch non devono necessariamente supportare tutte le porte riservate, ma solamente quelle marcate come *Richiesta*.

- *Richiesta* **ALL**: Rappresenta tutte le porte su cui lo switch può inoltrare un pacchetto. Nel caso venga richiesto allo switch di inoltrare un pacchetto su questa porta, lo inoltrerebbe su tutte le porte standard (si veda il cap. 3.2.4) escludendo la porta da cui è arrivato il pacchetto.

- *Richiesta* **CONTROLLER**: Rappresenta il canale di connessione con il controller. Un pacchetto che esce da questa porta viene incapsulato ed inoltrato al controller, un pacchetto in ingresso su questa porta è un pacchetto che arriva dal controller.

- *Richiesta* **TABLE**: Rappresenta l'inizio della pipeline di openflow. Può essere utilizzata solo come porta di output ed indica che il pacchetto deve essere trattato secondo la pipeline di openflow.

- *Richiesta* **IN_PORT**: Rappresenta la porta da cui è arrivato il pacchetto, può essere usata solo come porta di uscita e rimanda il pacchetto sulla porta da cui è arrivato.

- *Richiesta* **ANY**: E' un valore speciale dei comandi openflow che si attiva quando non viene specificata una porta di ingresso o di uscita. Non può essere usato direttamente nè come porta in ingresso, nè come porta di uscita.

- *Opzionale* **LOCAL**: Rappresenta l'interfaccia locale dello switch ed il suo stack di amministrazione.

- *Opzionale* **NORMAL**: Rappresenta la pipeline tradizionale dello switch. Può essere usata solo come porta di output e serve per consegnare un pacchetto alla pipeline tradizionale dello switch, per essere trattato secondo le normali regole di switching. Se lo switch non supporta una pipeline tradizionale, o non è in grado di passare un pacchetto da una pipeline all'altra, deve specificare che non supporta questa azione.

- *Opzionale* **FLOOD**: Rappresenta una porta associata alla pipeline fisica dello switch. Inoltra il pacchetto a tutte le porte standard dello switch.

Gli switch openflow nativi non hanno le porte **NORMAL** e nemmeno le porte **FLOOD** in quanto sono porte legate alla pipeline non openflow.

3.2.4 Porte Standard

Le porte definite come Porte Standard (*Standard Port*) sono tutte le porte fisiche, logiche e la porta riservata **LOCAL**. Tutte le porte standard possono essere usate come porte di ingresso o di uscita, possono essere usate in gruppi ed hanno un contatore dedicato.

3.3 Canale di Comunicazione Sicuro

La connessione tra il controller e lo switch viene effettuata normalmente tramite un canale crittografato, normalmente via TLS (Transport Layer Security). La connessione viene instaurata dallo switch su **una porta non**

dedicata ad openflow, verso il controller sulla porta TCP 6633. L'autenticazione viene fatta tramite certificato e lo switch, per supportare il TLS, deve permettere il salvataggio al suo interno della chiave privata del certificato. Nel caso di connessioni in chiaro, il canale che collega lo switch al controller diventa critico per il funzionamento e per la sicurezza dell'intera rete. É possibile che in una rete vi siano più controller per garantire una stabilità nella rete. In tal caso lo switch si collega a tutti i controller configurati e cerca di mantenere la connessione con tutti. É compito del controller gestire ad esempio una gerarchia di primario/secondario o di fast-failover nel caso di più controller nella stessa rete.

3.4 Pipeline OpenFlow

Nel caso di switch openflow puri esiste solamente la pipeline openflow, negli altri switch ibridi esistono due pipeline distinte, una che segue il percorso di openflow, l'altra che procede con il normale switching.
Nel caso di switch ibridi, lo switch deve mettere a disposizione un criterio per la classificazione del traffico, ad esempio l'uso di una specifica VLAN o l'assegnazione delle porte, in modo da dividere il traffico che và processato con openflow rispetto a quello che segue la normale sequenza di switch hardware dell'apparato. Questa distinzione è esterna ad openflow ed è curata interamente dal software dello switch. Openflow, sfruttando la porta riservata "NORMAL" può inoltrare di nuovo il pacchetto verso la pipeline dello switch. Il contrario non è invece possibile.
La generica pipeline di openflow si compone di diverse tabelle di flusso numerate in sequenza partendo da 0. Ogni tabella di flusso contiene diverse voci. Ogni switch openflow deve contenere almeno una tabella di flusso ed almeno una voce (*entry*). Il confronto (*match*) delle voci parte sempre dalla prima voce della tabella 0; se il pacchetto corrisponde, viene eseguita l'azione corrispondente, altrimenti continua la scansione nelle successive voci. L'istruzione può effettuare delle modifiche al pacchetto (ad esempio l'aggiunta di un tag VLAN), indirizzarlo direttamente verso una porta di uscita, oppure indirizzarlo verso un'altra tabella di flusso successiva (cioè con id maggiore di quella di provenienza), dove viene eseguito lo stesso procedimento (le tabelle di flusso sono realizzate normalmente con dispositivi TCAMs[3]). Se non viene trovata nessuna corrispondenza nelle tabelle di flusso viene eseguita quella di default, chiamata anche tabella di flusso mancante (*table-miss flow*, una sorta di default gateway), dove si decide se il pacchetto verrà scartato, inoltrato alla pipeline dello switch standard o inoltrato al controller.

Match Field	Priority	Counters	Instructions	Timeout	Cookie

Tabella 3.1: Composizione flow tables OpenFlow

3.5 Tabelle di Flusso

Ogni tabella di flusso (table-flow) è composta come in Figura 3.1 in questo modo:

- **Match Filed**: il campo su cui viene fatto il confronto del pacchetto. I campi contenuti sono la porta di ingresso, l'headers del pacchetto e facoltativamente dei metadata, ovvero dei dati usati per passare informazioni tra una tabella ed un altra.
- **Priority**: la priorità della voce.
- **Counters**: dei contatori, vengono aggiornati ogni qualvolta un pacchetto combacia con il *match field* e quindi viene processato dal sistema.
- **Instructions**: le istruzioni da eseguire.
- **Timeout**: massimo intervallo di tempo o di inattività prima che la voce venga cancellata dallo switch.
- **Cookie**: un hash[1] che utilizza il controller per identificare velocemente la voce. Questo campo non viene usato nel confronto del pacchetto.

Ogni voce viene identificata univocamente dal campo *match field* e dal campo *priority*. La voce con un carattere jolly (wildcard) in tutti i campi (ovvero tutti i campi sono omessi), ha priorità 0 ed è chiamata voce di default (*table-miss flow enty*), ovvero la riga che viene eseguita per ultima se nessun'altra voce ha una corrispondenza. Ogni voce inserita nella tabella ha due parametri associati *hard_ timeout* e *idle_ timeout*.
Se il parametro *hard_ timeout* è impostato ad un valore diverso da zero, la voce verrà rimossa dallo switch dopo il numero dato di secondi. Se il parametro *idle_ timeout* è impostato ad un valore diverso da zero, la voce verrà rimossa dallo switch dopo il numero dato di secondi che nessun pacchetto effettua un confronto con esisto positivo con la relativa voce della tabella. Una

[1]La funzione di hash è una funzione che mappa una serie di caratteri di lunghezza variabile in una stringa di valore definito, per informazioni `http://it.wikipedia.org/wiki/Hash`

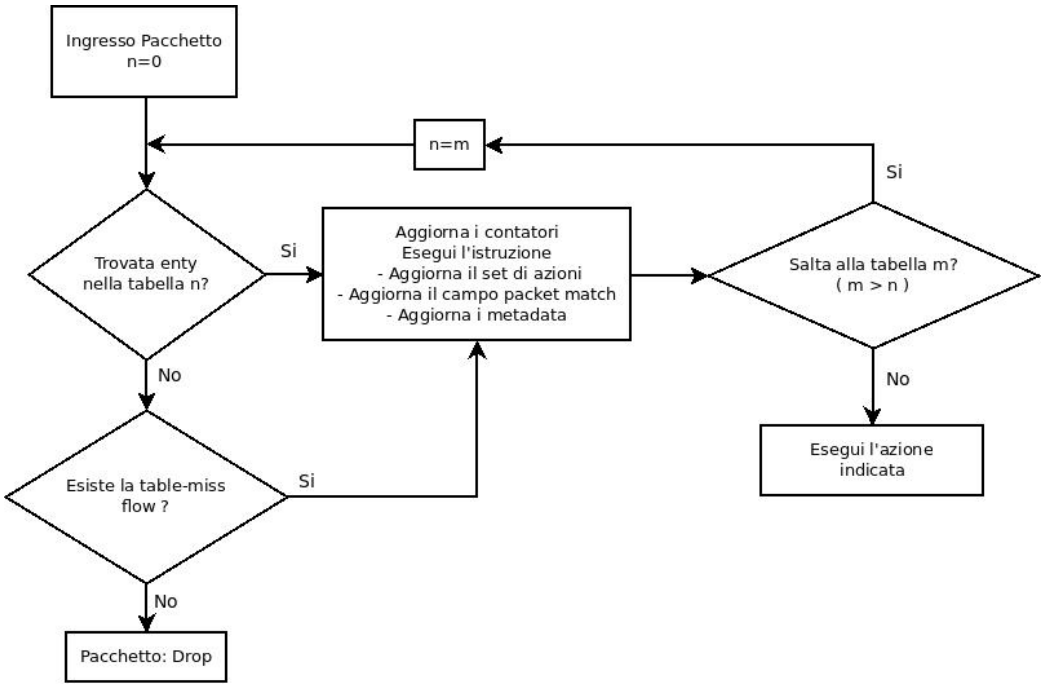

Figura 3.1: Flowchart OpenFlow packet matching

volta che la voce è stata rimossa, lo switch invia al controller un messaggio di rimozione della voce. Ogni messaggio contiene per intero la descrizione della voce rimossa, la ragione della rimozione e tutti i parametri (conteggio dei pacchetti, durata in secondi e statistiche) del pacchetto.

3.5.1 Packet Matching

Il processo di abbinamento (matching) avviene come nella Figura 3.1.
All'ingresso del pacchetto viene effettuata una ricerca nella prima tabella (la tabella 0). Se viene effettuato un abbinamento vengono eseguite le istruzioni dichiarate nel campo istruzioni; nel caso all'interno del campo vi sia un'istruzione di salto in un'altra tabella, questa viene eseguita e viene effettuato il confronto con le voci nella tabella indicata, se invece non vi è alcun salto, il pacchetto viene modificato secondo quanto richiesto ed istradato sulla porta di uscita. Nel caso durante la ricerca si trovino due o più voci, verrà eseguita solo quella con priorità più alta. Nel caso invece non si trovi alcun match nelle tabelle, viene cercato ed eseguito il *table-miss flow enty*, se non si trova nemmeno questa voce, il pacchetto verrà scartato.

Group Identifier	Group Type	Counters	Action Buckets

Tabella 3.2: Struttura Group Table OpenFlow

3.5.2 Table Miss

Ogni tabella di flusso deve supportare la voce di table-miss. Questo flusso ha per definizione tutti i parametri di match a ANY, priorità 0 ed un'azione predefinita che può essere quella di inoltrare il pacchetto al controller tramite la porta riservata CONTROLLER, scartato, oppure inoltrato su un'altra porta.

3.6 Tabelle di Gruppo

Le tabelle di gruppo (*Group Table*), composte come in Tabella 3.2 consistono in gruppi di regole. La possibilità di far eseguire ad un confronto un gruppo di azioni permette ad openflow di eseguire operazioni del tipo "uno a molti". Ogni Group Table contiene i seguenti campi:

- **Group Identifier**: un campo contenente un intero senza segno a 32 bit che identifica univocamente il gruppo.
- **Group Type**: determina la tipologia di azione da applicare al gruppo.
- **Counters**: il numero di pacchetti processati dal gruppo (un contattore).
- **Action Buckets**: una lista ordinata di istruzioni da eseguire.

3.6.1 Tipologie di Gruppi

- *Richiesto* **ALL**: eseguire tutte le azioni nel gruppo. Solitamente questa istruzione viene usata per gestire il traffico broadcast o multicast. Il pacchetto viene clonato e gestito da ogni sequenza di azioni.
- *Opzionale* **SELECT**: esegue solamente un'azione del gruppo. La selezioni dell'azione può essere eseguita in round-robin[2] tra quelle disponibili, oppure decisa da un processo esterno allo switch. Nel caso le azioni siano di output e una delle porte designate cambia lo stato in down, questa viene automaticamente esclusa dalla scelta.

[2] `http://it.wikipedia.org/wiki/Scheduling#RR`

Meter Identifier	Meter Bands	Counters

Tabella 3.3: Composizione meter tables OpenFlow

- *Richiesto* **INDIRECT**: esegue l'unica azione definita dal gruppo. Il gruppo supporta solamente un'azione. Ad esempio il traffico da inoltrare ad un "next-hop".
- *Opzionale* **FAST FAILOVER**: esegue la prima azione considerata operativa nel gruppo. Ogni azione è associata ad una specifica porta o gruppo che ne controlla lo stato. Questo gruppo permette di effettuare uno switch tra più porte senza dover interrogare il controller. Se tutte le porte si trovano nello stato di down, il pacchetto viene scartato senza generare errori.

3.7 Meter Table

La meter table definisce la banda associata ad ogni flusso. Questa tabellla viene utilizzata per implementare il QoS a livello di switch di rete. La Meter Table misura la banda consumata dai pacchetti associati al flusso. Più flussi possono essere associati alla stessa voce nella tabella, ma ogni flusso può essere associato ad una sola voce. La tabella è strutturata come in Figura 3.3.

- **Meter Identifier**: un campo che contiene un interno senza segno a 32 bit che identifica la regola.
- **Meter Bands**: una lista di profilazioni, dove ogni limite indica la banda associata ad ogni flusso.
- **Counters**: aggiornato ogni qualvolta un pacchetto effettua un confronto con esito positivo.

3.8 Contatori

Ogni tabella di flusso, porta, coda e gruppo ha dei suoi specifici contatori.
Secondo le specifiche questi contatori possono essere implementati via software in modo da rendere più semplice la parte hardware.
In tabella 3.4 e 3.5 sono riportati i contatoti secondo le specifiche openflow.
Tutti i contatori sono gestiti come interi senza segno. Se un parametro non

è supportato dallo switch, il relativo contattore sarà letto con l'equivalente di "-1". Quelle marcate con *Richiesta* sono sicuramente presenti in tutti gli apparati.

3.9 Istruzioni

Ogni voce contiene un set di istruzioni da eseguire quando viene effettuato il confronto con la voce della relativa tabella. Queste istruzioni posso riguardare eventuali modifiche da effettuare sul pacchetto o la pipeline a cui inoltrarlo. Il generico switch non deve supportare tutte le istruzioni, ma solamente quelle marcate come *Richiesta*. Il controller può interrogare lo switch e chiedere quali istruzioni *Opzionali* sono supportate dallo switch.

- *Opzionale:* **Meter *meter_id:*** invia direttamente il pacchetto ad una specifica voce della tabella di meter. Dopo essere stato conteggiato il pacchetto viene scartato.
- *Opzionale:* **Apply-Actions *action(s):*** esegue immediatamente l'azione specificata senza effettuare modifiche al gruppo di azioni da eseguire. Questa istruzione viene di solito utilizzata per modificare il pacchetto nel passaggio tra due tabelle o nel caso di multiple azioni in cui c'è una priorità, all'interno della stessa tabella.
- *Opzionale:* **Clear-Actions:** Cancella immediatamente il set di azioni da eseguire.
- *Richiesta:* **Write-Actions *action(s):*** unisce le azioni del set corrente; se l'azione esiste già, questa viene unita, altrimenti viene aggiunta.
- *Opzionale:* **Write-Metadata *metadata/mask:*** aggiorna il campo metadata con il parametro passato.
- *Richiesta:* **Goto-Table *next_table_id:*** indica la tabella successiva con cui proseguire la pipeline del pacchetto. Il numero della tabella deve essere maggiore di quello attuale. Le entry nell'ultima tabella non possono utilizzare questa istruzione. Gli switch che implementano una sola tabella possono non avere questa istruzione.

Il set di istruzioni associato ad ogni voce della tabella può contenere al massimo un'istruzione per tipologia. La priorità delle istruzioni rispecchia la lista

Counter	**Bits**	
Per ogni Flow Table		
Reference Count	32	*Richiesta*
Packet Lookups	64	
Packet Matches	64	
Per ogni flusso entry		
Received Packets	64	
Received Bytes	64	
Duration (seconds)	32	*Richiesta*
Duration (nanoseconds)	32	
Per ogni porta		
Received Packets	64	*Richiesta*
Transmited Packets	64	*Richiesta*
Received Bytes	64	
Transmitted Bytes	64	
Receive Drops	64	
Transmitted Drops	64	
Receive Errors	64	
Transmitted Errors	64	
Receive Frame ALignement Errors	64	
Receive Overrun Errors	64	
Receive CRC Errors	64	
Collisions	64	
Duration (seconds)	32	*Richiesta*
Duration (nanoseconds)	32	
Per Coda		
Transmit Packets	64	*Richiesta*
Transmit Bytes	64	
Transmit Overrun Errors	64	
Duration (seconds)	32	*Richiesta*
Duration (nanoseconds)	32	

Tabella 3.4: Lista dei contatoti - parte 1 -

Counter	**Bits**	
Per gruppo		
Reference Coutn (flusso Entry)	32	
Packet Count	64	
Byte Count	64	
Duration (seconds)	32	*Richiesta*
Duration (nanoseconds)	32	
Per "azioni" del gruppo		
Packet Count	64	
Byte Count	64	
Per entry della tabella Meters		
Flow Count	32	
Input Packet Count	64	
Input Byte Count	64	
Duration (seconds)	32	*Richiesta*
Duration (nanoseconds)	32	
Per Entry nella tabella Meter Bands		
In Band Packet Count	64	
In Band Byte Count	64	

Tabella 3.5: Lista dei contatoti - parte 2 -

appena esposta; ovvero prima vengono effettuate (se sono presenti) operazioni di *Meter*, poi di *Apply*, poi di *Write* ed infine quelle di *Goto*.
Uno switch **deve** rifiutare l'inserimento di una voce in una tabella se non è in grado di eseguire le operazioni richieste.

3.9.1 Set di Azioni

Ad ogni pacchetto è associata una lista di azioni (vuote di default). Una voce nelle tabelle può modificare questa azione tramite il "*write-actions*" o il "*clear-actions*". Il Set di azioni viene propagato nel passaggio di tabelle. Nel caso un'istruzione non contenga come ultima entry l'istruzione "*Goto-Table*", vengono applicate tutte le azioni in sospeso.

- **copy TTL inwards**: copia sul nuovo pacchetto il valore del TTL originale.
- **pop**: applica tutti i TAG al pacchetto.
- **push-MPLS**: applica il TAG per l'MPLS (RFC 3031 e RFC 3032) al pacchetto.
- **push-PBB** :applica il TAG per PBB (Provider Backbone Bridges 802.1ah) al pacchetto.
- **push-VLAN**: applica il TAG per una VLAN (802.11q) al pacchetto
- **copy TTL outwards**: copia sul nuovo pacchetto il valore del TTL "esterno" all'apparato.
- **decrement TTL**: decrementa il TTL del pacchetto.
- **set**: applica tutte le azioni al pacchetto.
- **qos**: applica tutte le azioni riferite al QoS al pacchetto.
- **group**: se specificato un gruppo, applica tutte le azioni relative al gruppo al pacchetto.
- **output**: se non è specificato nessun gruppo, inoltra il pacchetto alla porta specificata.

Un Action-Set può contenere al massimo un'azione per ogni tipo. Il parametro dell'azione può però essere multiplo (ovvero possono essere applicati più tag MPLS, o rimossi più tag). In questo caso però è necessario utilizzare

la "Apply-Action" subito dopo. Nel caso di azioni multiple su un pacchetto, l'ordine con cui vengono applicate le azioni è quello illustrato nell'elenco sopra. Se è specificata un'azione di output sia nel gruppo che nel set di azioni, quella del gruppo ha la precedenza. Nel caso invece non sia specificata alcuna azione di output, il pacchetto viene scartato.

3.9.2 Lista di Azioni

Nel caso di liste con più azioni, queste vengono eseguite sequenzialmente. Solamente quando si trova un'azione di tipo "Apply-Action" il pacchetto viene effettivamente modificato, in quanto fino a quel momento la pipeline lavora su una copia del pacchetto originale. Le azioni dell'action list sono cumulative sui pacchetti (ad esempio il tag con due VLAN differenti ha come risultato un pacchetto incapsulato in una VLAN, ed a sua volta incapsulato in un'altra VLAN).

3.9.3 Azioni

- *Richiesta:* **Output**: Il pacchetto viene inoltrato alla porta di uscita (si veda il paragrafo 3.2).
- *Opzionale:* **Set-Queue**: Imposta l'id per la coda da utilizzare (utilizzato nel caso di QoS).
- *Richiesta:* **Drop**: Azione di default se non viene specificato nulla. Il pacchetto viene scartato.
- *Richiesta:* **Group**: Inoltra il pacchetto al gruppo specificato.
- *Richiesta:* **Push-Tag/Pop-Tag**: Aggiungi o rimuovi un tag al pacchetto. I TAG supportati dipendono dallo switch ma a livello di openflow sono VLAN, MPLS e PBB.
- *Opzionale:* **Set-Field**: Le azioni associate a quest'azione permettono di modificare l'header del pacchetto. Per evitare problemi con la migrazione delle attuali reti, si raccomanda di utilizzare sempre il comando Set-Field anche per azioni sulle VLAN in quanto agisce sempre sul tag più esterno al pacchetto.
- *Opzionale:* **Change TTL**: Permette la modifica del TTL/hop-limit del pacchetto (IPv4 TTL, IPv6 Hop Limit, MPLS TTL). Si veda la tabella 3.6

Azione	**Parametro**	**Descrizione**
Set MPLS TTL	Nuovo valore del TTL (8 bits)	Sostituisce l'attuale valore del TTL del pacchetto MPLS con il nuovo valore. Viene applicata solamente se viene trovato un header MPLS
Decrement MPLS TTL		Decrementa di una unità l'attuale valore del TTL del pacchetto MPLS. Viene applicata solamente se viene trovato un header MPLS
Set IP TTL	Nuovov valore del TTL (8 bits)	Sostituisce il TTL nel caso di pacchetti IPv4 o l'hop-limit nel caso di pacchetti IPv6, con il nuovo valore ed aggiorna il checksum del pacchetto IP. Viene applicata solamente su pacchetti IPv4 o IPv6
Decrement TTL		Decrementa di una unità il TTL nel caso di pacchetti IPv4 o l'hop-limit nel caso di pacchetti IPv6. Viene applicata solamente su pacchetti IPv4 o IPv6
Copy TTL outwards		copia sul nuovo pacchetto il valore del TTL "esterno" all'apparato.
Copy TTL inwards		copia sul nuovo pacchetto il valore del TTL originale

Tabella 3.6: Pametri permessi per l'azione "Change TTL"

Capitolo 4

Il protocollo OpenFlow

In questo capitolo si espongono le caratteristiche del protocollo OpenFlow[1], del canale di comunicazione, delle tipologie di messaggi e dei segnali di controllo che vengono scambiati tra i diversi switch ed i controller.

[1]É stata scelta la versione 1.3 di OpenFlow, la quale è la più completa, mentre nelle prime esperienze pratiche verrà usata la versione 1.0 che è più semplice da gestire

4.1 Canale di Comunicazione

Il canale di comunicazione tra lo switch ed il controller (*openflow channel*) viene inizializzato dallo switch da una porta fisica non dedicata al funzionamento con openflow. Tale porta è adibita alla sola amministrazione e all'interfacciamento verso il controller. Il collegamento viene effettuato verso la porta TCP 6633 e normalmente utilizzando TLS[2] (Transport Layer Security).

Il protocollo di comunicazione tra lo switch ed il controller prevede tre tipologie generali di comunicazioni, inviate tramite il canale TLS.

- **Controller to switch**: messaggi iniziati dal controller usati per configurare o leggere lo stato dello switch.
- **Asynchronous**: messaggi inviati dallo switch al controller usati generalmente per notificare al controller delle modifiche della topologia di rete (ad esempio un nuovo host, lo spegnimento di una porta, etc.).
- **Symmetric**: messaggi inviati dallo switch o dal controller non scatenati da eventi esterni, ma come controlli interni.

4.1.1 Tipologie di Messaggi dal controller allo switch (*Controller-to-Switch Messages*)

- **Features**: il controller richiede allo switch di identificarsi e la lista delle funzionalità da lui supportate. Lo switch deve risponde con il suo identificativo (normalmente associato al MAC-Address della porta che effettua il collegamento con il controller) e le funzionalità da lui supportate. Normalmente questa comunicazione viene utilizzata durante la prima identificazione.
- **Configuration**: il controller invia allo switch delle aggiunte/modifiche/cancellazioni delle varie voci presenti nelle tabelle. Lo switch deve aggiornare le sue tabelle ed agire di conseguenza.
- **Read-State**: il controller legge lo stato dello switch (configurazione corrente, statistiche di utilizzo, contatori).
- **Packet-Out**: viene usato dal controller per inoltrare un pacchetto, normalmente ricevuto tramite il "Packet-In", in uscita su una specifica porta dello switch. Il messaggio deve contenere l'intero pacchetto da

[2]Si veda `http://it.wikipedia.org/wiki/Transport_Layer_Security`

inoltrare alla porta oppure un riferimento (buffer ID) nel caso il pacchetto sia memorizzato nello switch, la porta di destinazione ed una lista di azioni da applicare al pacchetto. Nel caso di una lista vuota il pacchetto viene scartato.

- **Barrier**: queste richieste di tipo "domanda/risposta" sono utilizzate dal controller come verifica che lo switch abbia ricevuto ed applicato correttamente delle istruzioni.

- **Role-Request**: messaggio di richiesta inoltrata dal controller per impostarsi il ruolo all'interno della rete (utilizzato in reti con più di un controller).

- **Asynchronous-Configuration**: messaggi usati dal controller per inserire/modificare dei filtri sui messaggi asincroni che vuole ricevere dagli switch (normalmente usati in reti con più di un controller durante la fase di negoziazione del canale openflow).

4.1.2 Tipologie di Messaggi asincroni (*Asyncronous Messages*)

- **Packet-In**: inoltra tramite la porta riservata CONTROLLER al controller, il pacchetto ricevuto. Normalmente viene eseguita quando un pacchetto non combacia con nessuna regola e quindi viene eseguita l'azione di defautl (table-miss flow entry) che può prevedere l'inivio al controller del pacchetto per l'analisi.

- **Flow-Removed**: informa il controller che è stata rimossa una voce da una tabella (ad esempio nel caso della scadenza di un timeout).

- **Port-Status**: informa il controller sul cambiamento di stato di una porta. Viene inviato anche il motivo del cambiamento (ad esempio spegnimento via software o mancanza di link).

4.1.3 Tipologie di messaggi simmetrici (*Symmetric Messages*)

- **Hello**: messaggi scambiati dal controller e dallo switch durante l'inizializzazione della connessione.

- **Echo**: il pacchetto di Echo deve trovare un suo corrispettivo nel pacchetto Echo-Reply. Vengono usati normalmente per verificare lo stato del canale tra lo switch ed il controller.
- **Error**: messaggio usato dallo switch o dal controller per notificare dei problemi.
- **Experimenter**: messaggio utilizzato per fornire funzionalità avanzate a livello utente per openflow.

Capitolo 5

Il software per le simulazioni: Mininet

Mininet[5] è un software che permette di creare reti virtuali formate da client (host), switch e controller OpenFlow.
Il software mette a disposizione dell'utente un'interfaccia a linea di comando (Command Line Interface o CLI[1]) e delle interfacce di programmazione (Application Program Interface o API[2]).

[1]Per dettagli `http://it.wikipedia.org/wiki/Interfaccia_a_riga_di_comando`

[2]Per dettagli `http://it.wikipedia.org/wiki/Application_programming_interface`

5.1 Installazione ed attivazione della macchina virtuale

Per installare l'ambiente di prova è sufficiente eseguire (dall'utente root) lo script "installa.sh"; questo script installerà tutto l'ambiente necessario.
Si troveranno le cartelle:

- **Documenti/Aggiornamento_Routerboard**: contenente i files per effettuare l'upgrade del sistema operativo delle routerboard, i files di configurazione di default ed il pacchetto per abilitare openflow.
- **Documenti/avior**: il software "avior".
- **Documenti/Macchina_Virtuale**: una macchina virtuale, usabile su Virtualbox[3], con supporto per openflow versione 1.3.
- **Documenti/Template_miniedit**: software miniedit e relativi template
- **floodlight**: il software relativo al controller.

5.2 Comandi di base

Una volta avviato l'ambiente con il comando "sudo mn" se come utente o semplicemente con "mn" se si hanno già i privilegi di root, ci si troverà davanti all'interfaccia a linea di comando del simulatore. Di default, se non vengono passati parametri, viene simulata una rete come quella di Figura 5.1. Se non viene specificato diversamente, gli host hanno un identificativo che inizia con la lettera "H" minuscola, gli switch con un una "S" minuscola ed i controllerc con una "C" minuscola, seguiti da un numero progressivo.

```
$ sudo mn
[sudo] password for user:

** Creating network
** Adding controller
** Adding hosts:
h1 h2
** Adding switches:
```

[3]VirtualBox è un software di virtualizzazione multipiattaforma, reperibile da `https://www.virtualbox.org/`

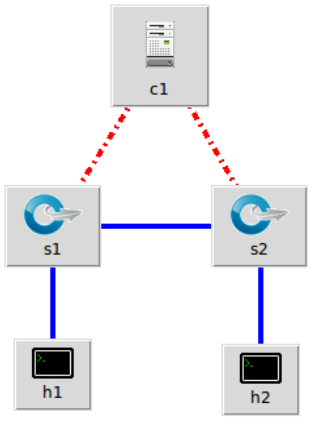

Figura 5.1: Schema esperienza Base

```
s1
** Adding links:
(h1, s1) (h2, s1)
** Configuring hosts
h1 h2
** Starting controller
** Starting 1 switches
s1
** Starting CLI:
```

Per uscire dall'ambiente basta dare il comando "exit" che causerà lo spegnimento dei 2 host virtuali, dello switch e del controller. Come ultimo dato restituisce il tempo totale di uptime dell'ambiente mininet.

```
mininet> exit
** Stopping 1 switches
s1 ..
** Stopping 2 hosts
h1 h2
** Stopping 1 controllers
c0
** Done
completed in 147.951 seconds
```

Alcuni comandi per vedere la struttura della rete creata da mininet sono:

- **nodes**: elenca tutti i nodi che fanno parte della simulazione (switch, hosts, controller etc).

  ```
  mininet> nodes
  available nodes are:
  c0 h1 h2 s1
  ```

  Si nota la presenza di un controller (c0), due host generici (h1 e h2) ed uno switch (s1).

- **dump**: elenca l'attuale struttura di rete; gli hosts, gli IP, i collegamenti ed il PID (Process ID) relativo al thread all'interno dell'host fisico.

  ```
  mininet> dump
  <Host h1: h1-eth0:10.0.0.1 pid=9701>
  <Host h2: h2-eth0:10.0.0.2 pid=9702>
  <OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=9707>
  <OVSController c0: 127.0.0.1:6633 pid=9693>
  ```

- **net**: elenca gli attuali collegamenti dei dispositivi.

  ```
  mininet> net
  h1 h1-eth0:s1-eth1
  h2 h2-eth0:s1-eth2
  s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
  c0
  ```

5.3 Come interagire con gli host

Il modo più semplice per interagire con gli host creati è quello di impartire i comandi alla CLI nella sintassi:

```
HOST COMANDO PARAMETRI
```

Dove HOST è l'host virtuale a cui far eseguire il comando, COMANDO è il comando da far eseguire e PARAMETRI sono gli eventuali parametri da passare. "COMANDO" può essere un qualunque script/eseguibile linux installato sull'host fisico (ad esempio ping, iperf, wget, phyton etc..) I comandi più usati sono:

- **hX ping hY**: esegue una serie di ping dall'host X all'host Y .

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.28 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.048 ms
^C
— 10.0.0.2 ping statistics —
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.048/0.668/1.288/0.620 ms
```

- **A wget B**: fa effettuare il download all'host A dell'URL specificato in B .

- **pingall**: esegue un ping da ogni host verso tutti gli altri, serve per verificare che l'ambiente funzioni e per obbligare gli host ad effettuare una prima richiesta ARP.

```
mininet> pingall
** Ping: testing ping reachability
h1 -> h2
h2 -> h1
** Results: 0% dropped (2/2 received)
```

- **iperf A B**: esegue un test di banda tra due hosts (A e B in questo caso).

```
mininet> iperf
** Iperf: testing TCP bandwidth between h1 and h2
waiting for iperf to start up...***
Results: ['33.5 Gbits/sec', '33.5 Gbits/sec']
mininet>
```

5.4 Struttura della rete all'interno di mininet

In mininet ogni host virtuale è un processo che è in esecuzione sulla macchina fisica. Ogni scheda di rete delle macchine virtuali è associata ad un'interfaccia virtuale della macchina fisica. I nomi delle interfacce sono riferite al dispositivo ed al nome che quell'interfaccia ha all'interno della macchina virtuale.

Ad esempio "h44-eth6" è l'interfaccia 7 (da eth0 a eth6) dell'host numero 44. Nella macchina virtuale si notano le varie schede associate ai diversi apparati, in questo caso due interfacce dello switch "s1".

```
s1-eth1 Link encap:Ethernet HWaddr 0e:f8:3f:7d:94:b3
        inet6 addr: fe80::cf8:3fff:fe7d:94b3/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:414091 errors:0 dropped:0 overruns:0 frame:0
        TX packets:409811 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:18124834510 (16.8 GiB) TX bytes:27050418 (25.7 MiB)

s1-eth2 Link encap:Ethernet HWaddr 86:4d:dd:ce:a2:bc
        inet6 addr: fe80::844d:ddff:fece:a2bc/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:409794 errors:0 dropped:0 overruns:0 frame:0
        TX packets:414108 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:27046608 (25.7 MiB) TX bytes:18124838320 (16.8 GiB)
```

Essendo macchine virtuali che sfruttano uno switch virtuale (OpenVSwitch[4][6]), nonostante le interfacce siano nominalmente collegate a 10Gb, è facile raggiungere con dei test velocità di oltre 30Gbits/sec. Questa cosa ovviamente, nella realtà non è possibile. É però prevista la possibilità di simulare un limite fisico di banda disponibile su ogni link ed anche un eventuale latenza relativa ai pacchetti che transitano su un determinato collegamento, utilizzando il software per la gestione della banda tc[5]

5.5 OpenFlow su mininet

L'ambiente di mininet dà supporto nativo a switch openflow. Essendo una paravirtualizzazione, sfruttando un'applicazione sulla macchina host cone NOX o POX, è possibile ricreare un ambiente completo openflow. Risulta pertanto relativamente facile simulare reti anche complicate (decine di switch e diversi

[4]Open vSwitch è un modulo per il kernel linux che permette di creare uno switch virtuale su un host linux.

[5]Per informazioni su come implementarlo si veda `http://man7.org/linux/man-pages/man8/tc.8.html`.

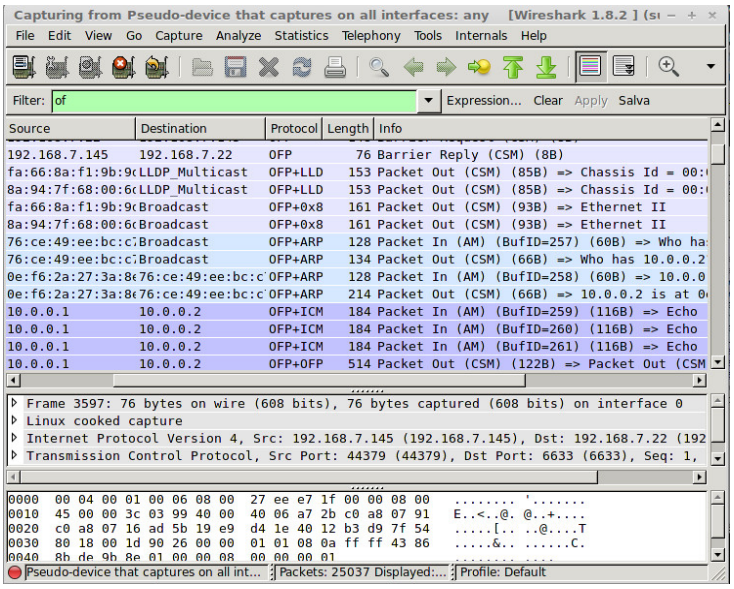

Figura 5.2: Plugin per Wireshark in funzione

hosts) basate su openflow. Essendo openflow un protocollo che va ad interagire principalmente con switch e router, la complessità della rete ha a che fare con il numero di switch e/o router che un pacchetto deve passare prima di arrivare a destinazione, e non direttamente in base al numero di host della rete.

5.6 Wireshark OpenFlow dissector

E' stato sviluppato un plugin per Wireshark per analizzare specificatamente il protocollo openflow. Il plugin si attiva con il filtro "of" ed esegue un filtraggio sui pacchetti, mostrando solamente i pacchetti relativi ad openflow, come si vede in figura 5.2

Capitolo 6

Floodligh

Floodlight[7] è un software per le SDN. Floodlight è scritto in java ed esegue un controller OpenFlow. Si è scelto questo controller, a discapito di altri più conosciuti come NOX[1] o POX[2] in quanto questo controller si presta molto bene a debug, fornisce una comoda interfaccia web per verificare i dati inseriti e delle semplici API con cui interfacciarsi tramite normali richieste http di tipo GET o POST.

[1] http://www.noxrepo.org/nox/about-nox/
[2] http://www.noxrepo.org/pox/about-pox/

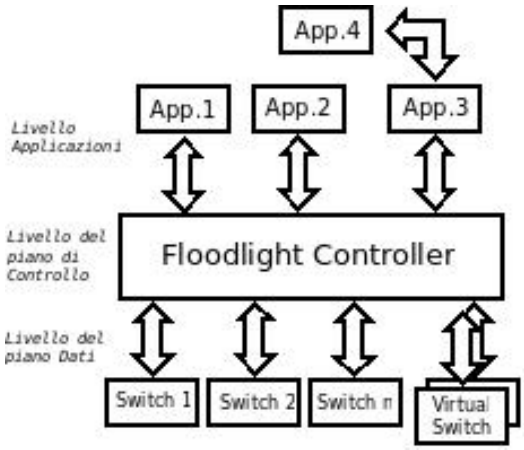

Figura 6.1: Struttura Funzionamento Floodlight

6.1 Floodlight come controller standalone

Tra i vari controller disponibili il più semplice e flessibile per effettuare dei test è sicuramente Floodlight[7]. Alcune delle sue caratteristiche sono:

- É un controller specifico per openflow e di conseguenza si interfaccia con tutti gli apparati che supportano il protocollo openflow (siano essi dispositivi fisici o virtuali).
- É rilasciato sotto licenza Apache, quindi può essere usato in qualsiasi ambito e per ogni scopo.
- É sviluppato da una comunità aperta.
- É relativamente facile da installare e da eseguire, in quanto il pacchetto è quasi completamente indipendente da librerie esterne.
- Testato e relativamente stabile.
- Api pubbliche e di facile interfacciamento.

Floodlight, essendo un software per le SDN[1, 2], deve riuscire ad interfacciarsi con lo strato superiore dato dalle applicazioni e con uno strato inferiore dato dai dati. Openflow supporta due modalità di inserimento delle voci nelle tabelle degli switch: modalità reattiva e proattiva.

- **Modalità Reattiva**: si ha quando un pacchetto raggiunge uno switch openflow e non combacia con nessuna voce delle tabelle. Il pacchetto viene inoltrato al controller che lo analizza, crea le voci da inserire nelle

tabelle degli switch affinché il pacchetto arrivi a destinazione, inserisce queste voci nei relativi switch e comunica allo switch di continuare con l'inoltro del pacchetto.

- **Modalità Proattiva**: si ha quanto le voci vengono inserite indipendentemente dai pacchetti in transito, da un'applicazione esterna.

Floodlight di default funziona in modalità Reattiva.
Nella cartella "floodlight/target/" esistono due files jar distinti:

- floodlight_reattiva.jar che è stato compilato con il supporto per la modalità reattiva.
- floodlight_proattiva.jar che è stato compilato con il supporto per la modalità proattiva.

6.1.1 Livello Applicazione

Il livello applicazione (*Application Tier*) è lo strato più alto del controller che deve permettere alle applicazioni di interfacciarsi con lui (si ricorda che normalmente lo scopo del controller è solo quello di passare informazioni agli switch, non di prendere decisioni sul creare/rimuovere voci). Le applicazioni si interfacciano con il controller tramite API, che permettono alle applicazioni di avere informazioni sugli switch connessi, sugli host e passare agli switch le voci da inserire nelle tabelle. Essendo la comunicazione fatta tramite delle API esterne al software centrale, nessuno vieta ad esempio di utilizzare diverse applicazioni, ognuna delle quali si occupa di una specifica tipologia di pacchetti.

6.1.2 Livello del piano di Controllo

Il livello del piano di controllo (Control Plane Tier) è fisicamente il software del controller: gestisce le connessioni con gli switch e fa da tramite tra switch ed applicazioni.

6.1.3 Livello del piano Dati

Il livello del piano dati (Data Plane Tier) è l'ultimo strato del controller, che si interfaccia direttamente (tramite openflow) alle tabelle interne allo switch. Come descritto nei capitoli precedenti, lo scopo del controller è quello di fare da accentratore per gli switch. In fase di avvio lo switch si collega al controller, il quale fà da intermediario tra l'applicazione che decide come

verranno trattati i flussi dei pacchetti e gli apparati che fisicamente dovranno instradate il pacchetto nella rete.

6.2 Interfacciare mininet con floodlight

Mininet è un applicazione che già supporta nativamente openflow, quindi è sufficiente specificare via riga di comando l'istruzione per indicare agli switch l'ip del controller remoto. Ad esempio, ipotizzando l'ip del controller 192.168.7.22 (si dia per scontato che la porta sia la porta di default 3366 senza TLS)

```
# mn - -controller=remote,192.168.7.22
```

6.3 Interfacciarsi con floodlight

Floodlight, una volta avviato si posiziona in ascolto su due porte:

- TCP:8080 Porta utilizzata per la parte web.
- TCP:6633 Porta utilizzata per le connessioni in ingresso dagli switch.

6.3.1 Interfaccia Web

Per avere una visualizzazione grafica dei dati raccolti dal controller ci si può collegare via web all'indirizzo *http://<controller_ip>:8080/ui/index.html*

Da qui si accede direttamente a:

- **Dashboard**: è il riassunto della situazione del controller.
- **Topology**: presenta un grafico dei collegamenti che il controller rileva tra i vari switch.
- **Switch**: è l'elenco degli switch presenti nella rete, dello stato delle relative porte e dei flussi associati.
- **Hosts**: è l'elenco di tutti gli host rilevati dal controller (e quindi da almeno uno switch) nella rete.

Da questa interfaccia web è solamente possibile vedere lo stato della rete e degli switch, per poter passare informazioni al controller è necessario utilizzare delle applicazioni esterne o le sue API.

6.3.2 Floodlight API

Per interfacciarsi con floodlight bisogna utilizzare le sue API. Il sistema più semplice, essendo chiamate HTTP in POST e GET, è tramite il software curl. Curl è un software che permette di impostare in modo intuitivo dei parametri e poi inviarli ad un server web esterno. La sintassi è relativamente semplice e si compone genericamente da:

```
curl -d '
{
        "parametro" : " valore ",
        "parametro_2" : " valore_2 ",
        "parametro_3" : " valore_3 ",
        ...
}
' URL
```

Per una lista completa dei comandi si veda la pagina dedicata sul sito di floodlight[3]. I parametri principali sono riassunti in tabella 6.1 e 6.2

Gli URL basilari sono:

- **/wm/staticflussoentrypusher/json**:
  Aggiungi/Rimuovi Flow
  HTTP POST per aggiungere flusso, HTTP DELETE per cancellarli

- **/wm/staticflussoentrypusher/list/<switch>/json**:
  Visualizza i flusso associati ad uno switch
  switch: Identificativo valido dello switch DPID

Se ad esempio si vuole indirizzare tutto il traffico che entra sulla porta 2, alla porta 1

```
curl -d
'{
        "switch": "<MAC_ADDRESS_SWITCH>",
        "name":"regola_1",
```

[3] `http://www.openflowhub.org/display/floodlightcontroller/Floodlight+REST+API`

Chiave	**Valore**	**Note**
switch	<switch ID>	Identificativo dello switch, nel formato xx:xx:xx:xx:xx:xx:xx:xx
name	<testo>	Nome del flusso, deve essere univoco
actions	<par> = <val>	Parametro, nel caso non sia settato niente e considerato a DROP
active	<booleano>	Se la voce è attiva o no
wildcards		Combacia sempre, voce di default
ingress-port	<numero>	Numero della porta dello switch, può essere in formato decimale oppure se preceduto da 0x in formato esadecimale
src-mac	<mac address>	Mac Address sorgente nella forma: xx:xx:xx:xx:xx:xx
dst-mac	<mac address>	Mac Address destinatario nella forma: xx:xx:xx:xx:xx:xx
vlan-id	<numero>	Identificativo della VLAN, può essere in formato decimale oppure se preceduto da 0x in formato esadeciamle.
src-ip	<Indirizzo IP>	Indirizzo IP sorgente, nella forma. xx.xx.xx.xx
dst-ip	<Indirizzo IP>	Indirizzo IP destinatario, nella forma. xx.xx.xx.xx
src-port	<numero>	Porta sorgente, può essere in formato decimale oppure se preceduto da 0x in formato esadecimale.
dst-port	<numero>	Porta destinazione, può essere in formato decimale oppure se preceduto da 0x in formato esadecimale.

Tabella 6.1: Lista delle proprietà basilari di FloodLight

Parametro	**Valore**	**Note**
output	<numero>, all, controller, local, ingress-port, normal, flood	In quale porta, gruppo di porte o porte riservate instradare il pacchetto. É meglio non usare l'opzione DROP, piuttosto non specificare alcuna azione per scartare il pacchetto
strip-vlan		Rimuove il tag VLAN
set-vlan-id	<numero VLAN>	Tagga i pacchetti con la VLAN specificata. Esadecimale se preceduto da 0x, altrimenti decimale
set-src-mac	<mac address>	Imposta il mac address sorgente al valore passato nella forma xx:xx:xx:xx:xx:xx
set-dst-mac	<mac address>	Imposta il mac address destinatario al valore passato nella forma xx:xx:xx:xx:xx:xx
set-src-ip	<ip address>	Imposta l'IP sorgente al valore passato nella forma xxx.xxx.xxx.xxx
set-dst-ip	<ip address>	Imposta l'IP destinatario al valore passato nella forma xxx.xxx.xxx.xxx
set-src-port	<number>	Imposta la porta sorgente al valore passato. Esadecimale se preceduto da 0x, altrimenti decimale
set-dst-port	<number>	Imposta la porta destinataria al valore passato. Esadecimale se preceduto da 0x, altrimenti decimale

Tabella 6.2: Lista delle azioni fondamentali di FloodLight

```
        “priority”:“2”,
        “active”:“true”,
        “ingress-port”:“2”,
        “actions”:“output=1”
}’
http://<IP_CONTROLLER>:8080/wm/staticflowentrypusher/json
```

La stessa cosa si applica agli altri parametri.

Capitolo 7

Simulazione di una rete con mininet e floodlight

7.1 Simulazione di due host virtuali con controller esterno

Per un primo esempio si utilizza una struttura come la Figura 10.5 :
Il primo esempio usa la versione compilata con il flag per il funzionamento in modalità reattiva, in modo da poter analizzare il software grafico.
Eseguiamo prima l'istanza di floodlight:

```
~/floodlight# java -jar target/floodlight_reattiva.jar
...
Listening for switch connections on 0.0.0.0/0.0.0.0:6633 ...
```

Poi avviamo mininet in modalità due host con controller remoto:

```
$ sudo mn -x - -controller=remote,<IP_CONTROLLER>,port=6633
- -mac
```

L'opzione - -mac imposta automaticamente i mac address degli switch in modo sequenziale, mentre l'opzione -x avvia i terminali, uno per ogni dispositivo coinvolto.
Nei log del controller comparirà la connessione tra lo switch ed il controller, lanciando wireshark ed impostando il filtro su "of" è possibile vedere la connessione (in charo) tra lo switch ed il controller.

7.2 Ping tra i due apparati

Se nel terminale "h1" si lancia un ping verso l'host h2 (il generico host hA ha come ip di default 10.0.0.A/16), si vede la risposta di h2 ai ping. Nella pagina relativa ai flusso associati allo switch si troveranno dei flow simili a quelli riportati nella tabella 7.1.

Nel dump di wireshark si vedono i pacchetti di broadcast per le richieste ARP (OFP+ARP) che, non trovando un match nelle tabelle di flusso (le tabelle di flusso di default sono vuote), saranno inoltrati al controller il quale risponderà di inoltrarti a tutte le porte. Subito dopo lo stesso percorso tocca al pacchetto ICMP (OFP+ICM); verrà inoltrato al controller il quale creerà il flusso corrispondente, aggiornerà la tabella dello switch e dirà allo switch di inoltrare il pacchetto secondo la tabella. Il primo pacchetto icmp infatti avrà un tempo di risposta maggiore, in quanto deve attendere che tutto il processo si concluda prima di arrivare a destinazione. Il comportamento si

Match	Azione
port=1, VLAN=-1, prio=0, src=2a:04:de:6e:c9:5b, dest=7a:fb:77:a6:9a:71	output 2
port=2, VLAN=-1, prio=0, src=7a:fb:77:a6:9a:71, dest=2a:04:de:6e:c9:5b	output 1

Tabella 7.1: Esempio di due flow per un ping tra due host connessi sullo stesso switch

ripete dopo che scade il timeout del flusso.
Tramite l'interfaccia web, o tramite le API è possibile analizzare i due flow (uno per i pacchetti "echo" ed uno per i pacchetti "echo-replay") che il controller genera per permettere il passaggio dei pacchetti. Tramite API invece è sufficiente usare il comando:

curl http://<CONTROLLER_IP>:8080/wm/core/switch/1/flow/json

Il cui output opportunatamente formattato è:


```
{"1":
  [1
      {
          "tableId":0,
          "match":
          {
              "dataLayerDestination":"4e:eb:24:9a:f3:8b",
              "dataLayerSource":"da:36:0a:59:d6:42",
              "dataLayerType":0x0000,
              "dataLayerVirtualLan":-1,
              "dataLayerVirtualLanPriorityCodePoint":0,
              "inputPort":1,
              "networkDestination":"0.0.0.0",
              "networkDestinationMaskLen:0,
              "networkProtocol":0,
              "networkSource":"0.0.0.0",
              "networkSourceMaskLen":0,
              "networkTypeOfService":0,
              "transportDestination":0,
```

```
                ”transportSource”:0,
                ”wildcards”:2629872
        },
        ”durationSeconds”:285,
        ”durationNanoseconds”:911000000,
        ”priority”:0,
        ”idleTimeout”:5,
        ”hardTimeout”:0,
        ”cookie”:9007199254740992,
        ”packetCount”:291,
        ”byteCount”:28182,
        ”actions”:
        [{
                type:OUTPUT,
                length:8,
                port:2,
                maxLength:-1,
                lengthU:8
        } ]
},
{
        tableId:0,
        match:
        {
                dataLayerDestination :da:36:0a:59:d6:42,
                dataLayerSource :4e:eb:24:9a:f3:8b,
                dataLayerType :0x0000,
                dataLayerVirtualLan :-1,
                dataLayerVirtualLanPriorityCodePoint:0,
                inputPort :2,
                networkDestination :0.0.0.0,
                networkDestinationMaskLen:0,
                networkProtocol :0,
                networkSource :0.0.0.0,
                networkSourceMaskLen :0,
                networkTypeOfService :0,
                transportDestination :0,
                transportSource :0,
                wildcards :2629872
        },
```

```
                durationSeconds :285,
                durationNanoseconds :905000000,
                priority :0,
                idleTimeout :5,
                hardTimeout :0,
                cookie :9007199254740992,
                packetCount :291,
                byteCount :28182,
                actions:
                [{
                        type :OUTPUT,
                        length :8,
                        port :1,
                        maxLength :-1,
                        lengthU :8
                } ]
        }
    ]
```

Nella risposta si riconoscono tutti i parametri relativi ai due flow, le azioni, la durata della voce nella tabella e le porte di ingresso ed uscita.
Se adesso si interrompe il processo del controller (CTRL+C), si esegue il controller in modalità proattiva

```
~/floodlight# java -jar target/floodlight_proattiva.jar
```

e si ripete lo stesso test, non ci saranno risposte ai ping. Questo finché non si creerà una voce della tabella e la si inserirà nel controller. Per fare questo si può utilizzare avior[8], un'interfaccia JAVA per floodlight, eseguibile con

```
java - jar avior_x86.jar
```

oppure inserire le voci direttamente in floodlight tramite le API:

```
curl -d '{
        "switch": "00:00:00:00:00:00:00:01",
        "name":"regola_1",
        "priority":"1",
        "active":"true",
```

```
        "ingress-port":"2",
        "actions":"output=1"
}' http://<CONTROLLER_IP>:8080/wm/staticflussoentrypusher/json
```

E per la seconda voce, ovvero per il traffico di ritorno:

```
curl -d '{
        "switch": "00:00:00:00:00:00:00:01",
        "name":"regola_2",
        "priority":"2",
        "active":"true",
        "ingress-port":"1",
        "actions":"output=2"
}' http://<CONTROLLER_IP>:8080/wm/staticflussoentrypusher/json
```

Verificando l'esatto inserimento con

```
curl http://<CONTROLLER_IP>:8080/wm/core/switch/1/flusso/json
```

É presente un'utility all'interno della directory ~/floodlight/apps/circuitpusher/ che permette, in una rete con più switch, di creare dei percorsi tra 2 indirizzi IP.

La sintassi è:

```
./circuitpusher.py –controller=<CONTROLLER_IP>:8080
–type ip –src <IP_SORGENTE> –dst <IP_DESTINAZIONE>
–add –name <NOME_FLOW>
```

(attenzione, dato che gli apparati sono comunque switch, questi devono conoscere il mac address associato all'ip su cui si sta lavorando, quindi gli host SORGENTE e DESTINAZIONE devono comunque avere tentato di effettuare traffico, in modo che gli switch conoscano il loro mac address e la porta su cui sono collegati).
Lo script interroga il controller sugli switch presenti nella rete, sugli host e sui loro collegamenti e costruisce il percorso dall'host A all'host B. Inserisce poi nel controller i vari flussi relativi al percorso dei pacchetti.
Per testare lo script si può utilizzare il template "tree" già presente in mininet con la sintassi:

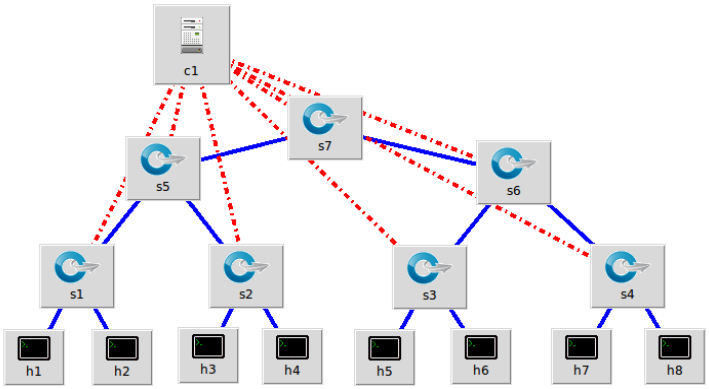

Figura 7.1: Schema struttura ad albero

sudo mn -x - -controller=remote,<CONTROLLER_IP>,port=6633 - -topo tree,3 - -mac

che genera una rete secondo lo schema di figura 7.1, e verificare un ping tra l'host 1 e l'host 8, attraversando quindi 5 switch diversi.

7.3 MiniEdit

In alternativa è possibile utilizzare miniedit[9], un software scritto in python che permette di configurare graficamente la rete mininet e di simularla. É possibile importare in miniedit le configurazioni di default di mininet e quindi di modificare o visualizzare la topologia di rete che si andrà ad eseguire.

Capitolo 8

Mikrotik RB750 ed OpenFlow

Mikrotik[1] è un'azienda che realizza apparati di rete. Attualmente produce router con sistema operativo proprietario Mikrotik "RouterOS" e diversi apparati Wireless a 2.4GHz, a 5GHz e su banda licenziata. Questi apparati sono abbastanza performanti per la loro fascia di prezzo e tramite un upgrade ufficiale del sistema operativo e l'installazione di un pacchetto opzionale (scaricabile dal sito ufficiale), è possibile abilitare determinati apparati a lavorare con OpenFlow.

[1] http://www.mikrotik.com/

8.1 Mikrotik 750

L'apparato che si utilizzerà per l'esperienza è la **Mikrotik RB750**. Si tratta di un router con le seguenti caratteristiche:

Codice Prodotto	RB750
Frequenza CPU	400 MHz
Modello CPU	AR7241-AH1A
CPU core	1
Architettura	MIPS-BE
RAM	32 MB
Porte 10/100Mbit	5
Tensione di alimentazione	Da 10V a 28V
Tensione di alimentazione su PoE	Da 10V a 28V
Massimo consumo	2.5W
Licenza di RouterOS	Level4 [1]

Nelle immagini sono riportate Routerboard RB750GL, la differenza è solo nelle porte (1Gbit invece che a 100Mbit). La routerboard si presenta come in Figura 8.1
I 5 led sopra la scheda indicano lo stato (up/down) delle singole porte. Questi led sono comandati via software e non via hardware. É quindi possibile che si verifichino situazioni (ad esempio lo spegnimento software della scheda) in cui i led sono accesi nonostante non ci sia nessun cavo collegato.
Frontalmente, come si vede in figura 8.2, oltre al connettore di alimentazione, si vede il pulsante di reset [2] ed il led di accensione della scheda. La prima porta permette alla scheda di essere alimentata via PoE[10].
Dopo un reset, oppure come configurazione di fabbrica, la scheda si presenta con IP : 192.168.88.1 sulle porte 2-5, utilizzando come nome utente **admin** e con password vuota.
In caso di problemi, sotto la scheda (come si vede in figura 8.3) sono stampati i mac address (iniziale e finale) di ognuna delle 5 porte della scheda.

[1] Si veda http://wiki.mikrotik.com/wiki/Manual:License#License_Levels per maggiori dettagli sulle limitazioni

[2] Per resettare la routerboard, premere il pulsante di reset durante la fase di boot fino a ché il LED non inizia a lampeggiare, rilasciare il pulsante ed attendere il riavvio completo della routerboard

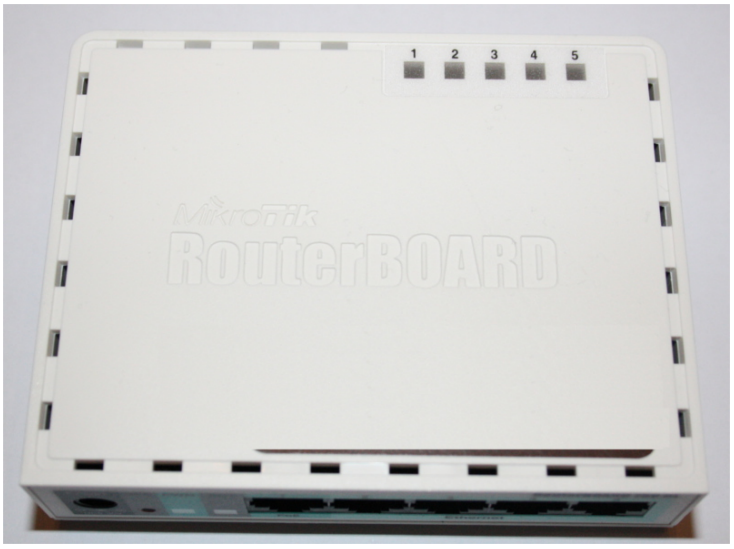

Figura 8.1: Routerboard RB750GL

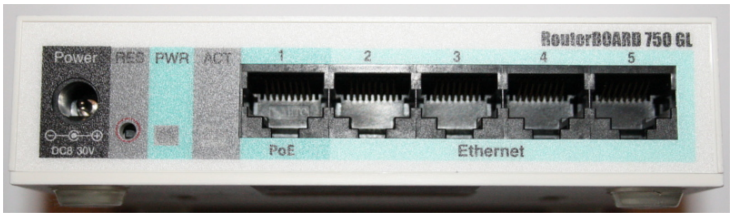

Figura 8.2: Vista Frontale Routerboard

Figura 8.3: Indirizzi MAC address stampati sotto la scheda

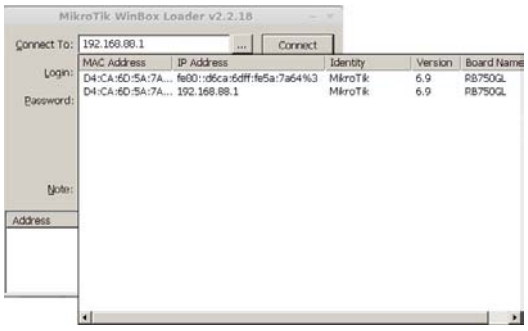

Figura 8.4: In base al parametro (Mac Address, IPv4 o IPv6) sopra cui si clicca, si sfrutta quel livello di collegamento.

8.2 Utilizzare la Routerboard

Il metodo più semplice per collegarsi alla scheda è quello di utilizzare l'utility fornita dalla casa madre: WinBox[3].
Una volta eseguita, scansiona la rete per trovare la scheda. Ci si può collegare in tre diversi modi alla scheda:

- **Via IPv4**: Conoscendo l'ip della scheda, o effettuando una ricerca, il software permette di collegarsi all'indirizzo ip della scheda.
- **Via Mac Address**: Nel caso la scheda non abbia IP configurati, o non sono accessibili ci si può collegare tramite mac address.
- **Via IPv6**: Sfruttando il neighbor discovery di IPv6, il software è in grado di trovare l'ip link local della scheda ed utilizzarlo per collegarsi via IP.

Ovviamente, nel caso di schede configurate con IPv4 e IPv6, sulla schermata compariranno due voci, uno per l'IPv4 ed uno per l'IPv6, come si vede in figura 8.4. É anche possibile collegarsi via TCP/IP tramite telnet o ssh oppure avviare un terminale dall'interno di winbox.

I menù comandi sono organizzati ad albero, quindi se per eseguire un comando si utilizza, ad esempio, */ip route print*

> [admin@MikroTik] > /ip route print
> Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,

[3]Scaricabile da `http://download2.mikrotik.com/winbox.exe`, l'utility funziona egregiamente anche su linux con wine

B - blackhole, U - unreachable, P - prohibit # DST-ADDRESS
PREF-SRC GATEWAY DISTANCE
0 ADC 192.168.88.0/24 192.168.88.1 ether2-master-l... 0

si può anche utilizzare, con gli stessi risultati:

[admin@MikroTik] > /ip route
[admin@MikroTik] /ip route> print

I comandi base base per utilizzare la scheda sono

- **?**: mostra i comandi e una breve descrizione
- **[tab]**: il pulsante tabulazione, come su linux, completa i comandi.
- **colore azzurro**: menù di sistema.
- **colore viola**: comandi di sistema.
- **..**: salgo di un livello nell'alberatura.
- /: torno al livello iniziale.

Per la lista completa si rimanda alla guida ufficiale Mikrotik per la console[4], oppure a quella per winbox[5].

8.3 Upgrade del Sistema Operativo

La prima cosa da fare per poter utilizzare la scheda con openflow è la modifica dell'ip e del default gateway, l'aggiornamento ad un sistema operativo che supporti openflow ed infine l'installazione del relativo pacchetto openflow. Come primo passo si deve configurare l'ip della scheda. Per fare questo, una volta collegati ad una console della scheda, prima si aggiunge un secondo ip, ci si collega al secondo ip e si rimuove quello di default.

[admin@MikroTik] > ip address add address=<INDIRIZZO_IP>
interface=ether2-master-local netmask=<SUBNET_MASK>
[admin@MikroTik] > quit

[4]Si veda `http://wiki.mikrotik.com/wiki/Manual:Console`
[5]Si veda `http://wiki.mikrotik.com/wiki/Manual:Winbox`

$telnet <INDIRIZZO_IP>
Login:admin
password:

[admin@MikroTik] > ip address print

Si controlli il riferimento all'indirizzo IP da rimuovere

[admin@MikroTik] > ip address remove numbers=0
[admin@MikroTik] > ip route add gateway=<IP_GATEWAY>

Si verifichi la configurazione con un ping ad un indirizzo IP pubblico

[admin@MikroTik] > ping <INDIRIZZO_PUBBLICO> count=4

Come prima cosa si effettui l'aggiornamento del bootloader con:

[admin@MikroTik] > system routerboard upgrade

Verrà richiesto un riavvio; lo si effettui con

[admin@MikroTik] > system reboot

Per completare l'upgrade è necessario collegarsi con winbox all'apparato e trascinare il file "routeros-mipsbe-6.11.npk" sulla finestra; a caricamento completato la scheda si riavvia ed effettua l'aggiornamento.
La documentazione ufficiale per procedere all'aggiornamento è reperibile su `http://wiki.mikrotik.com/wiki/Manual:Upgrading_RouterOS`

8.4 Installazione ed abilitazione del pacchetto OpenFlow

Per installare il pacchetto openflow si trascini il file (come fatto per aggiornare il sistema operativo) "openflow-6.11-mipsbe.npk" e "Base_config_172.backup" sulla finestra di winbox e, ad upload finito, si riavii la routerboard.

8.5 Ambiente di prova: OpenFlow, floodlight e mikrotik

Per poter collegare la routerboard RB750 a floodlight occorre configurare un IP sulla scheda, abilitare l'istanza di openflow, specificare l'ip del controller, ed infine abilitare le porte ad openflow.
Come prima cosa conviene resettare la scheda al fine di rimuovere le code di NAT [6], il firewall ed il server DHCP[7] che sono previsti nella configurazione iniziale della scheda.
Per fare ciò si colleghi la routerboard sulla porta 5, ci si colleghi in telnet e si carichi la configurazione presente nel file "Base_config_172.backup" con il comando

> [admin@MikroTik] > /system backup load
> name=Base_Config_172.backup
> Restore and reboot? [y/N]: y

Dopo il riavvio la Routerboard sarà raggiungibile tramite l'ip 172.16.194.1/24 sulla porta 5.
A questo punto si imposti l'ip corretto sull'interfaccia 5 e si rimuova il vecchio ip; si utilizzi la stessa procedure illustrata nel capitolo 8.3

> [admin@MikroTik] > /ip address
> [admin@MikroTik] > add address=<NUOVO_IP>/<SUBNET>
> interface=ether5
> [admin@MikroTik] > print
> [admin@MikroTik] > remove <ID_ASSOCIATO_AL_VECCHIO_IP>

A questo punto si può creare l'istanza openflow e indicare l'ip del controller.

> [admin@MikroTik] > /openflow add name=OF_switch controllers=<IP_CONTROLLER> disable=no

Se l'ip del controller è su un'altra subnet rispetto alla scheda, si deve impostare anche un gateway.

> [admin@MikroTik] > /ip route [admin@MikroTik] > add distance=1 gateway=<IP_GATEWAY>

[6] Netword Address Translation, utilizzato per mascherare una rete locale sotto un unico IP pubblico
[7] http://it.wikipedia.org/wiki/DHCP

Si associno le porte da gestire tramite openflow allo switch virtuale appena creato. La routerborad non ha una porta fisica dedicata al controller, quindi una delle 5 porte **non potrà** essere associata allo switch openflow, e servirà alla scheda per potersi collegare al controller. Aggiungere quella porta (nel nostro caso la porta 5) allo switch openflow significa perdere l'accesso alla macchina e dover procedere con un reset completo della scheda.

```
[admin@MikroTik] > /openflow port
[admin@MikroTik] > add disabled=no
interface=ether1 switch=OF_switch
[admin@MikroTik] > add disabled=no
interface=ether2 switch=OF_switch
[admin@MikroTik] > add disabled=no
interface=ether3 switch=OF_switch
[admin@MikroTik] > add disabled=no
interface=ether4 switch=OF_switch
```

Si avvii il controller floodlight e nei log si noterà la connessione della routerboard al controller.

Capitolo 9

Sicurezza di OpenFlow

OpenFlow presenta diversi vantaggi sotto il profilo della semplicità di manutenzione degli apparati e di scalabilità della rete, ma espone anche a delle problematiche di sicurezza.

9.1 Punti di forza di OpenFlow

Una struttura di rete, basata su switch e controller openflow presenta diversi vantaggi:

- **Facilità di sostituzione degli apparati**: per come è stato studiato openflow, nel caso di una sostituzione di un apparto (per problemi hardware o per cambio tecnologia), basterà configurgli un ip, collegarlo al controller, e lasciare che il controller gli scarichi le configurazioni necessarie.
- **Facilità di ampliamento della rete**: se necessito di un ampliamento della rete, semplicemente collego il nuovo switch alla rete e lo configuro in modo che si colleghi al controller.
- **Nessun pacchetto viene inoltrato se non autorizzato**: avendo come standard politiche di DROP dei pacchetti, se un flusso non viene autorizzato, questo viene bloccato.
- **Controllo completo dal controller**: dal controller openflow in ogni momento è possibile sapere ogni host dove è connesso, quanto traffico (e di che tipo) stà passando su un determinato link e lo stato di ogni porta.
- **Configurazione minima in periferia**: una volta reso operativo il controller e le applicazioni, la configurazione da fare in periferia sugli switch è l'assegnare un indirizzo ip ed indicare il controller a cui collegarsi.
- **Possibilità di implementare a livello di switch sistemi di autenticazione**: avendo la possibilità di gestire i flussi dei pacchetti, è possibile implementare semplicemente un'autenticazione di tipo 802.1x[1] col vantaggio che, nel caso di più mac address su una singola porta, le autenticazioni possono essere fatte a livello di singolo mac address.
- **Possibilità di effettuare policy sul traffico a velocità di switching**: sfruttando la possibilità di implementare delle regole per instradare traffico diverso su link diversi basandosi ad esempio sulla porta di destinazione, è facile creare bilanciatori di traffico o policy che però funzionano a velocità paragonabili a quelle di switching e non a quelle di routing.

[1] Per maggiori informazioni `http://www.ieee802.org/1/pages/802.1x-2004.html`

- **Possibilità di creare anelli**: è possibile creare anelli o loop tra più switch. Sarà compito del controller instradare i pacchetti su un link o su un altro, avendo di fatto creato un failover senza configurare nulla.

- **Possibilità di delegare allo switch il failover**: è possibile delegare allo switch di gestire il failover/load-balancing in modo nativa, scaricando di fatto il carico di lavoro del controller.

9.2 Punti deboli di OpenFlow

Una struttura di rete, basata su switch e controller openflow presenta però anche degli svantaggi:

- **Necessità di hardware compatibile**: tutto l'hardaware deve supportare openflow.

- **Necessità di uno o più controller centrale**: nel caso il controller si fermasse, tutta la rete sarebbe paralizzata. É quindi necessario, nel caso di realtà medio grandi (che sono quelle che possono avere dei vantaggi da openflow), di prevedere dei controller di backup.

- **Criticità del canale di comunicazione**: un eventuale attacco al sistema che riuscisse a consentire l'accesso al canale di comunicazione tra uno switch ed il controller, permetterebbe di poter amministrare tutto ciò che è collegato a valle.

- **Criticità del collegamento con le applicazioni**: un eventuale attaccante che riuscisse ad avere accesso alle applicazioni che comandano lo switch, oppure al flusso di dati tra applicazioni e controller, si troverebbe ad avere accesso a tutta la rete, e poter indirizzare il traffico a piacimento.

9.3 Un caso reale

Un'azienda che ha implementato OpenFlow in modo nativo sulla sua rete è Google. L'attuale rete openflow di google è estesa a livello mondiale, come si vede in figura 9.1 e trasporta un volume di traffico considerevole 9.2.

Gli attuali servizi che richiedono banda in modo sostanzioso, offerti dalla rete di google sono principalmente[11].
Come citato in figura 9.2, se google fosse un ISP (Internet Service Provider), ad oggi sarebbe il secondo più grande carrier al mondo.

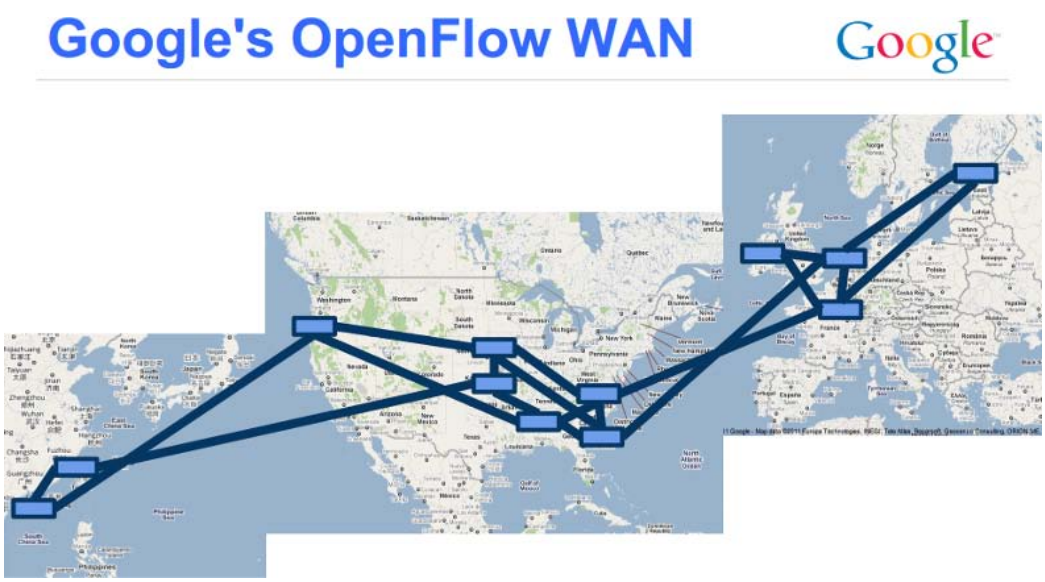

Figura 9.1: Disposizione rete WAN google

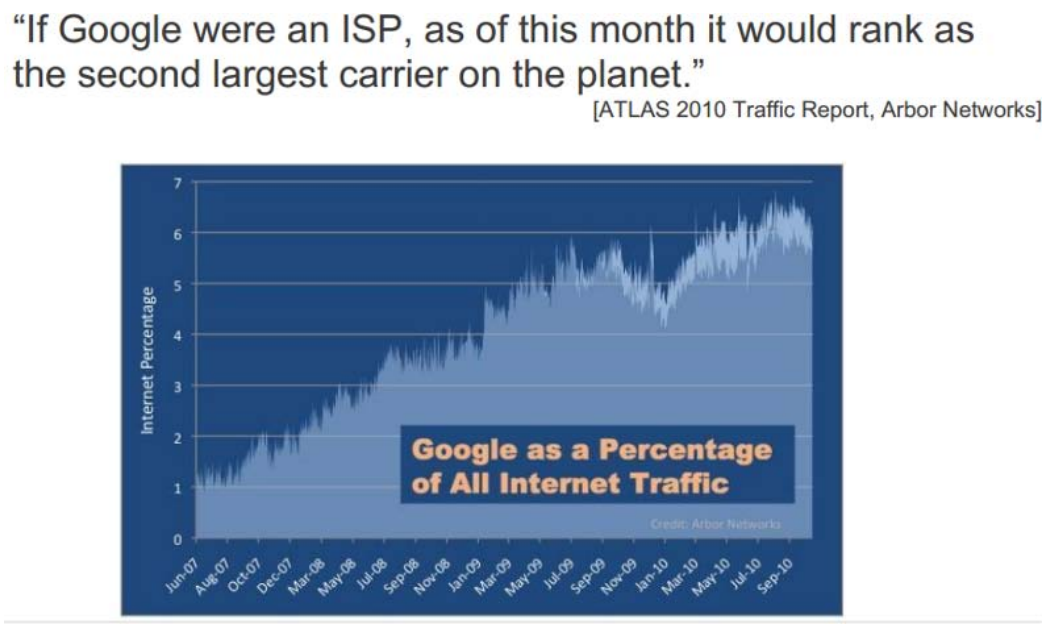

Figura 9.2: Traffico generato da Google

Figura 9.3: Connesioni tra due datacenter

- **Youtube**: servizio di video via web `http://www.youtube.com`
- **Ricerca Web**: il motore di ricerca "Google" `http://www.google.com`
- **Google+**: social network
- **Google Photos e Hangout**: servizio di storage di foto e di chiamata/videochiamata
- **Google Maps**: mappe consultabili liberamente online
- **Google App, Android e Chrome**: servizi messi a disposizione per le aziende da google, il play store di android, il browser Chrome e i relativi aggiornamenti.

Attualmente google ha due reti diverse; una rete pubblica dove gli utenti si connettono per usufruire dei servizi di google ed un'altra rete interna che collega tutti i datacenter di google. Ogni nodo pubblico è collegato ad altri attraverso una rete pubblica come in figura 9.3.

Il lavoro di google per implementare openflow è iniziato nell'estate del 2010 per arrivare a compimento ad inizio 2012 con lo scopo di ottimizzare i percorsi tra i nodi, gestire in modo più efficiente il down di un link/datacenter, ed ottimizzare il consumo di banda tra i vari datacenter al fine di poter offrire maggiore uptime nel servizio Il passaggio è stato fatto gradatamente su tutta la rete fino ad arrivare ad una rete totalmente gestita da openflow (figura 9.4).

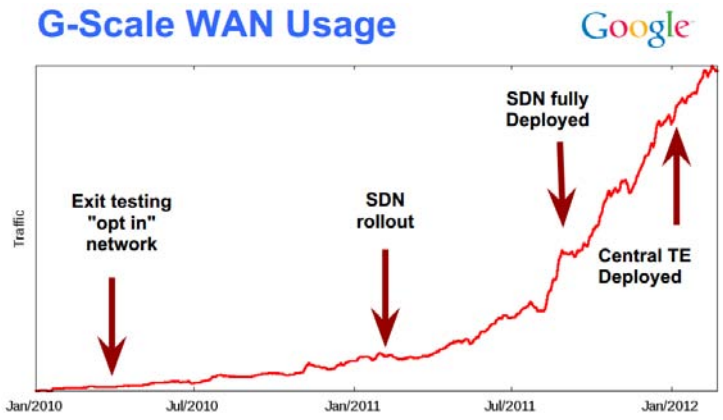

Figura 9.4: Grafico della banda nella migrazione verso openflow

Capitolo 10

Esercizi e simulazioni

Raccolta di esperienze didattiche su OpenFlow.

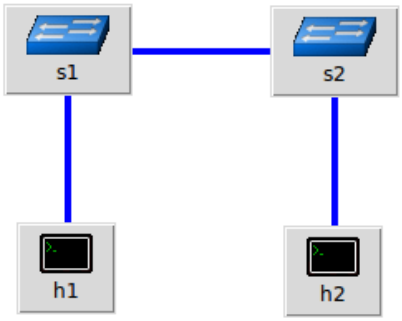

Figura 10.1: Schema esperienza 1

No.	Time	Source	Destination	Protocol	Length	Info	VLAN
	7.672684000	42:57:ac:e8:e2:61	Broadcast	ARP	42	Who has 10.0.0.2? Tell 10.0.0.1	
	7.672884000	5a:2c:2b:db:cb:2c	42:57:ac:e8:e2:61	ARP	42	10.0.0.2 is at 5a:2c:2b:db:cb:2c	
	7.672893000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x249c, s	
	7.673052000	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x249c, s	
	8.671683000	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x249c, s	
	8.671712000	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x249c, s	
	12.677775006	5a:2c:2b:db:cb:2c	42:57:ac:e8:e2:61	ARP	42	Who has 10.0.0.1? Tell 10.0.0.2	
	12.677789006	42:57:ac:e8:e2:61	5a:2c:2b:db:cb:2c	ARP	42	10.0.0.1 is at 42:57:ac:e8:e2:61	

Figura 10.2: Screenshot Traffico analizzato

10.1 Analisi rete normale in modalità switch

Si prenda ad esempio una rete formata da due host (h1 ed h2) e da due normali switch (s1 ed s2) come da figura 10.1.

Una volta avviata la simulazione possiamo ragionevolmente ipotizzare che gli switch non conoscano i mac address dei due host. Lanciando wireshark sull'host fisico ed agganciandolo sulla scheda di rete s1-eth1 (quindi sulla prima scheda di rete dello switch 1) possiamo analizzare tutto il traffico che entra ed esce dalla scheda. Lanciando un ping da h1 verso h2 con il comando

h1 ping h2 -c 2[1]

vedremmo prima il traffico ARP e poi il traffico ICMP generato da h1. Utilizzando il comando *h1 ifconfig eth0* ed *h2 ifconfig eth0*, alla voce "HWaddr" si possono vedere i mac address associati alle schede virtuali e ricostruire il traffico tra i due host per rispondere al ping.

Se ipotizziamo che l'host 1 abbia mac address 00:00:00:00:00:01 e l'host 2 00:00:00:00:00:02, la sequenza ricostruita è quella illustrata in tabella 10.1. Si nota che all'inizio i due host si scambiano i rispettivi mac address (e gli

[1]L'opzione "-c 2" permette di bloccare il comando in automatico dopo aver eseguito 2 ping.

Pacchetto	Sorgente	Destinazione	Contenuto
ARP	00:00:00:00:00:01	Broadcast	Who has 10.0.0.2? Tell 10.0.0.1
ARP	00:00:00:00:00:02	00:00:00:00:00:01	10.0.0.2 is at 00:00:00:00:00:02
ICMP	10.0.0.1	10.0.0.2	ICMP Echo request
ICMP	10.0.0.2	10.0.0.1	ICMP Echo reply

Tabella 10.1: Tabella traffico arp e icmp

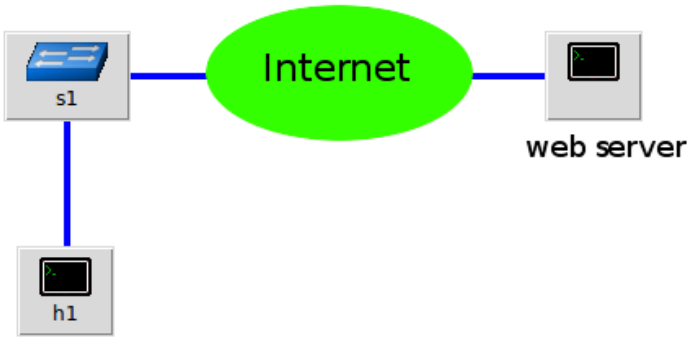

Figura 10.3: Schema esperienza 2

switch li memorizzano associandoli alle porte), e poi inizia il vero e proprio traffico, nel nostro caso traffico ICMP.

Se si permettesse solamente il traffico ICMP e non il traffico ARP, non sarebbe mai possibile vedere la richiesta "ICMP ECHO" (ipotizzando verosimilmente che switch ed host non abbiano dichiarazioni ARP statiche al loro interno).

10.2 Analisi generica connessione di rete

L'esperienza è simile alla precedente solamente che questa volta si analizza l'intera connessione. Con riferimento allo schema di figura 10.3, si analizzi una sessione telnet verso l'esterno. Espandendo i pacchetti catturati con wireshark si possono notare ip sorgente, ip destinazione, porta sorgente e la porta destinazione. Si noti il mac address destinazione del pacchetto (il pacchetto TCP è incapsulato all'interno del frame ethernet di trasporto, per cui il mac address destinazione è quello del gateway). Un esempio è visibile in figura 10.4

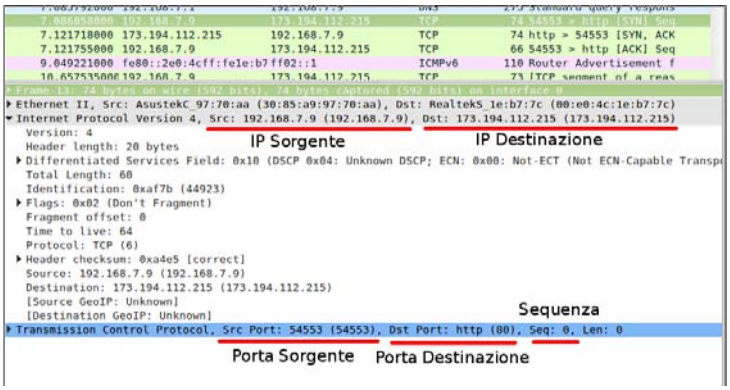

Figura 10.4: Analisi pacchetti con wireshark di una connessione telnet

Si può ad esempio collegarsi via telnet a google sulla porta 80 (servizio WEB)

Come primo esempio è consigliabile ricevere un errore, in quanto la home page è abbastanza sostanziosa come traffico dati. In italico le risposte del server.

$ telnet www.google.it 80
Trying 173.194.112.215...
Connected to www.google.it.
Escape character is '^]'.
GET /
HTTP/1.0 302 Found
Cache-Control: private
... The document has moved ... Connection closed by foreign host.

Se invece volessimo scaricare la vera home page di google, dovremmo prima dire al server per che sito ci stiamo collegando

$ telnet www.google.it 80
Trying 173.194.112.207...
Connected to www.google.it.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.google.it

HTTP/1.1 200 OK
Date: Tue, 24 Jun 2014 19:21:26 GMT
Expires: -1

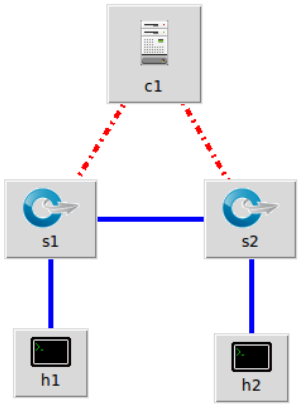

Figura 10.5: Schema "Rete Openflow"

Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie:
domain=.google.it ...
...

10.3 Rete Openflow

Si prenda come esempio la rete schematizzata in figura 10.5. É presente nella cartella *Documenti/Template_ miniedit/2host.mn*, il template da caricare in miniedit per eseguire la simulazione.

Caricato il file, si configuri l'ip del controller. Lanciando prima floodlight in modalità proattiva e poi in modalità reattiva è possibile analizzare i flow inseriti ed inserirli tramite le API come illustrato nel capitolo 7.1. É possibile utilizzare anche Avior per inserire le regole nel controller e visualizzare i flow attivi.

10.4 Rete Openflow complessa

Prendendo come esempio la rete di figura 10.6, è possibile applicare lo stesso procedimento utilizzato nell'esperienza precedente per creare un percorso tra l'host 1 e l'host 8, attraversando di fatto quasi tutti gli switch. É presente il template nella cartella *Documenti/Template_ miniedit/albero.mn*, da caricare in miniedit per eseguire la simulazione. Si può testare anche lo script

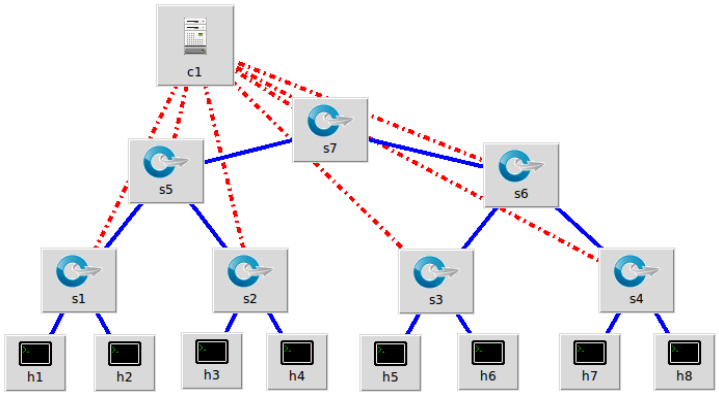

Figura 10.6: Schema prima esperienza openflow

circuitpusher.py che potrebbe essere a tutti gli effetti una delle applicazioni di floodlight, come spiegato nel capitolo 7.2.

10.5 Bilanciamento del traffico

Lo sviluppo di Floodlight ha avuto un notevole rallentamento ed è limitato al supporto della versione 1.0 di Openflow[2]. La versione 1.0 di Openflow permette di effettuare tutte le esperienze fin qui descritte ma non permette di lavorare a livello IP sui pacchetti.
Per avere un'idea della differenza e dell'evoluzione del protocollo nelle varie versioni si riporta in figura 10.7 lo schema di principio di openflow versione 1.0, mentre in figura 10.8 lo schema di principio di openflow versione 1.3.
In figura 10.9 e figura 10.10 si riportano la descrizioni di una generica flow table per openflow versione rispettivamente 1.0 e 1.3.
In figura 10.11 e figura 10.12 si riporta l'evoluzione dei messaggi controller-to-switch.

Per poter quindi sfruttare le potenzialità di openflow si è costretti a cambiare controller. Il controller scelto è opendaylight[13].
Questo controller dispone del supporto per Openflow 1.3 e successivi, è appoggiato da diversi grossi produttori (Brocade, Cisco, HP, IBM, Juniper...), ed è di facile installazione ed utilizzo. Una volta eseguito il processo (lanciando lo script *run.sh* presente nella directory *opendaylight*), si posiziona in ascolto sulla porta 6633 per ricevere il collegamento dagli switch e sulla

[2]Per maggiori informazioni `http://www.openflowhub.org/display/floodlightcontroller/FAQ+Floodlight+OpenFlow+Controller`

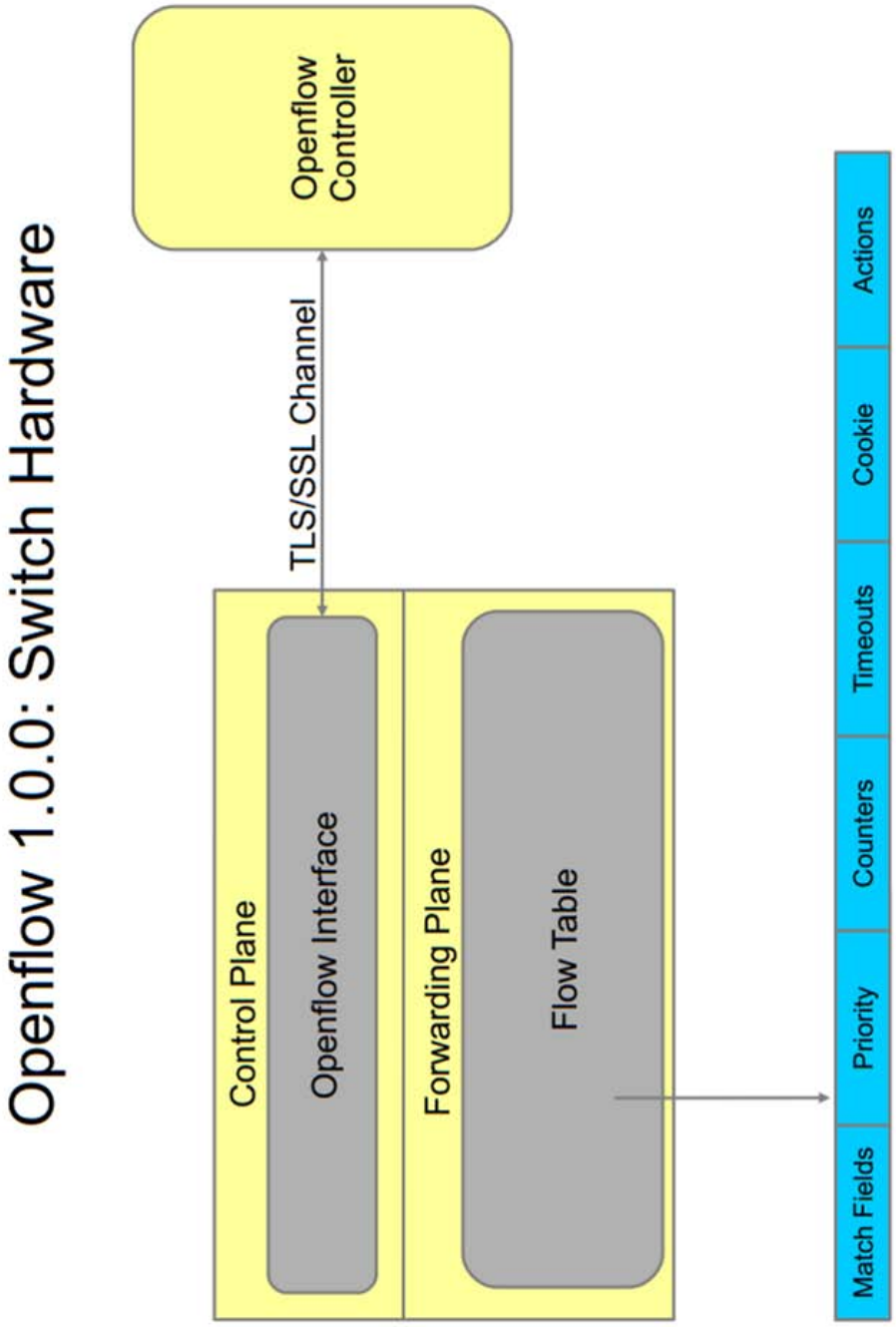

Figura 10.7: Schema di principio OpenFlow versione 1.0[12]

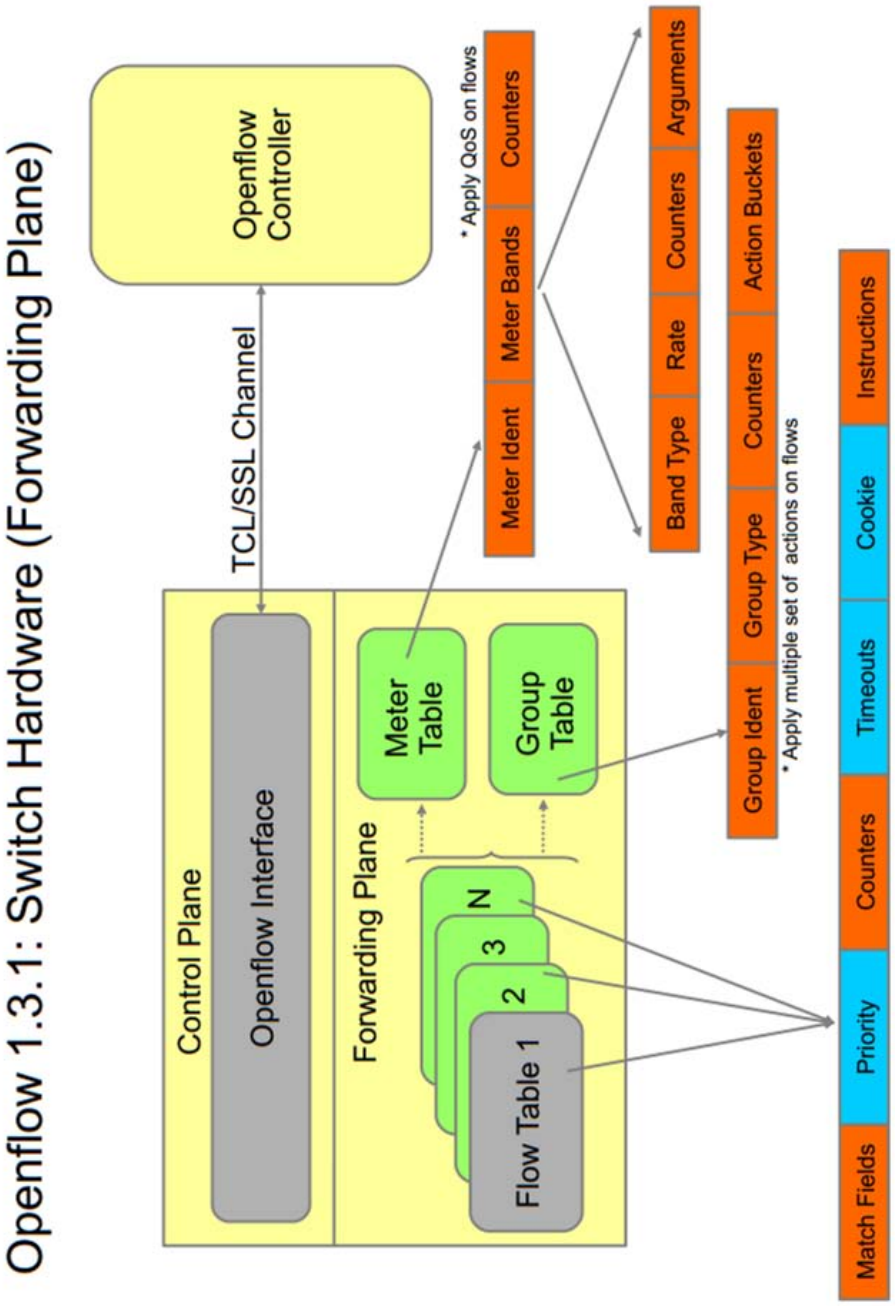

Figura 10.8: Schema di principio OpenFlow versione 1.3[12]

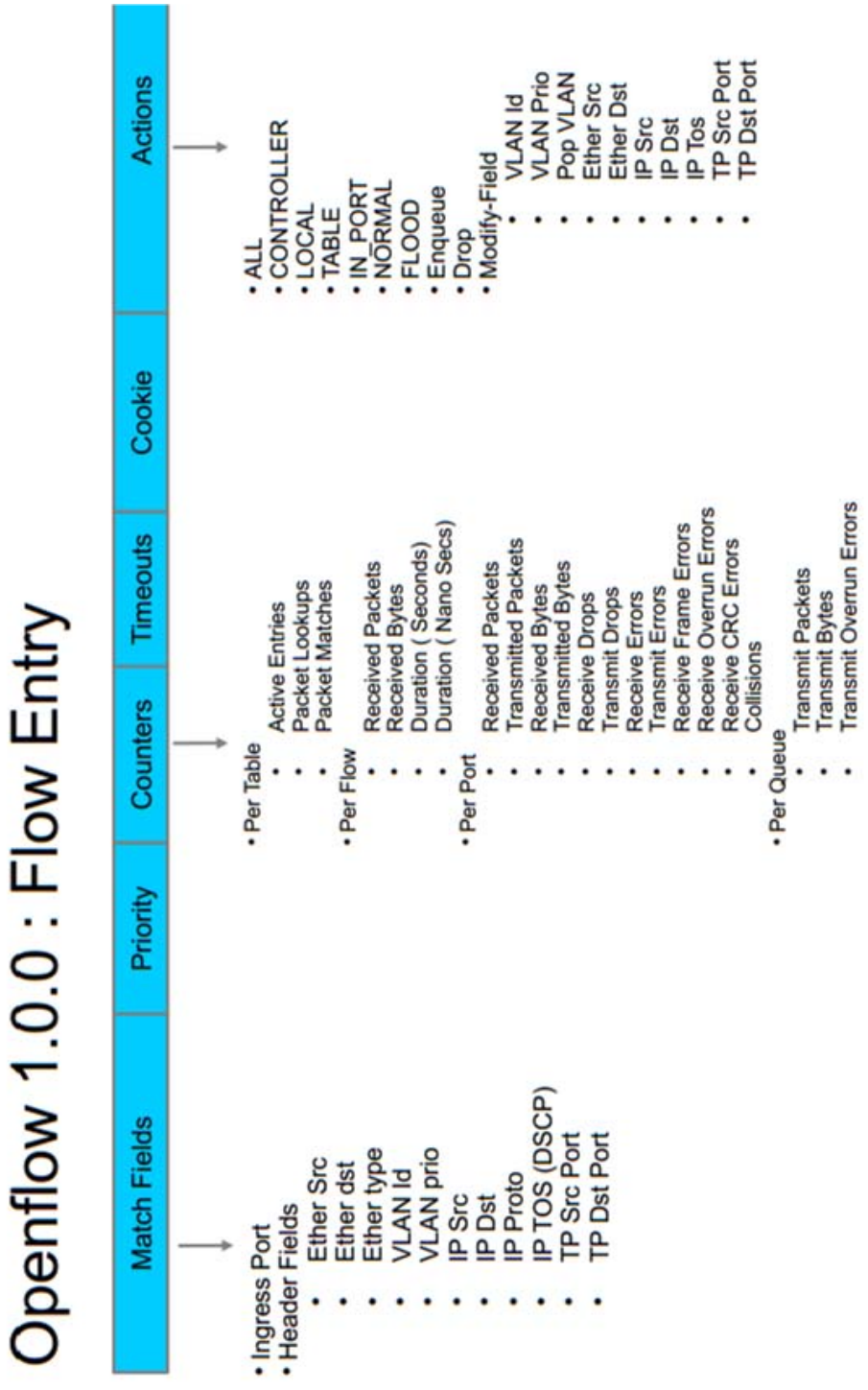

Figura 10.9: Schema Flow Entry per OpenFlow versione 1.0[12]

Figura 10.10: Schema Flow Entry per OpenFlow versione 1.3[12]

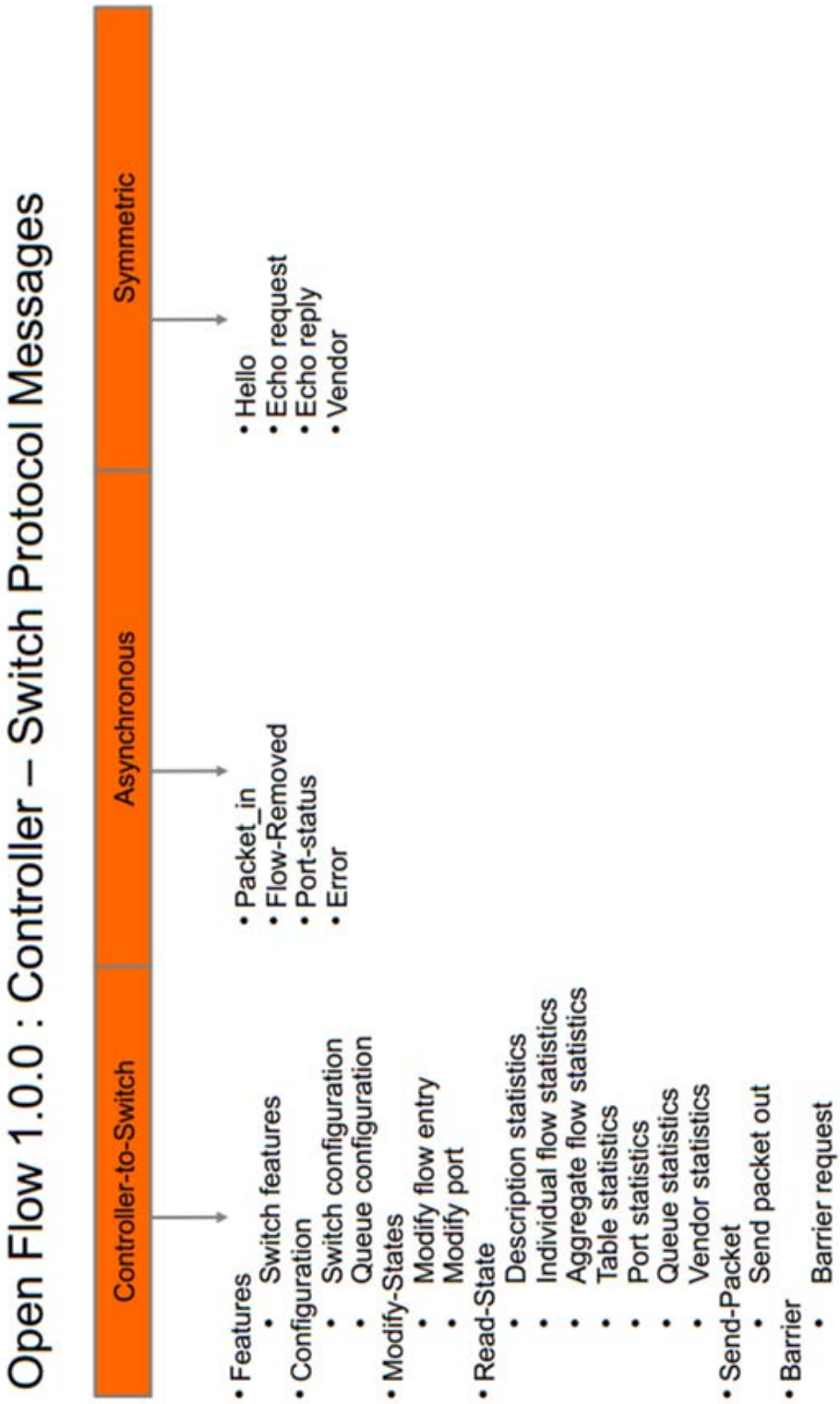

Figura 10.11: Schema messaggi Controller to Switch per OpenFlow versione 1.0[12]

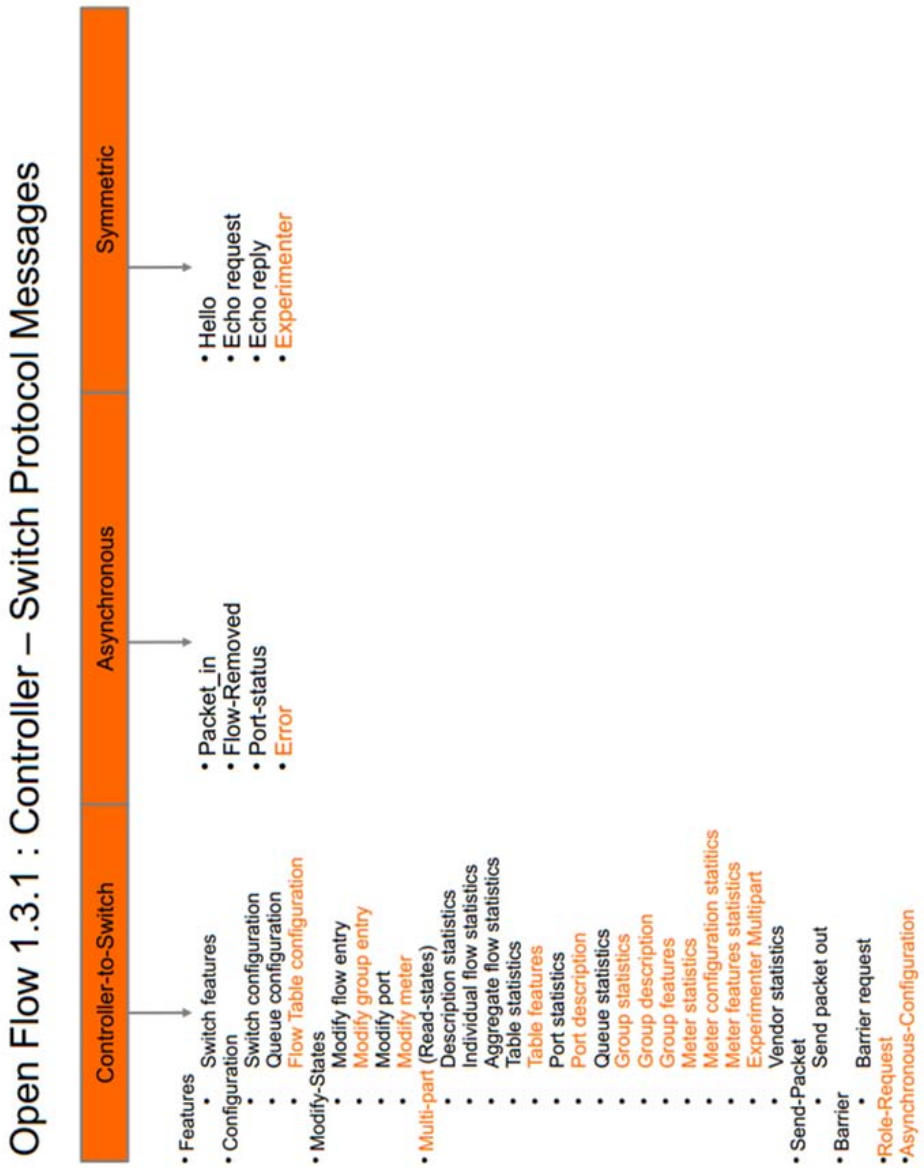

Figura 10.12: Schema messaggi Controller to Switch per OpenFlow versione 1.3[12]

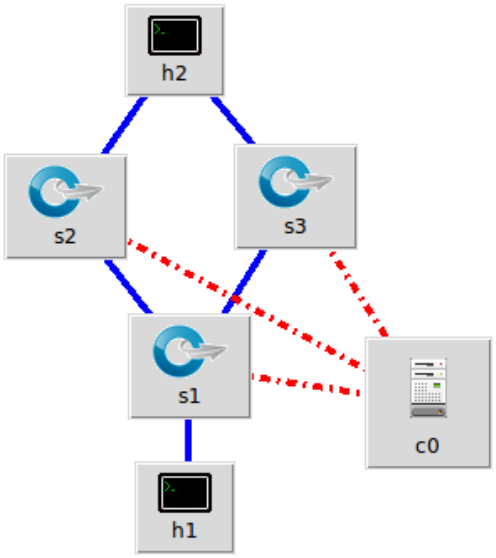

Figura 10.13: Schema esperienza bilanciamento traffico

porta 8080 fornisce un'interfaccia di amministrazione. Il login di default è *admin/admin*. A questo punto si può replicare la rete schematizzata in figura 10.13 con gli apparati mikrotik ed eseguire una divisione del traffico tra i 2 link. Per poter simulare questa rete tramite mininet è necessario il supporto ad openflow versione 1.3 del pacchetto openvswitch. Questa versione è disponibile nativamente in debian testing ma, a causa delle dipendenze dalla versione del kernel e delle librerie di sistema, non è portabile su debian stable. É stata preparata quindi un'immagine di una macchina virtuale, presente in *Macchina_ Virtuale/OpenFlow-Switch.ova* la quale ha il supporto per openflow versione 1.3. Per alcuni esempi di messaggi che possono essere analizzati, si veda `http://sdnhub.org/tutorials/openflow-1-3/`.

Bibliografia

[1] Wikipedia, "Software-defined networking." http://en.wikipedia.org/wiki/Software-defined_networking. [Verificato in data 05 Aprile 2014].

[2] N. McKeown, "Talks." http://tiny-tera.stanford.edu/~nickm/talks.html. [Verificato in data 05 Aprile 2014].

[3] Wikipedia, "Content-addressable memory." http://en.wikipedia.org/wiki/Content-addressable_memory. [Verificato in data 05 Aprile 2014].

[4] Open Network Foundation, "Openflow." https://www.opennetworking.org/sdn-resources/onf-specifications/openflow v 1.4.0. [Verificato in data 05 Aprile 2014].

[5] Mininet, "An instant virtual network on your laptop (or other pc)." http://www.mininet.org/. [Verificato in data 05 Aprile 2014].

[6] O. vSwitch Project, "Why open vswitch." http://git.openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob_plain;f=WHY-OVS;hb=HEAD. [Verificato in data 05 Aprile 2014].

[7] F. Team, "Project floodlight." http://www.projectfloodlight.org/floodlight/. [Verificato in data 05 Aprile 2014].

[8] M. College, "Marist sdn lab." http://openflow.marist.edu/avior. [Verificato in data 05 Aprile 2014].

[9] G. Gee, "Miniedit software." http://gregorygee.wordpress.com/category/miniedit/. [Verificato in data 05 Aprile 2014].

[10] IEEE, "802.3af." http://en.wikipedia.org/wiki/PoE. [Verificato in data 05 Aprile 2014].

[11] U. Holzle, "Openflow @ google." http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf. [Verificato in data 05 Aprile 2014].

[12] A. L. F. C. Project, "Openflow 1.3 support for opendaylight." https://wiki.opendaylight.org/images/d/dc/Openflow1.3_Support_for_Opendaylight.pdf. [Verificato in data 05 Aprile 2014].

[13] A. L. F. C. Project, "Opendaylight sdn." http://www.opendaylight.org/. [Verificato in data 05 Aprile 2014].