



UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Progettazione e sviluppo di una interfaccia web-based per la  
visualizzazione e il monitoraggio di reti basate su SDN

Relatori:

Dott: Antonio Viridis

Prof: Enzo Mingozzi

Candidato:

Gabriele Pianigiani

ANNO ACCADEMICO 2022/2023

# Indice

## Sommario

Introduzione .....	3
Capitolo 1 .....	5
Reti tradizionali e Data Center .....	5
1.1 Router Tradizionali .....	5
1.2 Data Center e Virtualizzazione .....	6
1.2.1 Data Center Networking .....	6
1.2.2 Requisiti di un Data Center .....	7
1.3 Mercati Verticali e Mercati Orizzontali .....	7
Capitolo 2 .....	8
SDN .....	8
2.1 Perché SDN? .....	8
2.2 SDN .....	8
2.3 Architettura di SDN .....	9
2.4 Vantaggi SDN .....	10
2.5 Controller SDN .....	11
2.5.1 OpenFlow .....	11
2.5.2 Controllore Floodlight .....	12
2.5.3 REST API .....	13
2.5.4 SDN proattivo e reattivo .....	14
Capitolo 3 .....	15
Design ed Implementazione .....	15
3.1 Backend .....	15
3.1.1 Pagina principale .....	16
3.1.2 Tabelle .....	16
3.1.3 Update .....	17
3.1.4 Funzioni .....	18
3.2 Frontend .....	20
3.2.1 Creazione topologia .....	20
3.2.2 Aggiornamento asincrono .....	21
Capitolo 4 .....	23
Interfaccia web .....	23
4.1 Topology .....	23
4.2 Flow table .....	27

4.3 Port table.....	27
4.4 Throughput table.....	28
Capitolo 5 .....	29
Testing .....	29
5.1 Mininet.....	29
5.2 Iperf.....	30
5.2.1 Test con iperf .....	32
5.3 Deployment.....	35
Bibliografia .....	36

# Introduzione

Una rete basata su SDN (Software Defined Networking) è un'architettura che consente la gestione di una rete informatica completamente basata su software. Nelle reti tradizionali sono presenti due componenti fondamentali per la gestione di esse: il piano dati ed il piano di controllo, i quali svolgono funzioni distinte e lavorando in modo cooperativo costituiscono parte integrante del buon funzionamento di una rete. Piano dati e piano di controllo nelle reti tradizionali sono implementati in hardware dedicati e si trovano in un unico dispositivo, tipicamente un router. Con le reti SDN non si ha più questo tipo di architettura, ma il piano di controllo risulta astratto dall'hardware, ciò significa che il controllo della rete è centralizzato, intelligente e viene gestito tramite software.

Grazie ai suoi numerosi vantaggi rispetto al concetto di rete classico, SDN è interessante per un gran numero di applicazioni, tale architettura è adatta per i seguenti fini:

- **Quality of Service (QoS):** la panoramica centrale di tutti i nodi di rete consente all'amministratore di reagire in tempo reale alle conoscenze acquisite e regolare di conseguenza il traffico dati in modo da poter fornire in ogni momento la larghezza di banda promessa a tutti i clienti.
- **Gestione dei dispositivi indipendente dal produttore:** la focalizzazione su un protocollo uniforme come OpenFlow rende SDN una soluzione eccellente per combinare e gestire dispositivi di diversi produttori in un'unica rete.
- **Ampliamento funzionale della rete indipendente dal produttore:** la libertà della tecnologia SDN è anche una buona soluzione per scenari in cui le reti dovrebbero in ogni momento poter essere facilmente espandibili con nuove funzioni.
- **Routing guidato dall'applicazione:** SDN crea la base per consentire alle applicazioni di terze parti di cambiare e adattare i percorsi dei dati in rete, per questo l'unità di controllo deve disporre di un'interfaccia adeguata.
- **Definizione e distribuzione centralizzata delle politiche di sicurezza (Security Policies):** le politiche di sicurezza possono essere facilmente ed efficacemente trasmesse ai singoli switch di rete tramite l'unità di controllo centrale.

Un elemento fondamentale nelle reti SDN è il controllore SDN, nel progetto il controllore utilizzato è Floodlight, il quale è stato sviluppato seguendo lo standard OpenFlow. I controllori esistenti come lo stesso Floodlight, forniscono delle interfacce grafiche che consentono di visualizzare informazioni relative alla rete, come ad esempio, il numero di dispositivi che costituiscono la rete, indirizzi IP e MAC di tali dispositivi, a quali switch e a quali porte di essi sono collegati gli host della rete, il grafo rappresentante la topologia della rete interessata ecc. Oltre all'interfaccia i controllori SDN forniscono una serie di API che consentono di configurare i dispositivi di rete, di ottenere alcune informazioni sui dispositivi e alcune statistiche.

L'interfaccia web d'interesse, ovvero quella fornita dal controllore Floodlight, a differenza di altri controllori, consente solo una visualizzazione statica di queste informazioni, non fornisce statistiche dinamiche sul traffico in tempo reale e sul carico a cui è sottoposta la rete, inoltre Floodlight mostra un grafo per rappresentare la topologia, nel quale non vengono indicate altre informazioni se non gli identificatori dei dispositivi ed il numero di pacchetti trasmessi e inviati da ciascuno switch.

L'obiettivo del progetto consiste nel creare un'interfaccia web-based che sopprima alle mancanze delle interfacce presentate da alcuni controllori nel mercato come Floodlight e mostri in modo grafico e diretto la topologia di una rete informatica basata su SDN consentendo ad un ingegnere di rete di monitorare e valutare il suo stato nel passare del tempo, controllando traffico, throughput, quantità di byte che vengono trasmessi nei vari link della rete in tempo reale. Tutte queste informazioni verranno ricavate utilizzando le API fornite dal controllore.

Tutto ciò viene mostrato direttamente all'interno del grafo rappresentante la topologia della rete osservata in modo da consentire un'analisi diretta e immediata del sistema, inoltre vi è la possibilità di visualizzare le stesse informazioni in maniera più approfondita ed estesa in apposite sezioni dell'interfaccia in cui vengono mostrate tabelle più dettagliate, le quali vengono aggiornate in tempo reale.

# Reti tradizionali e Data Center

Prima di passare alle reti SDN occorre approfondire alcuni concetti riguardanti le reti tradizionali menzionate in precedenza e l'evoluzione dei data center, i quali hanno spinto e contribuito alla nascita di reti basate su software.

## 1.1 Router Tradizionali

Un router tradizionale, tipicamente, implementa in hardware due elementi fondamentali:

- **Piano di Dati**

Il quale si occupa di eseguire delle fasi precise, in generale, riceve un pacchetto, controlla una struttura dati interna (ad esempio una tabella) ed effettua una decisione su come gestire quel pacchetto (switching, forwarding a livello 3). Tale struttura dati è stata riempita dagli algoritmi di routing presenti nel piano di controllo.

- **Piano di Controllo**

È il piano in cui girano i protocolli di routing (BGP, OSPF ecc), si occupa di riempire la struttura dati usata nel piano dati, prende decisioni su come gestire, instradare ed elaborare i dati, inoltre comunica con i control plane degli altri router.

Tutto questo è implementato in un unico dispositivo, è efficace nell'aggiornamento se avviene un cambiamento nella rete, l'unico problema consiste nel tempo di convergenza di questi protocolli di routing, il quale risulta essere molto lungo (decine di secondi) soprattutto a fronte di reti di grandi dimensioni.

Nelle reti SDN quasi tutto il data plane è implementato in hardware, mentre il control plane è implementato interamente in software per essere il più flessibile possibile.

Nei moderni calcolatori si era arrivati al punto tale che il bottleneck non era quasi più la rete, ma gli apparati di calcolo.

## 1.2 Data Center e Virtualizzazione

L'introduzione dei data center ha dato un grande scossone al mondo delle reti, essi hanno architetture tutte loro, con batterie di decine di switch e router.

Al giorno d'oggi nei data center si ha una virtualizzazione, questo insieme di hardware, CPU e dischi sono ora utilizzati come una base per una infrastruttura virtualizzata che permette di creare sopra questo hardware generico una serie di servizi e applicazioni, slegati dall'hardware effettivo che li ospita.

Poiché le reti convenzionali basate su hardware fisico raramente soddisfano le esigenze delle aziende moderne, rispetto alle tradizionali infrastrutture, questi data center, che consentono ai clienti di accedere a risorse informatiche virtualizzate, sono caratterizzati da un elevato grado di flessibilità e da un eccellente controllo dei costi: a differenza di un framework hardware fisso, le risorse desiderate possono essere scalate in qualsiasi momento in modo agevole.

### 1.2.1 Data Center Networking

Le macchine virtuali devono essere connesse tra di loro, il cliente del servizio ha la percezione di interagire con una singola macchina, in realtà l'architettura è composta da centinaia di rack, ogni rack contiene decine di Server Blade, ognuno dei quali ha una grande potenza di calcolo.

Ogni rack possiede uno switch detto Top of Rack, il quale mette in comunicazione tutti i Server Blade all'interno di un unico rack.

Gli switch Top of Rack sono messi in comunicazione tramite switch di Aggregazione, i quali a loro volta comunicano tramite i Core Switch, questi ultimi sono direttamente collegati alla WAN.

Le comunicazioni all'interno del data center non sono solo comunicazioni Nord-Sud, ovvero tra cliente e data center, ma anche comunicazioni Est-Ovest ovvero tra i vari servizi all'interno del sistema.

Tale architettura presenta i seguenti problemi:

- Se viene distrutto il sistema, è probabile che vengano distrutte tutte le macchine;
- Tutte queste macchine sono molto vicine tra loro;
- I clienti sono tanti, cambiano spesso idea, hanno requisiti flessibili.

### 1.2.2 Requisiti di un Data Center

I principali requisiti di un data center rispetto alla rete sono:

- **Automazione elevata**
  - L'intervento umano deve essere il più possibile limitato, perché risulta molto lento e noioso.
  - Il cliente paga e automaticamente i server sono configurati e pronti a erogare il servizio che richiede il cliente (i server devono essere creati e distrutti molto velocemente).
- **Alta scalabilità**
  - Il numero di server cambia molto velocemente e presto potremo avere anche miliardi di processi server in un unico Data Center.
- **Multipathing**
  - Nasce la necessità di avere percorsi diversi per un'unica destinazione.
  - Il problema del Single Point of Failure rimane, se uno switch si guasta non è ammissibile che un servizio sia interrotto.
  - In caso di alto carico, aumenta anche l'efficienza perché posso distribuire il carico sui link.
- **Multi Tenancy**
  - I clienti sono tanti, ciascuno con requisiti diversi, molti in coesistenza in un'unica macchina fisica e ognuno vuole che il servizio fornito sia perfetto, i clienti richiedono il sistema come se fosse solo in esclusiva per loro.

### 1.3 Mercati Verticali e Mercati Orizzontali

Per diversi anni si è fatto utilizzo di sistemi monolitici, ovvero verticali, i quali sono estremamente dedicati ad un dominio specifico (ad esempio gestione di una banca).

Un produttore offre la macchina Server, un sistema operativo dedicato e applicazioni dedicate, questa tendenza si trovava nelle reti tradizionali, in passato (ma anche adesso) le reti si basavano su Hardware specializzato (router, switch) con sistema operativo specializzato (ad esempio Cisco IOS per Cisco) e applicazioni specializzate per configurazioni di alto livello.

Al giorno d'oggi la tendenza è quella di sviluppare:

- **Hardware più generico**
  - CPU general purpose, processori molto generici adatti a diversi scopi, quindi molto più flessibili.
- **Sistemi operativi generici**
  - Mascherano la complessità della CPU.
  - Espongono primitive ai programmatori che permettono di sviluppare programmi, ad esempio, per configurare un router o uno switch.



## Capitolo 2

# SDN

### 2.1 Perché SDN?

SDN ha trovato il suo sviluppo in quanto nel campo delle reti tradizionali si riscontrano alcuni limiti:

- Troppi tipi di control plane che risolvono diversi problemi di networking, e tutti lo risolvono in modo distribuito, ovvero ogni dispositivo ha un proprio control plane specifico.
- Sistemi molto chiusi, ci sono pochissimi progetti open source, ad esempio Cisco offre un sistema molto chiuso, ogni produttore ha dei protocolli specifici.
- I monopolisti (Cisco e altri) innovano poco perché non hanno abbastanza competizione nel mercato.
- L'accesso al livello dati deve essere effettuato direttamente sull'hardware.
- L'architettura è statica, quindi difficile da personalizzare.

C'è una grande differenza di velocità tra l'evoluzione di software e protocolli di rete, in generale, la velocità di evoluzione dei protocolli di rete è più lenta rispetto alla velocità di evoluzione dei software. Questo perché i protocolli di rete devono essere compatibili con una vasta gamma di dispositivi e tecnologie, il che richiede tempo e attenzione per garantire che i nuovi protocolli funzionino correttamente con quelli esistenti.

### 2.2 SDN

Software Defined Networking nasce intorno al 2010 e descrive un'architettura che consente una gestione della rete completamente basata su software, la rete viene programmata come una applicazione. In cosa si differenzia SDN rispetto al networking tradizionale?

Piano dati e piano controllo vengono separati fisicamente, gli oggetti di rete si occupano di effettuare le operazioni del piano dati e di centralizzare il piano di controllo.

Il piano dati risulta distribuito tra più dispositivi, mentre il piano di controllo risulta logicamente centralizzato, in quest'ultimo caso non ci sarà un unico dispositivo che gestisce tutto, sarà distribuito tra vari dispositivi per motivi di sicurezza, affidabilità e scalabilità, a differenza delle reti tradizionali il piano di controllo è separato dall'hardware, è implementato esclusivamente in software e per questo risulta quindi molto più flessibile in termini di amministrazione di rete rispetto ad altre architetture.

I dispositivi di rete (router e switch) in questo caso risultano meno "intelligenti", spesso sono solo costituiti da hardware, quindi sono molto più veloci.

## 2.3 Architettura di SDN

L'architettura SDN tipica raffigurata in figura 2.1, si compone di tre elementi:

### Livello Basso

Costituito da apparati di rete molto veloci ma non “intelligenti”, questi dispositivi sono Switch programmabili i quali:

- Ricevono un pacchetto.
- Effettuano una procedura di “match & action” con una struttura dati al loro interno.
  - Se non sanno cosa fare chiedono istruzioni al Control Plane.
  - In base alle istruzioni ricevute modificano le strutture dati di riferimento.
- Effettuano un'azione dettata dalla struttura dati stessa.

### Livello Medio

In questo livello risiede il Controller SDN, logicamente centralizzato. Il controller è a tutti gli effetti un sistema operativo, ospitato in una macchina hardware generica. Riceve indicazioni dalle applicazioni dei clienti per decidere come instradare un pacchetto di dati, il control plane le traduce in istruzioni per il livello basso (dispositivi) che passano tramite delle SouthBound API (SBI) come ad esempio OpenFlow.

Un programmatore può scrivere programmi che risiedono sul controllore o che non risiedono sul controllore ma che sfruttano le sue interfacce e questi programmi possono essere usati dai clienti.

### Livello Alto

Dove risiedono le applicazioni dei clienti, i quali spesso sono aziende che richiedono un servizio da una certa piattaforma.

Ai clienti non interessa del livello basso, comunicano solo con il control plane, utilizzando delle NorthBound API e quest'ultimo comunicherà con il livello basso.

Per le NorthBound API non ci sono standard che specificano come il controllore debba essere configurato.

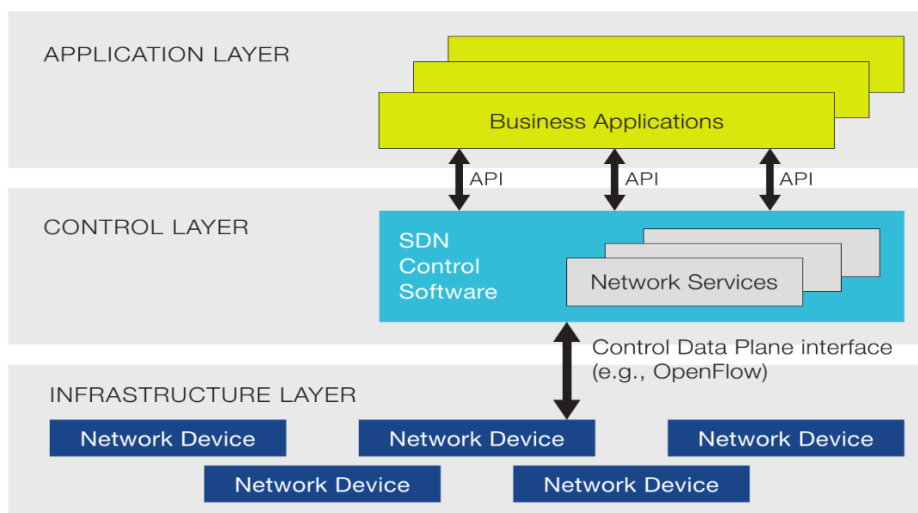


Figura 2.1.

## 2.4 Vantaggi SDN

Parallelamente alle richieste di potenza di calcolo dei computer, anche le richieste di prestazioni delle reti sono in continuo aumento. Mentre le reti digitali diventano sempre più grandi e complesse, aumentano contemporaneamente anche il grado di virtualizzazione e il desiderio di massima flessibilità e scalabilità. Poiché i dispositivi convenzionali, dotati di intelligenza propria e che elaborano gran parte dei processi in modo indipendente, non sono stati in grado di soddisfare questi requisiti, è stato sviluppato il concetto di Software Defined Networking.

I vantaggi rispetto alle reti tradizionali possono essere riassunti come segue:

- **Separazione del piano dati e controllo.**
- **Dispositivi semplici e più economici.**
- **Bassi costi di manutenzione e di amministrazione per l'intera rete.**
- **Maggiore automazione della rete:**
  - La gestione del servizio deve essere efficiente e veloce, per questo si va verso la virtualizzazione, perché creare o distruggere qualcosa costa davvero poco e inoltre i dispositivi non si guastano dal punto di vista fisico perché sono virtuali.
  - Inoltre, se la macchina fisica che ospita la macchina virtuale si guasta, automaticamente si può clonare la macchina virtuale, spostarla su un'altra macchina fisica e riavviarla.
- **Openness (Sistema molto aperto):**
  - Molto aperto, dato che i software di gestione delle macchine virtuali sono molto generici e sono indipendenti da come sono realizzati switch e router.
  - Adesso Cisco e i maggiori produttori possono avere concorrenza da aziende o da singoli sviluppatori molto più piccoli perché non serve produrre hardware, è sufficiente produrre solo il sistema operativo per il control plane.
- **Non è necessaria alcuna configurazione di singoli dispositivi o sistemi operativi.**
- **Consente l'allocazione dinamica e il monitoraggio delle risorse in tempo reale.**

## 2.5 Controller SDN

Elemento dell'architettura che è in grado di:

- Comunicare ad uno switch di cambiare la propria tabella se c'è un nuovo requisito da soddisfare.
- Il controller riceve una richiesta di istruzioni dallo switch e dopo aver eseguito dei calcoli invia risposta allo switch.
- Deve avere una visione della rete completa, chiede agli switch sotto di lui, quali sono i loro vicini, in modo da poter creare un grafo con la topologia della rete.
- Deve comunicare con altri controller attraverso le interfacce East-West, ovvero interfacce che servono per la comunicazione con dispositivi dello stesso livello.

In generale i controller sono un applicativo molto complesso, spesso sono scritti in Java.

Un controllore è costituito da:

- **Southbound API**
  - Openflow, per comunicare con gli switch
- **Moduli:** consentono di effettuare le seguenti operazioni:
  - Discovery: moduli e strutture dati per scoprire la topologia, scoprire quali sono i dispositivi attualmente attivi, scoprire quali sono gli End-Point nella rete (i Server).
  - Gestione dei Flussi.
  - Statistiche
- **Northbound API:** sono API di programmazione, che consentono al programmatore di scrivere programmi sul controllore stesso:
  - Rest API
  - Python API
  - Java API

Per le Southbound API esistono degli standard, mentre per le Northbound API non li abbiamo.

Il vantaggio principale di SDN rispetto alle architetture tradizionali consiste nella possibilità di riprogrammare le reti SDN, i cambiamenti sono applicati automaticamente dalle applicazioni esterne e dai processi di monitoraggio per reagire agli eventi esterni, un'interfaccia Northbound è esposta ai controllori per ottenere questi vantaggi.

### 2.5.1 OpenFlow

OpenFlow è uno dei primi e più diffusi standard SDN, definisce il protocollo di comunicazione in ambiente SDN che consente al controllore di comunicare con il piano di inoltro (switch, router, ecc.) per apportare modifiche alla rete.

Si occupa di mettere in comunicazione il controller e i vari switch, ovvero:

- Permette al controller di comunicare agli switch le proprie decisioni.
- Definisce l'astrazione della Flow Table.

Uno switch che sfrutta OpenFlow è composto da:

- **Data Path**
  - Riceve un pacchetto, legge i campi ed esegue l'azione.
- **Flow Table**
  - Tabella interpellata dal data path.
- **Control Path**
  - Elemento dello switch che comunica con il controller.
  - Aggiorna la Flow Table.

Openflow definisce:

- Come lo switch comunica con il controllore.
- Il comportamento che deve essere eseguito dallo switch in relazione a determinati match.
- Definisce un flusso come un insieme di pacchetti che sono trasferiti da un set di endpoint di rete verso un altro set di endpoint di rete.

OpenFlow può compiere varie azioni:

- **Switching:** forwarding di livello 2, ossia controlla l'indirizzo MAC di destinazione e si comporta di conseguenza.
- **Routing:** forwarding di livello 3, ossia controlla l'indirizzo IP di destinazione e si comporta di conseguenza.
- **Firewall:** scarta dei pacchetti (ad esempio può essere bloccato tutto ciò che passa nella porta 22).

## 2.5.2 Controllore Floodlight

In precedenza, è stato menzionato il controllore utilizzato durante il progetto, ovvero Floodlight, un controllore SDN sviluppato da una comunità di sviluppatori open source, è stato progettato per funzionare con switch, router e switch virtuali che supportano lo standard OpenFlow.

Il controllore Floodlight è in grado di modificare il comportamento dei dispositivi di rete attraverso un insieme ben definito di istruzioni, le quali risultano necessarie all'infrastruttura sottostante per la gestione del traffico. Ciò consente alle aziende di adattarsi meglio alle loro mutevoli esigenze e di avere un migliore controllo sulle proprie reti.

Il controllore Floodlight è un framework scritto in Java, può essere vantaggioso per gli sviluppatori perché offre loro la possibilità di adattare facilmente il loro software e sviluppare applicazioni, include REST API che facilitano la programmazione di interfacce.

Testato con switch fisici e virtuali compatibili con OpenFlow, il controllore Floodlight può funzionare in una varietà di ambienti e può coincidere con le applicazioni che le aziende hanno già a loro disposizione. Può anche supportare reti in cui gruppi di switch compatibili con OpenFlow sono collegati tramite switch tradizionali non OpenFlow.

Floodlight ha una struttura modulare, è costituito da moduli, divisi in moduli del controllore e moduli dell'applicazione, ciascuno dei quali implementa una funzionalità, i pacchetti in ingresso vengono elaborati in cascata, tutti i moduli inclusi ed il loro ordine sono specificati all'interno del file **floodlightdefault.properties**, file di configurazione del controllore utilizzato per configurarne il comportamento, questo file è mostrato nella seguente figura:

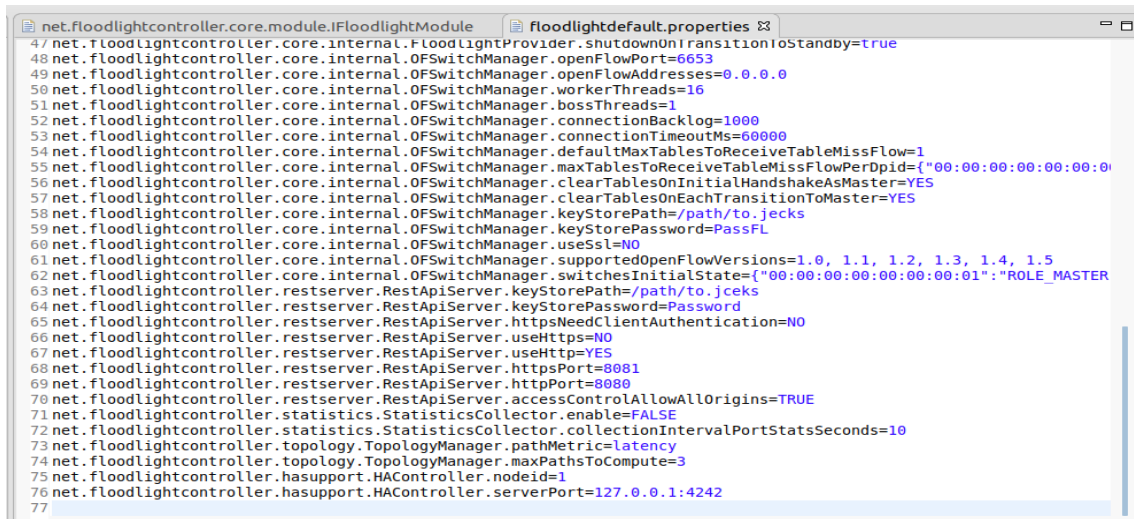
The image shows a screenshot of a text editor window displaying the 'floodlightdefault.properties' file. The editor has two tabs: 'net.floodlightcontroller.core.module.IFloodlightModule' and 'floodlightdefault.properties'. The 'floodlightdefault.properties' tab is active, showing a list of configuration properties for the Floodlight controller. The properties are listed in a single column, with line numbers 47 through 77 visible on the left. The properties include settings for the FloodlightProvider, OFSwitchManager, RestApiServer, StatisticsCollector, and TopologyManager. Some properties are set to true, false, or specific values like 6653, 0.0.0.0, 16, 1, 1000, 60000, 1, 1, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 8081, 8080, TRUE, FALSE, 10, 3, 1, and 127.0.0.1:4242. The properties are: 47 net.floodlightcontroller.core.internal.FloodlightProvider.shutdownOnTransitionToStandby=true, 48 net.floodlightcontroller.core.internal.OFSwitchManager.openFlowPort=6653, 49 net.floodlightcontroller.core.internal.OFSwitchManager.openFlowAddresses=0.0.0.0, 50 net.floodlightcontroller.core.internal.OFSwitchManager.workerThreads=16, 51 net.floodlightcontroller.core.internal.OFSwitchManager.bossThreads=1, 52 net.floodlightcontroller.core.internal.OFSwitchManager.connectionBacklog=1000, 53 net.floodlightcontroller.core.internal.OFSwitchManager.connectionTimeoutMs=60000, 54 net.floodlightcontroller.core.internal.OFSwitchManager.defaultMaxTablesToReceiveTableMissFlow=1, 55 net.floodlightcontroller.core.internal.OFSwitchManager.maxTablesToReceiveTableMissFlowPerDpid={"00:00:00:00:00:00"}, 56 net.floodlightcontroller.core.internal.OFSwitchManager.clearTablesOnInitialHandshakeAsMaster=YES, 57 net.floodlightcontroller.core.internal.OFSwitchManager.clearTablesOnEachTransitionToMaster=YES, 58 net.floodlightcontroller.core.internal.OFSwitchManager.keyStorePath=/path/to.jceks, 59 net.floodlightcontroller.core.internal.OFSwitchManager.keyStorePassword=PassFL, 60 net.floodlightcontroller.core.internal.OFSwitchManager.useSsl=NO, 61 net.floodlightcontroller.core.internal.OFSwitchManager.supportedOpenFlowVersions=1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 62 net.floodlightcontroller.core.internal.OFSwitchManager.switchesInitialState={"00:00:00:00:00:01":"ROLE\_MASTER"}, 63 net.floodlightcontroller.restserver.RestApiServer.keyStorePath=/path/to.jceks, 64 net.floodlightcontroller.restserver.RestApiServer.keyStorePassword=Password, 65 net.floodlightcontroller.restserver.RestApiServer.httpsNeedClientAuthentication=NO, 66 net.floodlightcontroller.restserver.RestApiServer.useHttps=NO, 67 net.floodlightcontroller.restserver.RestApiServer.useHttp=YES, 68 net.floodlightcontroller.restserver.RestApiServer.httpPort=8081, 69 net.floodlightcontroller.restserver.RestApiServer.httpPort=8080, 70 net.floodlightcontroller.restserver.RestApiServer.accessControlAllowAllOrigins=TRUE, 71 net.floodlightcontroller.statistics.StatisticsCollector.enable=FALSE, 72 net.floodlightcontroller.statistics.StatisticsCollector.collectionIntervalPortStatsSeconds=10, 73 net.floodlightcontroller.topology.TopologyManager.pathMetric=latency, 74 net.floodlightcontroller.topology.TopologyManager.maxPathsToCompute=3, 75 net.floodlightcontroller.hasupport.HAController.nodeid=1, 76 net.floodlightcontroller.hasupport.HAController.serverPort=127.0.0.1:4242, 77.

Figura 3.1: file di configurazione controllore floodlight.

### 2.5.3 REST API

Le API (Application Programming Interface), in generale, sono un insieme di definizioni e protocolli con i quali vengono realizzati e integrati software applicativi. Se si desidera interagire con un sistema per recuperare informazioni o eseguire una funzione, un'API facilita la comunicazione con il sistema che può così comprendere e soddisfare la richiesta.

È anche un mezzo con il quale un'organizzazione può condividere risorse e informazioni assicurando al contempo sicurezza, controllo e autenticazione, poiché stabilisce i criteri di accesso.

Un'API REST (REpresentational State Transfer), nota anche come API RESTful, è un'interfaccia di programmazione delle applicazioni (API o API web) conforme ai vincoli dello stile architetturale REST, che consente l'interazione con servizi web RESTful.

REST è un insieme di vincoli architetturali, non un protocollo né uno standard. Chi sviluppa API può implementare i principi REST in diversi modi.

Quando una richiesta client viene inviata tramite un'API RESTful, questa trasferisce al richiedente o all'endpoint uno stato rappresentativo della risorsa. L'informazione viene consegnata in uno dei diversi formati tramite HTTP: JSON, HTML, Python, PHP o testo semplice.

Affinché un'API sia considerata RESTful, deve rispettare i seguenti criteri:

- Un'architettura client-server composta da client, server e risorse, con richieste gestite tramite HTTP.
- Una comunicazione client-server stateless, che quindi non prevede la memorizzazione delle informazioni del client tra le richieste GET.
- Dati memorizzabili nella cache che ottimizzano le interazioni client-server.
- Un'interfaccia uniforme per i componenti, in modo che le informazioni vengano trasferite in un formato standard.

REST è un insieme di linee guida applicabili quando necessario, il che rende le API REST più rapide, leggere e con una maggiore scalabilità.

Floodlight definisce le API per accedere all'interfaccia Northbound utilizzando il paradigma RESTful, ogni controllore espone un insieme di risorse che possono essere utilizzate per recuperare informazioni di rete oppure per gestire cambiamenti di configurazione ed eseguire operazioni.

#### 2.5.4 SDN proattivo e reattivo

Il controllore SDN può funzionare secondo due approcci diversi a seconda delle esigenze e delle caratteristiche di una rete, da un lato, c'è la possibilità per il controllore di inoltrare nuove informazioni e modifiche a tutti i nodi di rete partecipanti tramite Broadcast o Multicast. Questo concetto di SDN cosiddetto proattivo è vantaggioso se la centralizzazione dell'intelligenza non ha importanza.

Più nodi ha una rete, più alto sarà il carico per la rete, il che si traduce in una limitata scalabilità. Nelle grandi reti, il modello reattivo è quindi un'alternativa diffusa: in questo caso, il piano di controllo assicura il corretto funzionamento di tutti i componenti mediante un trasferimento controllato delle informazioni, in cui sono bloccati solo i dispositivi interessati.

SDN proattivo presenta i seguenti vantaggi:

- Facilità d'implementazione;
- I pacchetti sono trasmessi attraverso il percorso più breve.

Lo svantaggio di questo metodo consiste nell'aumento del carico di rete con l'aggiunta di nuovi nodi.

SDN reattivo presenta il vantaggio di non inviare informazioni a chiunque, ma ciascun dispositivo riceve solo le informazioni per esso rilevanti. Lo svantaggio si ha nel caso in cui sono presenti problemi nel fornire informazioni, i quali portano automaticamente a ritardi.

# Design ed Implementazione

Il progetto è composto da un totale di 14 file e una cartella **images** contenente delle icone visualizzate nell'interfaccia (icone di switch, host, link e tabelle). In particolare, i file sono così organizzati:

- Undici file PHP:
  - index.php
  - function.php
  - flow\_table.php
  - port\_table.php
  - bandwidth\_table.php
  - session.php
  - update\_flow\_table.php
  - update\_port\_table.php
  - update\_bandwidth\_table.php
  - update\_summary.php
  - update\_topology.php
- Due file JavaScript:
  - script.js
  - autoUpdate.js
- Un file CSS:
  - style.css

Lo sviluppo di una qualsiasi applicazione web passa attraverso due fasi: backend e frontend, con backend si intende la fase di sviluppo che si occupa di elaborare e gestire i dati, mentre con frontend si intende la fase in cui vengono presentati i dati e si sviluppa l'interfaccia utente.

## 3.1 Backend

Nel caso specifico di questo progetto, il linguaggio di programmazione utilizzato in fase di backend è stato il PHP, in questa fase sono stati creati i file con estensione “.php” citati in precedenza, per prima cosa sono state definite le variabili necessarie a contenere le informazioni da mostrate nell'interfaccia, tutte le variabili utilizzate sono locali eccetto una variabile, in particolare un vettore denominato `$_SESSION["bandwidth_session_array"]` necessario per mantenere salvati i valori di byte trasmessi, byte ricevuti e tempo trascorso, in modo da poter calcolare i valori aggiornati di throughput per ciascuna interfaccia degli switch presenti nella rete. Una volta definite le variabili, lo step successivo è stato quello di creare una serie di funzioni, le quali facendo utilizzo delle REST API fornite dal controllore, sono in grado di interagire con il controllore stesso per ottenere le



informazioni da elaborare e successivamente da fornire all'utente. Definite le funzioni sono stati creati file contenenti la pagina principale e le tabelle estese che l'utente può visualizzare, in ciascuna di queste pagine vengono utilizzate le funzioni precedentemente create per accedere ai dati, i quali una volta ricavati, sono stati inseriti nelle sezioni opportune e visualizzati.

Una volta raccolte le informazioni, è necessario che queste vengano aggiornate in modo da mostrare statistiche in tempo reale, per questo nell'ultima fase sono stati creati i file di update, i quali interagendo con il controllore tramite le funzioni definite in precedenza, ottengono i dati aggiornati delle relative sezioni dell'interfaccia che richiedono un aggiornamento.

### 3.1.1 Pagina principale

In questa pagina l'ingegnere di rete che utilizza tale applicazione visualizza in alto il numero di componenti che costituiscono la rete, ovvero il numero di switch, numero di host e numero di link, dove per numero di link si intende il numero di link di collegamento tra i vari switch, i quali sono bidirezionali. Immediatamente sotto queste informazioni è presente una form che consente di scegliere un range di valori per il throughput su cui basarsi per monitorare la rete.

Sotto questi elementi è presente la topologia vera e propria, costituita da un grafo indicante i vari dispositivi (switch, host) e i link tra di essi, già all'interno della topologia è possibile visualizzare le informazioni utili per verificare lo stato della rete.

Di lato, sulla sinistra è presente una barra di navigazione che contiene il collegamento oltre che alla home page (pagina corrente), anche a tre tabelle nelle quali è possibile visualizzare informazioni e statistiche sulla rete in maniera più estesa e completa.

Le informazioni riguardanti il numero di componenti, e le statistiche visualizzate all'interno della topologia vengono ricavate attraverso le funzioni che sfruttano le API.

### 3.1.2 Tabelle

Le tabelle che consentono la visualizzazione dettagliata di statistiche e parametri di performance riguardanti la rete sono le seguenti:

- **Flow table:** la tabella mostra per ciascuno switch, il numero di flussi attivi, i pacchetti e i byte totali appartenenti a quel determinato flusso, oltre a queste informazioni viene mostrato il tempo trascorso in secondi da quando è iniziato il monitoraggio.
- **Port table:** per ciascuno switch la tabella visualizza ogni porta, per ciascuna porta indica lo stato (attiva o inattiva), il numero di pacchetti ricevuti e trasmessi, il numero di byte ricevuti e trasmessi fino a quell'istante, i pacchetti trasmessi e ricevuti che sono stati scartati, e quelli che sono arrivati con errori. Anche in questo caso l'ultima colonna della tabella mostra il tempo trascorso in secondi.
- **Throughput table:** mostra per ciascuno switch e per ciascuna interfaccia di ognuno di essi, il valore in bit/sec del throughput calcolato all'interno di una finestra temporale di 10 secondi, il valore di throughput massimo registrato fino a quell'istante e un contatore di tempo, il

quale indica da quanto tempo il throughput di una porta è nullo, queste ultime tre informazioni risultano raddoppiate in quanto per ciascuna interfaccia viene calcolato sia il throughput per i dati trasmessi che per i dati ricevuti. L'ultima colonna della tabella contiene il tempo trascorso in secondi.

Anche in questo caso, in ciascuno di questi file vengono chiamate le funzioni all'interno del file `function.php` (menzionate in precedenza) le quali restituiscono i dati che verranno visualizzati all'interno di ciascuna tabella.

Per il calcolo del throughput di ciascuna interfaccia esiste un API che consente di attivare la raccolta di statistiche della rete tramite l'interazione con il controllore e di conseguenza di ottenere il throughput di ciascun link, a causa di un problema non risolto del controllore, non è stato possibile utilizzare tale API, per questo i valori di throughput sono stati calcolati effettuando del post-processing con i dati raccolti e inseriti nella tabella delle porte.

Per calcolare il valore per la prima volta è stato diviso il numero di byte trasmessi nel caso del throughput in trasmissione, oppure i byte ricevuti nel caso di throughput in ricezione, con il tempo trascorso e moltiplicato il tutto per 8 in modo da ottenere il risultato in bps. Nelle volte successive alla prima, il throughput è stato calcolato dividendo la differenza tra i nuovi byte trasmessi/ricevuti appena aggiornati e i vecchi byte ricavati dall'aggiornamento precedente per la differenza tra il tempo trascorso appena aggiornato ed il tempo trascorso al momento del precedente aggiornamento, anche in questo caso il tutto viene moltiplicato per 8 in modo da ottenere il risultato in bps.

Per mantenere il numero di byte e il tempo trascorso ricavati ad un aggiornamento precedente, ogni volta che si ricavano nuovamente questi valori vengono memorizzati nel vettore di sessione precedentemente menzionato, in modo tale che rimangano per il calcolo del throughput all'istante successivo, il vettore di sessione è definito all'interno del file `session.php`.

### 3.1.3 Update

Nei file di update viene gestito l'aggiornamento delle varie sezioni dell'interfaccia, l'aggiornamento avviene in maniera asincrona, ogni 10 secondi tramite richieste AJAX effettuate dalle funzioni definite nel file `autoUpdate.js`.

I file di update sono:

- **`update_flow_table.php`;**
- **`update_port_table.php`;**
- **`update_bandwidth_table.php`;**
- **`update_summary.php`;**
- **`update_topology.php`.**

All'interno dei file `update_flow_table.php`, `update_port_table.php`, `update_bandwidth_table.php` non si fa altro che ricavare nuovamente i dati attraverso le funzioni contenute in `function.php`, inserire questi dati in una tabella HTML e assegnare la tabella in HTML ad una variabile che viene

restituita. La stessa cosa viene svolta in `update_summary.php`, in questo caso ad essere aggiornato non è una tabella, ma bensì le informazioni generali sul numero di componenti della rete.

Il file `update_topology.php` si occupa di aggiornare il grafo della topologia di rete, si differenzia leggermente dagli altri file di `update` in quanto la topologia viene creata attraverso una funzione JavaScript, in questo file vengono ricavati tutti i parametri di tale funzione utilizzando le funzioni definite in `function.php`, questi parametri vengono assegnati ad un array associativo (`$updated_data`), il quale viene convertito in formato JSON per poter essere letto dalla funzione JavaScript che richiama la funzione per creare la topologia, con i nuovi parametri ricavati.

### 3.1.4 Funzioni

Come già detto in precedenza, nel file `function.php` sono definite le funzioni che interagiscono direttamente con il controllore per ottenere le informazioni riguardo la rete analizzata. Per richiedere informazioni al controllore si utilizzano le REST API del controllore Floodlight, ad esempio per ottenere il numero di componenti all'interno della rete si specifica il seguente url:

`http://127.0.0.1:8080/wm/core/controller/summary/json`, in cui nella prima parte è specificato l'indirizzo del controllore e nella seconda parte è indicato l'uri dell'API.

In questo caso il metodo utilizzato per ottenere i dati è il metodo GET, mentre per accedere alla REST API si utilizza il metodo `curl` (client for URL) il quale consente di trasferire dati da e verso un server, `curl` consente di comunicare con un server specificando la posizione (sotto forma di URL) e i dati che si desidera inviare, supporta diversi protocolli, inclusi HTTP e HTTPS, e funziona su quasi tutte le piattaforme.

In tutte le funzioni che interagiscono direttamente con il controllore si trova lo stesso codice per le richieste, cambia solo l'url indicato in base ai dati che si vogliono ricavare, un esempio è mostrato in figura 3.1:

```
$url= 'http://127.0.0.1:8080/wm/core/controller/summary/json';  
$curl = curl_init($url);  
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);  
$response = curl_exec($curl);  
curl_close($curl);  
$arr = json_decode($response, true);
```

Figura 3.1: esempio di utilizzo di REST API per ottenere il numero di componenti della rete.

Il risultato restituito dalla chiamata di una REST API Floodlight è un file in formato JSON, pertanto, nell'ultima riga del codice in figura, si converte il file JSON in un array associativo php in modo da manipolarlo nelle righe successive della funzione per ottenere i dati necessari.

Le funzioni di interazione con il controllore contenute in function.php sono le seguenti:

- **get\_summary()**: restituisce un array contenente numero di host, link e switch totali all'interno della rete;
- **get\_switch\_id()**: restituisce in un vettore tutti gli identificatori degli switch ordinati in ordine crescente (gli id corrispondono all'indirizzo MAC dello switch visto che per creare la rete è stato specificato --mac nel comando Mininet);
- **get\_switch\_link()**: restituisce un vettore contenente i link tra switch, ogni link è identificato da entrambi gli id degli switch concatenati tramite il carattere trattino (-);
- **get\_host\_switch\_link()**: restituisce un vettore contenente i link tra host e switch, anche in questo caso i link vengono identificati come nel caso precedente;
- **get\_flow\_table\_row(\$switch\_id\_array, \$i)**: restituisce in un vettore le informazioni contenute nella flow table relative ad un singolo switch, in questo caso, a differenza delle funzioni precedenti, sono presenti due parametri, il primo è il vettore contenente gli identificatori di ciascuno switch, il secondo è l'indice di un determinato switch;
- **get\_port\_state(\$switch\_id\_array, \$i)**: restituisce un vettore contenente lo stato (attiva/inattiva) di ciascuna porta dello switch indicato come parametro, i parametri sono gli stessi della funzione precedente;
- **get\_port\_table\_raw(\$switch\_id\_array, \$i)**: restituisce in un vettore le informazioni contenute nella port table relative ad un singolo switch, i parametri sono gli stessi della funzione precedente;
- **get\_bandwidth(\$switch\_id\_array)**: restituisce un vettore contenente il throughput in ricezione e in trasmissione di ciascuna porta, per ciascuno switch della rete, gli elementi del vettore contengono il throughput, il tempo trascorso, il throughput massimo rilevato e il tempo di inattività di un'interfaccia in ricezione e trasmissione, tutti concatenati tra di loro e separati da un trattino (-). Restituisce le informazioni da inserire all'interno della throughput table, come parametro ha il vettore degli identificatori di switch. In questa funzione si agisce sulla variabile di sessione \$\_SESSION["bandwidth\_session\_array"] nella quale vengono salvati i valori dei byte letti, tempo trascorso, valore massimo di throughput osservato e tempo di inattività trascorso in modo che possano essere utilizzati per calcolare i nuovi valori aggiornati.

All'interno del file sono presenti anche due ulteriori funzioni che non interagiscono con il controllore: **bandwidth\_position(\$bandwidth\_string)** e **array\_sort(\$single\_switch, \$single\_switch\_length)**, la prima restituisce la posizione nel vettore di sessione relativa alla porta dello switch indicata nella variabile \$bandwidth\_string, la seconda ha come parametri un vettore relativo ad uno switch contenente informazioni relative a ciascuna porta e la lunghezza di tale vettore, il risultato restituito è lo stesso vettore ordinato in ordine crescente in base ai numeri delle porte dato che il file restituito dal controllore restituisce informazioni non ordinate in base alle porte, ma in ordine casuale.

## 3.2 Frontend

Una volta ricavati ed elaborati i dati, questi ultimi sono stati presentati all'utente tramite la fase di frontend, per prima cosa sono stati definiti gli elementi HTML contenenti quei dati e successivamente sono stati definiti gli stili nel file style.css in modo da rendere migliore la visualizzazione dal punto di vista dell'utente. Una volta definito ciò, è stata implementata la gestione dell'interazione tra l'utente e l'interfaccia tramite JavaScript e AJAX. In particolare, è stata creata la funzione che ricevendo le informazioni ricavate attraverso le funzioni PHP già descritte, disegna la topologia di rete, e consente di visualizzare statistiche semplicemente ponendo il mouse sopra i link e gli switch, oltre a questo sono stati definiti i controlli per evitare che l'utente inserisca dei valori non accettabili come range di throughput da monitorare. Tutte le informazioni variabili nel tempo devono essere aggiornate, per questo successivamente sono state definite delle funzioni che tramite AJAX, richiamano in modo asincrono i file di update definiti in PHP e aggiornano le sezioni con i dati ottenuti da questi file.

### 3.2.1 Creazione topologia

In script.js è definita la funzione che si occupa di creare il grafo rappresentante la topologia di rete, la definizione della funzione è la seguente in figura:

```
create_network(switch_id_array, switch_link_array, host_switch_link, host_id_array,  
             host_switch_link_id, switch_statistics, switch_link_id, switch_flow_array, bandwidth_array)
```

Figura 3.2.

I cui parametri hanno il seguente significato:

- **switch\_id\_array**: vettore degli identificatori di ciascuno switch;
- **switch\_link\_array**: vettore contenente i link tra switch;
- **host\_switch\_link**: vettore contenente i link tra host e switch;
- **host\_id\_array**: vettore contenente gli identificatori di ciascun host;
- **host\_switch\_link\_id**: vettore contenente gli identificatori dei link tra host e switch;
- **switch\_statistics**: vettore contenente statistiche riguardanti i link tra gli switch;
- **switch\_link\_id**: vettore contenente gli identificatori dei link tra switch;
- **switch\_flow\_array**: vettore contenente le informazioni contenute nella flow table di ciascuno switch;
- **bandwidth\_array**: vettore contenente i valori di throughput calcolati per ciascuna porta di ogni switch.

Questa funzione si occupa di creare i nodi del grafo, i quali sono costituiti da host e switch, ai nodi host vengono assegnati gli identificatori contenuti nel rispettivo vettore, lo stesso per i nodi switch, una volta creati i nodi del grafo vengono creati i link tra di essi, a ciascun link sono assegnati gli identificatori dei due nodi che il link stesso collega e l'identificatore del link stesso, se il link è tra host e switch l'identificatore viene assegnato dal vettore switch\_link\_id, altrimenti dal vettore degli identificatori dei link tra host e switch. Per rappresentare la topologia è stata utilizzata una libreria JavaScript esterna, inclusa in index.php, chiamata vis-network, tale libreria funziona su qualsiasi

browser moderno e consente di visualizzare reti con nodi e link, supporta colori, forme, immagini, stili personalizzati, nel progetto i nodi sono stati rappresentati con delle icone raffiguranti i componenti della rete (switch e host) e i link sono stati rappresentati con colori diversi in base al throughput corrente.

Questa funzione richiama altre due funzioni:

- **get\_switch\_position(switch\_statistics, hub, port)**: restituisce la posizione della porta dello switch coinvolta nel link selezionato dall'ingegnere in modo che vengano visualizzate correttamente le informazioni su quel determinato link quando viene puntato con il mouse;
- **updateEdgeColors(bandwidth\_array, edges, host\_switch\_link\_id, switch\_link\_id)**: aggiorna i colori dei link della rete in base ai valori di throughput su tali link.

All'interno della funzione create\_topology viene gestito anche l'evento in cui l'utente passa con il mouse sopra un link e uno switch, nel primo caso vengono mostrate delle informazioni relative al link indicato dal mouse, queste informazioni non vengono più visualizzate non appena il mouse è rimosso dal link, nel secondo caso, quando il mouse indica uno switch, viene mostrata la relativa flow table e allo stesso modo non appena il mouse viene rimosso dallo switch, queste informazioni non vengono più visualizzate.

### 3.2.2 Aggiornamento asincrono

In autoUpdate.js sono definite diverse funzioni che si occupano di aggiornare le parti dinamiche dell'interfaccia, l'aggiornamento avviene tramite richieste asincrone utilizzando AJAX, ogni 10 secondi ciascuna di queste funzioni viene richiamata con la funzione setInterval, ciascuna funzione richiama a sua volta le pagine di update in PHP, ricevono i dati aggiornati e li inseriscono nelle sezioni dell'interfaccia interessate, in questo modo l'aggiornamento avviene senza dover effettuare il refresh delle pagine.

Le funzioni contenute nel file sono le seguenti:

- **updateFlowTable()**: aggiorna la flow table;
- **updatePortTable()**: aggiorna la port table;
- **updateBandwidthTable()**: aggiorna la throughput table;
- **updateSummary()**: aggiorna il numero di host, switch e link nella rete;
- **updateTopology()**: aggiorna la topologia.

Un esempio di queste funzioni è riportato in figura 3.3:

```
function updateFlowTable() {  
    var xhr = new XMLHttpRequest();  
    xhr.onreadystatechange = function() {  
        if(xhr.readyState == 4 && xhr.status == 200){  
            document.querySelector('#flow_table_container').innerHTML = xhr.responseText;  
        }  
    };  
    xhr.open('GET', 'update_flow_table.php', true);  
    xhr.send();  
}  
  
setInterval(updateFlowTable, 10000);
```

Figura 3.3: funzione di aggiornamento della flow table.

Tutte le funzioni hanno questa struttura, updateTopology() si differenzia dalle altre in quanto non assegna direttamente i dati alla sezione interessata, ma richiama la funzione create\_network i cui parametri vengono convertiti in oggetti JavaScript dato che il risultato restituito dalla funzione update\_topology.php è in formato JSON.

All'interno del controllo, il codice eseguito nella funzione updateTopology() è il seguente:

```
var updatedData = JSON.parse(xhr.responseText);  
  
create_network(updatedData.switch_id_array, updatedData.switch_link_array,  
    updatedData.host_switch_link, updatedData.host_id_array,  
    updatedData.host_switch_link_id, updatedData.switch_statistics,  
    updatedData.switch_link_id, updatedData.switch_flow_array, updatedData.bandwidth_array);
```

Figura 3.4

# Interfaccia web

L'interfaccia web presenta un totale di 4 pagine, la pagina principale (home) è dedicata alla visualizzazione della topologia e all'interazione con l'ingegnere di rete che dovrà utilizzarla, mentre le altre tre pagine contengono ciascuna una delle tre tabelle descritte in precedenza.

## 4.1 Topology

È la pagina principale dell'interfaccia web, all'interno di questa pagina l'ingegnere di rete può visualizzare in maniera diretta e immediata lo stato della rete, in particolare può analizzare il traffico e il throughput che si trovano sui link della rete osservata in tempo reale e visualizzare eventuali cambiamenti di topologia e di capacità di trasmissione.

Nella parte alta della pagina è visualizzato in sintesi il numero di componenti che fanno parte della rete, in particolare il numero di switch, il numero di host ed il numero di link, come illustrato in figura 4.1:

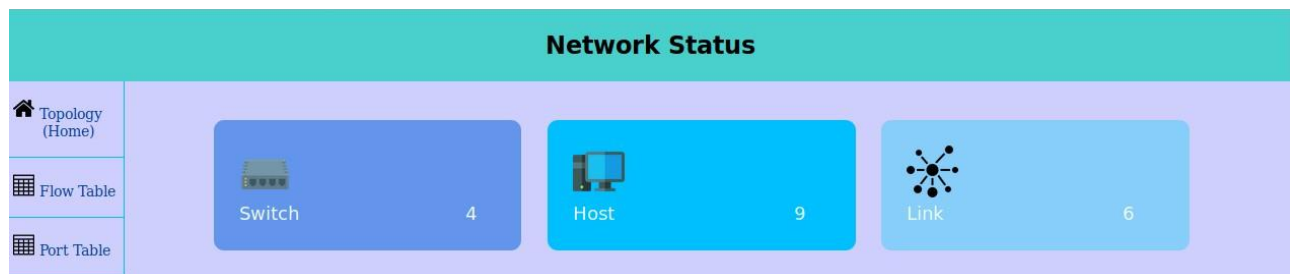


Figura 4.1.

Appena sotto questi elementi è presente una form costituita da due campi di input e da un bottone, l'utente può inserire un range di throughput da monitorare in bps, ovvero può inserire il limite inferiore nel campo "min" ed il limite superiore nel campo "max".

Premendo il bottone "submit", dopo aver inserito i valori interessati, l'utente può visualizzare quali link hanno un throughput minore, maggiore o compreso tra i due valori, in figura 4.2 è raffigurata questa sezione della pagina. Se l'utente preme il bottone e non ha inserito i valori, oppure ha inserito dei valori scorretti, ovvero valori negativi, valori contenenti dei caratteri diversi da numeri o se il minimo inserito è maggiore del massimo inserito, in questi casi il sistema mostra un messaggio per invitare l'utente ad inserire i valori correttamente e mentre i valori inseriti nella form non sono corretti, i link della rete assumono tutti lo stesso colore blu e non variano in base al throughput, le altre informazioni invece continuano ad essere aggiornate.



Throughput Monitoring

Choose throughput range:

min: 1700 max: 1100000 Submit

Figura 4.2.

Una volta inseriti i valori corretti, l'ingegnere di rete può subito notare se in un link il throughput è minore, maggiore o compreso tra i valori scelti, questo viene mostrato direttamente nel grafo rappresentante la topologia della rete attraverso dei colori, i link di colore rosso hanno un throughput maggiore rispetto al limite superiore inserito dall'utente, i link di colore giallo hanno un throughput compreso tra il limite inferiore ed il limite superiore, estremi inclusi, mentre i link verdi hanno un throughput minore del limite inferiore inserito.

Con questa scelta si vuole conferire all'ingegnere che dovrà monitorare la rete, una visione semplice ed immediata su come si sta comportando quella rete, senza scendere nel dettaglio, questo per rendersi conto della situazione senza dover controllare ad esempio una tabella più dettagliata, la quale richiederebbe maggiore tempo per essere analizzata.

La topologia è visualizzata sotto il menu di selezione nella pagina principale, nella seguente figura è mostrata nel caso in cui i valori inseriti dall'utente non siano corretti.

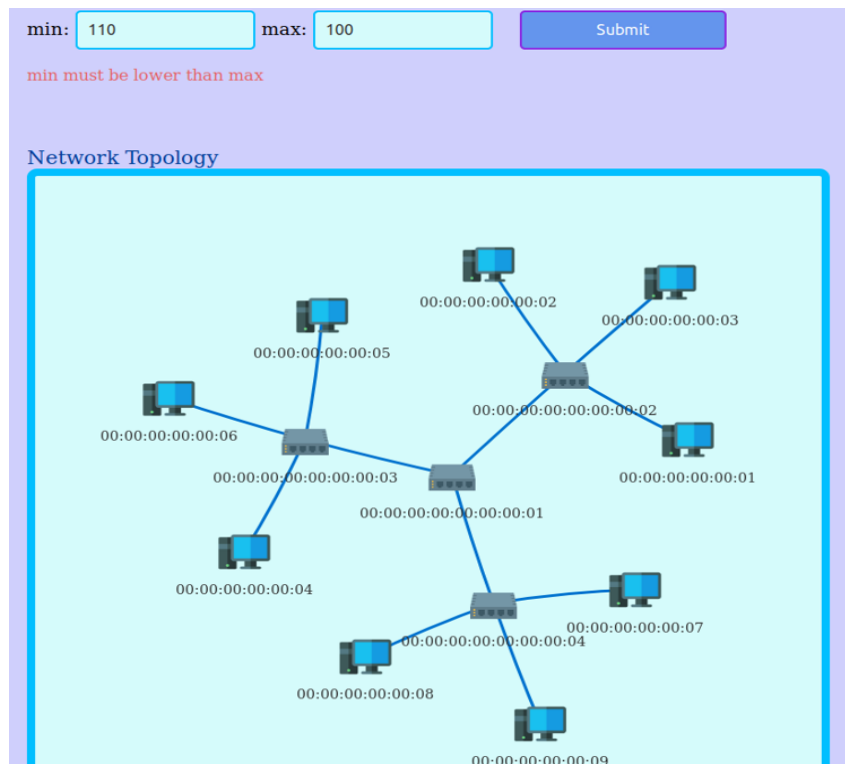


Figura 4.3: l'utente ha inserito un valore maggiore nel campo min rispetto al valore del campo max.

Nel caso in cui i valori inseriti siano corretti, la topologia è rappresentata come in figura 4.3, nella quale sono presenti colori diversi sui link in base al throughput che essi hanno rispetto al valore selezionato in quell'istante.

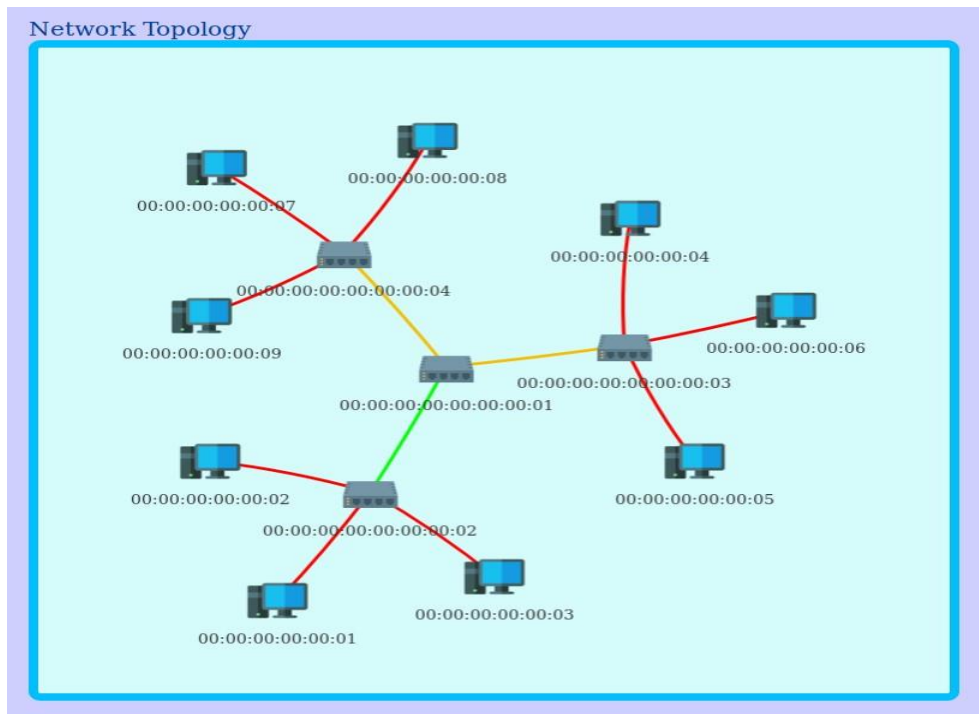


Figura 4.4.

Da notare come nel grafo i vari dispositivi siano accompagnati da un identificatore univoco, il quale corrisponde al loro indirizzo MAC, questa informazione e il colore dei link sono le uniche informazioni che si ottengono guardando il grafo, se l'utente vuole una visione più dettagliata, ma comunque rapida, può puntare con il mouse link e switch, in questo caso vengono visualizzate alcune informazioni riguardanti l'elemento puntato. Quando l'utente punta con il mouse su uno switch, si apre una finestra che mostra l'id di tale switch, il numero di flussi attivi, i pacchetti e i byte transitati da quello switch fino a quell'istante, l'istante di tempo è espresso in secondi ed è misurato dall'avvio del controllore SDN una volta creata la rete con Mininet, in figura è rappresentato un esempio di ciò che viene visualizzato in questo caso:

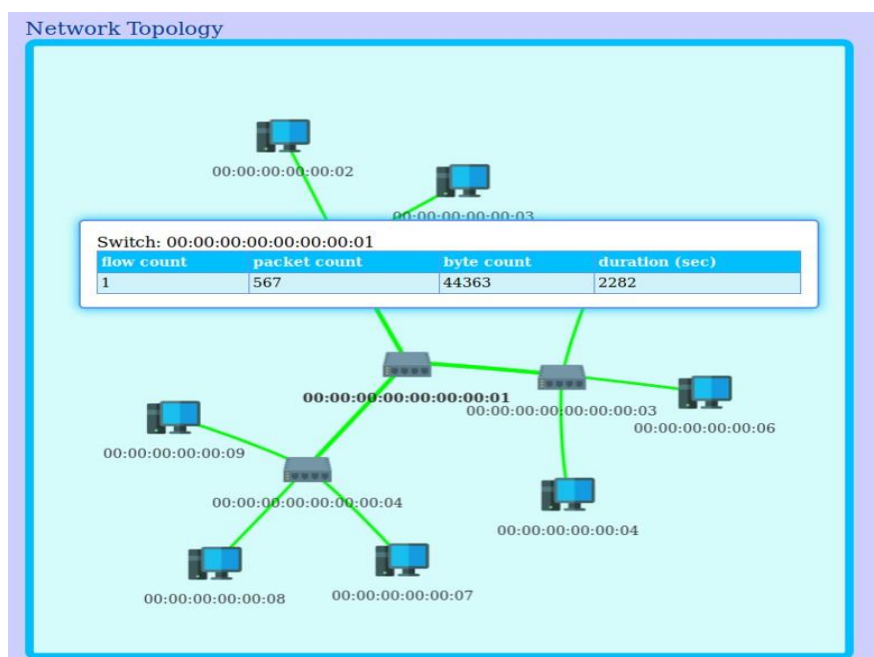


Figura 4.5: vengono mostrate informazioni riguardo lo switch selezionato (switch 1).

Puntando un link tra host e switch vengono visualizzati gli id dei due dispositivi, la porta dello switch a cui è connesso quel link, il throughput corrente su tale link e alcune informazioni in formato tabellare tra cui: i pacchetti ricevuti e trasmessi, il totale di byte ricevuti e trasmessi dallo switch, il numero di eventuali pacchetti scartati in entrambe le direzioni, il numero di eventuali pacchetti con errori in entrambe le direzioni ed il tempo trascorso in secondi.

Un esempio è riportato in figura 4.6:

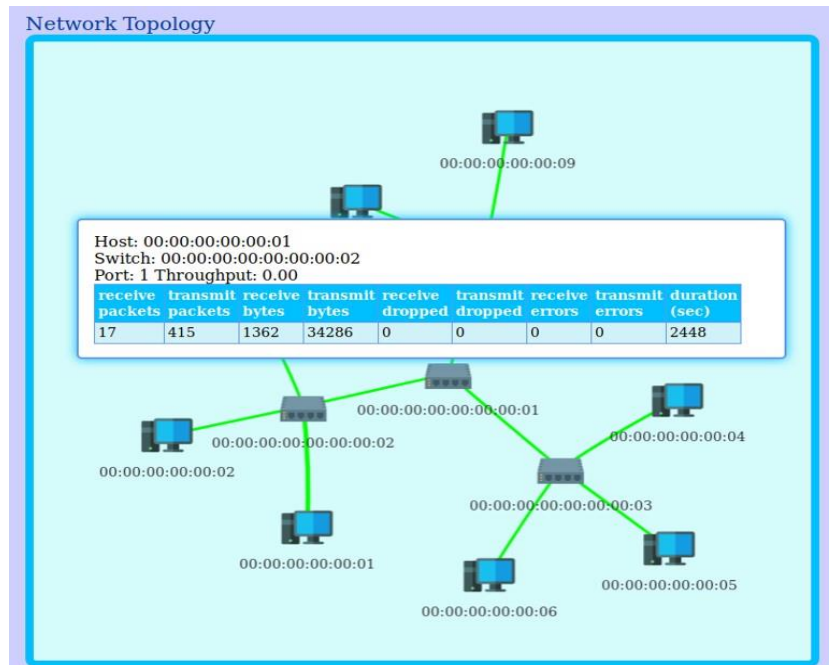


Figura 4. 6: visualizzazione informazioni del link tra host 1 e switch 2.

Puntando un link tra due switch vengono mostrate le stesse informazioni del caso precedente, in questo caso risultano raddoppiate dato che i link tra switch sono bidirezionali, viene aggiunta anche la direzione, src ad indicare la sorgente e dst ad indicare la destinazione, un esempio è riportato in figura 4.7:

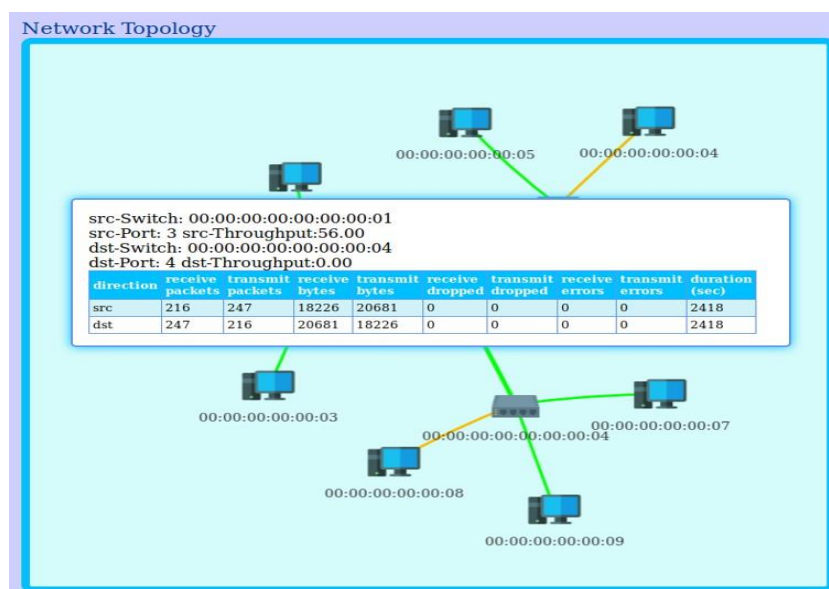


Figura 4.7: informazioni relative al link tra switch 1 e switch 4.

Sul lato sinistro della pagina è riportata una barra di navigazione verticale, nella quale è possibile scegliere tre tipi di tabelle, le tabelle visualizzate contengono le informazioni che l'utente può consultare direttamente dalla topologia con alcune informazioni aggiuntive, in questo modo, può osservare in maniera più approfondita e unificata le statistiche e le performance della rete.

## 4.2 Flow table

Contiene le informazioni visualizzate quando l'utente punta con il mouse un qualsiasi switch, nella figura seguente è riportata tale tabella.

Network Status					
<div>Topology (Home)</div> <div>Flow Table</div> <div>Port Table</div> <div>Throughput Monitoring</div>	Flow Table				
	switch id	flow count	packet count	byte count	duration (sec)
	00:00:00:00:00:00:01	1	115	9136	373
	00:00:00:00:00:00:02	1	52	4003	373
	00:00:00:00:00:00:03	1	52	4003	373
	00:00:00:00:00:00:04	1	52	4003	373

Figura 4.8.

## 4.3 Port table

Contiene le informazioni visualizzate quando l'utente punta con il mouse un link, in questo caso viene mostrato anche lo stato dell'interfaccia, ovvero se attiva o no, inoltre fornisce una visione unificata di tutte le interfacce di ciascuno switch, sono inclusi link sia tra due switch che tra host e switch, nella figura seguente è riportata tale tabella.

Network Status												
<div>Topology (Home)</div> <div>Flow Table</div> <div>Port Table</div> <div>Throughput Monitoring</div>	Port Table											
	switch id	port number	state	receive packets	transmit packets	receive bytes	transmit bytes	receive dropped	receive dropped	receive errors	transmit errors	duration (sec)
	00:00:00:00:00:00:01	local	LINK_DOWN	0	0	0	0	0	0	0	0	77
	00:00:00:00:00:00:01	1	LIVE	36	43	4293	4788	0	0	0	0	77
	00:00:00:00:00:00:01	2	LIVE	37	42	4368	4713	0	0	0	0	77
	00:00:00:00:00:00:01	3	LIVE	37	42	4376	4713	0	0	0	0	77
	00:00:00:00:00:00:02	local	LINK_DOWN	0	0	0	0	0	0	0	0	77
	00:00:00:00:00:00:02	1	LIVE	12	53	1012	5588	0	0	0	0	77
	00:00:00:00:00:00:02	2	LIVE	12	51	1016	5428	0	0	0	0	77
	00:00:00:00:00:00:02	3	LIVE	11	52	926	5498	0	0	0	0	77
	00:00:00:00:00:00:02	4	LIVE	43	36	4788	4293	0	0	0	0	77
	00:00:00:00:00:00:03	local	LINK_DOWN	0	0	0	0	0	0	0	0	77
	00:00:00:00:00:00:03	1	LIVE	13	52	1102	5498	0	0	0	0	77
	00:00:00:00:00:00:03	2	LIVE	13	52	1102	5498	0	0	0	0	77
	00:00:00:00:00:00:03	3	LIVE	12	52	1012	5498	0	0	0	0	77
	00:00:00:00:00:00:03	4	LIVE	42	37	4713	4368	0	0	0	0	77
	00:00:00:00:00:00:04	local	LINK_DOWN	0	0	0	0	0	0	0	0	77
	00:00:00:00:00:00:04	1	LIVE	11	52	926	5498	0	0	0	0	77
	00:00:00:00:00:00:04	2	LIVE	11	52	926	5498	0	0	0	0	77
	00:00:00:00:00:00:04	3	LIVE	11	52	926	5498	0	0	0	0	77
	00:00:00:00:00:00:04	4	LIVE	42	37	4713	4376	0	0	0	0	77

Figura 4.9.

## 4.4 Throughput table

Tabella che visualizza il throughput calcolato per finestre temporali di 10 secondi su ogni link, inoltre visualizza due informazioni che non vengono mostrate nella topologia, ovvero il throughput massimo rilevato e il tempo trascorso da quando il throughput di un'interfaccia ha assunto come valore zero bps. Le informazioni relative al throughput, al throughput massimo e al tempo di inattività sono mostrate sia per il traffico in ingresso che per il traffico in uscita da un'interfaccia, la tabella è mostrata nella seguente figura.

Network Status								
<div>Topology (Home)</div> <div>Flow Table</div> <div>Port Table</div> <div>Throughput Monitoring</div>	Throughput Table							
	switch id	port number	receive throughput (bps)	transmit throughput (bps)	max throughput rcv (bps)	max throughput tran (bps)	inactive time rcv (sec)	inactive time tran (sec)
	00:00:00:00:00:00:00:01	1	60.00	116.00	161.13	173.69	0	0
	00:00:00:00:00:00:00:01	2	116.00	60.00	160.75	313.60	0	0
	00:00:00:00:00:00:00:01	3	60.00	116.00	156.85	173.69	0	0
	00:00:00:00:00:00:00:01	local	0.00	0.00	0.00	0.00	259	259
	00:00:00:00:00:00:00:02	1	3,970,658,560.00	3,133,862.40	10,483,583,881.60	9,089,958.40	0	0
	00:00:00:00:00:00:00:02	2	3,133,680.00	3,970,658,742.40	9,089,790.40	10,483,584,524.80	0	0
	00:00:00:00:00:00:00:02	3	0.00	182.40	56.00	257.60	10	0
	00:00:00:00:00:00:00:02	4	116.00	60.00	173.69	161.13	0	0
	00:00:00:00:00:00:00:02	local	0.00	0.00	0.00	0.00	259	259
	00:00:00:00:00:00:00:03	1	56.00	126.40	56.00	313.60	0	0
	00:00:00:00:00:00:00:03	2	0.00	182.40	56.00	313.60	20	0
	00:00:00:00:00:00:00:03	3	0.00	182.40	46.87	257.60	89	0
	00:00:00:00:00:00:00:03	4	60.00	116.00	313.60	160.75	0	0
	00:00:00:00:00:00:00:03	local	0.00	0.00	0.00	0.00	259	259
	00:00:00:00:00:00:00:04	1	0.00	182.40	56.00	214.40	10	0
	00:00:00:00:00:00:00:04	2	0.00	182.40	56.00	215.15	10	0
	00:00:00:00:00:00:00:04	3	0.00	238.40	51.11	257.60	89	0
	00:00:00:00:00:00:00:04	4	116.00	60.00	173.69	156.85	0	0
	00:00:00:00:00:00:00:04	local	0.00	0.00	0.00	0.00	259	259

Figura 4.1.1

# Testing

Il progetto vede il suo sviluppo all'interno di una macchina virtuale con sistema operativo Ubuntu che consente di utilizzare tutti gli strumenti necessari per la creazione dell'interfaccia web-based, una volta completata la creazione è necessario effettuare alcuni test per verificare la correttezza delle informazioni raccolte dalla rete analizzata, gli strumenti utilizzati per la realizzazione dei test sono i seguenti:

- Mininet
- Iperf

## 5.1 Mininet

Mininet è una piattaforma open source di emulazione di reti virtuali. Con un singolo comando esegue una raccolta di host finali, switch, router e collegamenti su una singola macchina in pochi secondi, costituisce un ottimo modo per sviluppare, ed effettuare test con i sistemi SDN utilizzando OpenFlow.

Utilizza la virtualizzazione per far sembrare un singolo sistema come una rete reale completa, che esegue lo stesso kernel, crea switch ed esegue codice applicativo, un host Mininet si comporta proprio come una macchina reale. I programmi eseguiti possono inviare pacchetti attraverso una interfaccia Ethernet simulata, con una determinata velocità di collegamento e ritardo.

I pacchetti vengono elaborati da switch, router o middlebox Ethernet virtuali, quando due programmi, come un client e un server iperf, comunicano tramite Mininet, le prestazioni misurate cercano di imitare quelle di due macchine reali.

Utilizzare Mininet comporta i seguenti vantaggi:

- È veloce, l'avvio di una rete semplice richiede solo pochi secondi.
- È possibile creare topologie personalizzate: un singolo switch, topologie più grandi simili a Internet, un data center o qualsiasi altra cosa.
- È possibile eseguire programmi reali, si può utilizzare tutto ciò che è su Linux, dai server web agli strumenti di monitoraggio delle finestre TCP.
- È possibile personalizzare l'inoltro dei pacchetti, gli switch di Mininet sono programmabili utilizzando il protocollo OpenFlow. I progetti di reti SDN eseguiti in Mininet possono essere facilmente trasferiti agli switch hardware OpenFlow.
- È possibile eseguire Mininet ovunque, su un laptop, su un server, in una VM, su una macchina Linux nativa o in un cloud.
- Le reti di switch OpenFlow possono essere connesse ad un controller reale.
- I risultati possono essere duplicati e condivisi.
- È facile da usare, si può creare ed eseguire esperimenti Mininet scrivendo script Python semplici.

- Mininet è un progetto open source, il suo codice sorgente è disponibile ed è possibile anche modificare questa documentazione per correggere eventuali errori o aggiungere chiarimenti o informazioni aggiuntive.

Il comando mostrato in figura 5.1 è eseguito sul terminale e crea la topologia di rete utilizzata per effettuare i test, la rete creata ha le seguenti caratteristiche:

- **--topo tree**: rete con topologia ad albero, in cui tutti gli switch sono direttamente collegati ad uno switch radice;
- **Depth=2**: rete con due livelli di profondità, ovvero un livello di switch direttamente connessi ad uno switch radice;
- **Fanout=3**: indica il numero di host connessi agli switch di livello inferiore;
- **--ipbase=10.0.0.0**: specifica l'indirizzo ip della rete;
- **-- mac**: di default agli host e switch in Mininet vengono assegnati indirizzi MAC casuali, il che può rendere difficile identificarli, utilizzando questo comando, l'indirizzo MAC di un host o switch può essere impostato in modo che corrisponda al suo ID, rendendo più semplice individuare il nodo utilizzando il suo indirizzo MAC;
- **-- switch ovsk**: viene utilizzato per specificare il tipo di switch da utilizzare nella topologia. In questo caso, ovsk sta per Open vSwitch kernel space;
- **--controller remote**: indica a Mininet di utilizzare un controller remoto piuttosto del controller locale di default;
- **Ip=27.0.0.1**: indica l'indirizzo del controllore reale;
- **Port=6653**: indica la porta del controllore reale;
- **Protocols=OpenFlow13**: indica il protocollo utilizzato;
- **--link=tc**: viene utilizzato per specificare il tipo di link come Traffic Control (TC), è utilizzato per impostare la larghezza di banda, il ritardo, la perdita e la lunghezza massima della coda di un collegamento.

```
student@osboxes:~$ sudo mn --topo tree,depth=2,fanout=3 --ipbase=10.0.0.0 --mac --switch ovsk
--controller remote,ip=127.0.0.1,port=6653,protocols=OpenFlow13 --link=tc
```

Figura 5.1: creazione di una rete con Mininet.

## 5.2 Iperf

Eseguendo il comando **xterm** in Mininet, si apre un terminale corrispondente al terminale dell'host specificato nel comando, una volta aperto tale terminale è possibile effettuare le normali operazioni eseguibili su un terminale di un generico host.

In figura 5.2 è riportata l'esecuzione del comando per gli host h1 ed h9:

```
mininet> h1 xterm &
mininet> h9 xterm &
```

Figura 5.2.



Una volta aperti i terminali è possibile eseguire dei comandi in essi, nel progetto questa procedura è risultata utile per generare traffico nella rete inviando un certo numero di byte tra due host lontani, come ad esempio h1 ad h9 e tra due host collegati allo stesso switch, come h1 ed h2, verificando che il throughput e i byte trasmessi con tale operazione venissero visualizzati correttamente dall'interfaccia. Questo perché, se non si genera traffico, nella rete non circolano molti pacchetti e risulta difficile verificare la correttezza dell'applicazione, il throughput misurato è nullo o assume valori non nulli, ma comunque bassi.

Per effettuare questi test per prima cosa occorre ricavare l'indirizzo ip degli host tramite il comando **ifconfig**, il quale fornisce in output l'indirizzo e altre informazioni sull'host mostrate in figura 5.3:

```
root@osboxes:~# ifconfig
h9-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.9 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::200:ff:fe00:9 prefixlen 64 scopeid 0x20<link>
    ether 00:00:00:00:00:09 txqueuelen 1000 (Ethernet)
    RX packets 30 bytes 4088 (4.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 13 bytes 1066 (1.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 5.3: comando ifconfig in h1 e relativo output.

Una volta ricavati gli indirizzi è possibile generare traffico attraverso il comando **iperf** eseguito negli host interessati.

Iperf è un software open source scritto in C e funziona su varie piattaforme, tra cui Windows, Linux, macOS e Unix, è uno strumento di test di rete in grado di creare flussi di dati TCP e UDP e misurare il rendimento di una rete che li trasporta. Consente all'utente di impostare vari parametri che possono essere utilizzati per testare o, in alternativa, per ottimizzare o mettere a punto una rete.

È uno strumento multiplatforma in grado di produrre misurazioni delle prestazioni standardizzate per qualsiasi rete.

Tra i parametri può misurare la larghezza di banda massima ottenibile sulle reti IP, può essere utilizzato per ottimizzare vari parametri relativi a tempistiche, protocolli (TCP, UDP, SCTP con IPv4 e IPv6) e buffer. Per ogni test l'output tipico contiene un rapporto della quantità di dati trasferiti e del throughput/bitrate misurato, la perdita e altri parametri.

Iperf ha funzionalità client e server e può creare flussi di dati per misurare il throughput tra le due estremità in una o entrambe le direzioni. Ad esempio, eseguendo il comando in figura 5.4, l'host su cui viene eseguito (h9 in questo caso) agisce da server e si mette in ascolto nella porta TCP 5001.



```

root@osboxes:~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 85,3 KByte (default)

```

Figura 5.4.

Una volta assegnato il server, un altro host può essere configurato come client per inviare dei byte al server, in figura 5.5 è riportato un esempio di comando iperf per creare un client con il relativo output, in questo caso h1 funziona da client, e invia un certo numero di byte (1.25 GB) al server (10.0.0.9) ad una bandwidth di 1GB/sec (-b 1G) in un intervallo di 10 secondi.

```

root@osboxes:~# iperf -c 10.0.0.9 -b 1G
-----
Client connecting to 10.0.0.9, TCP port 5001
TCP window size: 298 KByte (default)
-----
[  3] local 10.0.0.1 port 59396 connected with 10.0.0.9 port 5001
[ ID] Interval      Transfer    Bandwidth
[  3]  0.0-10.0 sec  1.25 GBytes  1.07 Gbits/sec

```

Figura 5.5.

Il server (h9) una volta ricevuti i byte ne mostra il numero, la bandwidth a cui sono stati trasmessi e il tempo di trasmissione impiegato, come illustrato in figura 5.6:

```

-----
[  4] local 10.0.0.9 port 5001 connected with 10.0.0.1 port 59396
[ ID] Interval      Transfer    Bandwidth
[  4]  0.0-10.0 sec  1.25 GBytes  1.07 Gbits/sec

```

Figura 5. 6.

### 5.2.1 Test con iperf

Un primo test che si può effettuare è quello di inviare una certa quantità di byte ad una bandwidth stabilita tra due specifici host e osservare se nel grafo di rete i link vengono colorati coerentemente con il traffico reale nella rete. In figura 5.7 è stato utilizzato il comando iperf per inviare 1 Mbyte con una banda di 1 Mbit al secondo tra due host collegati allo stesso switch, in questo caso h1 ed h2 collegati a s2, si può notare come il throughput sia elevato e compreso tra i valori inseriti nei campi della form solamente nei link tra h1 e s2 e tra h2 e s2, essendo questi link di colore giallo e tutti gli altri di colore verde.

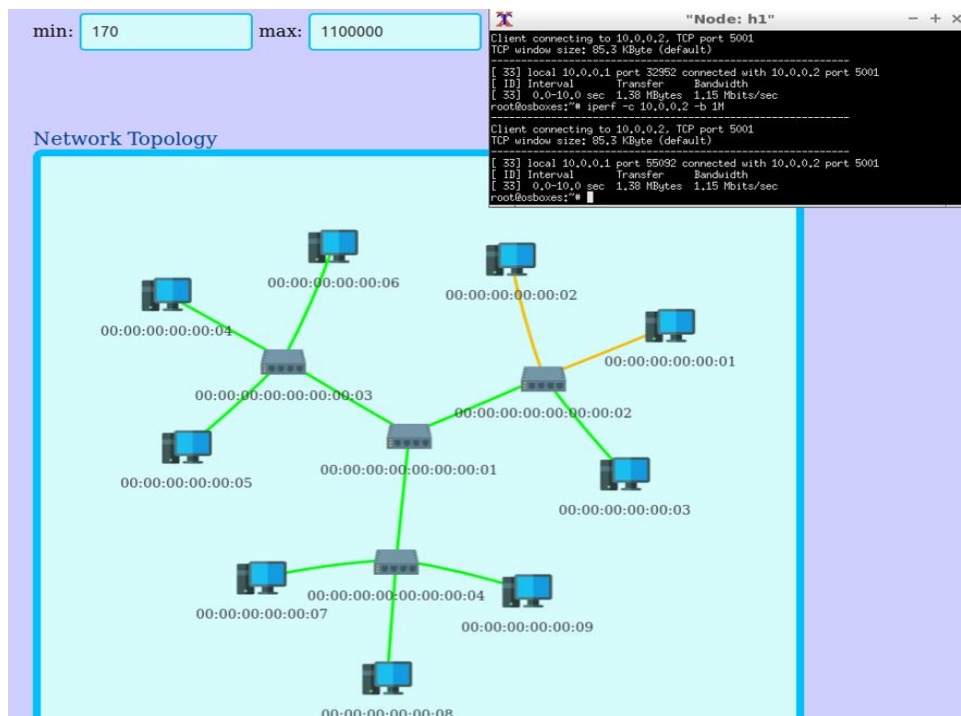


Figura 5.7.

Lo stesso test è stato ripetuto tra due host lontani, come h1 ed h9, inviando da h1 gli stessi byte alla stessa banda indicati nel test precedente, in questo caso si evidenzia in giallo solamente il percorso tra h1 ed h9, come mostrato in figura.

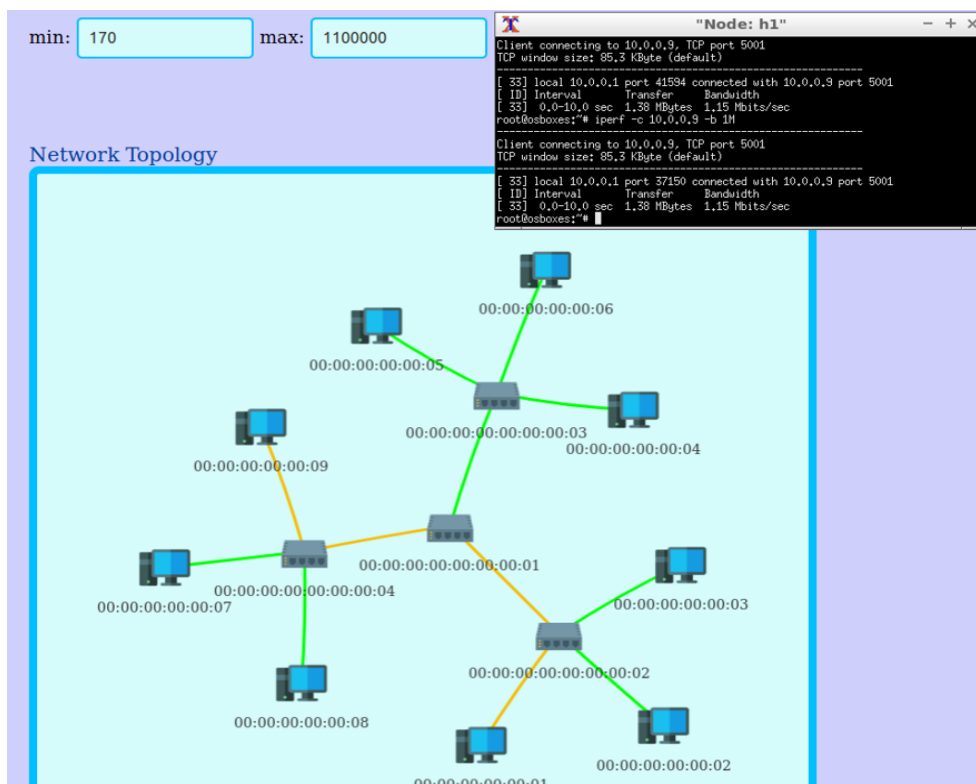


Figura 5.8.

Una volta verificata la correttezza delle informazioni nella topologia, rimane da verificare se il contenuto delle tabelle è corretto, per la port table, è possibile inviare una certa quantità di byte da un host all'altro e guardando la tabella, i valori nei campi receive bytes e transmit bytes corrispondono ai byte effettivamente trasmessi.

Inviando dei byte da h1 ad h9 si osserva che il throughput massimo di 11 miliardi di bit al secondo è riportato in tutte le interfacce degli switch che si trovano sul percorso da h1 ad h9, in particolare in figura 5.9 si nota che il throughput massimo durante il trasferimento di questi byte è lo stesso per l'interfaccia 1 di s1 in ricezione, per l'interfaccia 3 di s1 in trasmissione, l'interfaccia 1 di s2 in ricezione (interfaccia tra s2 e h1), l'interfaccia 4 di s2 in trasmissione, l'interfaccia 4 di s4 in ricezione e l'interfaccia 3 di s4 in trasmissione (interfaccia tra s4 e h9). Inoltre, i valori di throughput massimo per le interfacce di s3 sono bassi, questo perché s3 non si trova nel percorso tra h1 ed h9.

Network Status									
Topology (Home)	Throughput Table								
Flow Table	switch id	port number	receive throughput (bps)	transmit throughput (bps)	max throughput recv (bps)	max throughput tran (bps)	inactive time recv (sec)	inactive time tran (sec)	duration (sec)
Port Table	00:00:00:00:00:00:01	1	2,398,796,343.20	1,635,860.00	11,319,646,797.60	10,876,829.60	0	0	499
Throughput Monitoring	00:00:00:00:00:00:01	2	60.00	224.80	160.75	432.80	0	0	499
	00:00:00:00:00:00:01	3	1,635,860.00	2,398,796,343.20	10,876,829.60	11,319,646,797.60	0	0	499
	00:00:00:00:00:00:01	local	0.00	0.00	0.00	0.00	499	499	499
	00:00:00:00:00:00:02	1	2,397,992,614.40	1,635,873.60	11,320,449,614.40	10,876,948.80	0	0	499
	00:00:00:00:00:00:02	2	0.00	238.40	9,089,790.40	10,483,584,524.80	200	0	499
	00:00:00:00:00:00:02	3	0.00	294.40	56.00	862.40	250	0	499
	00:00:00:00:00:00:02	4	1,635,807.20	2,397,992,674.40	10,876,882.40	11,320,450,466.40	0	0	499
	00:00:00:00:00:00:02	local	0.00	0.00	0.00	0.00	499	499	499
	00:00:00:00:00:00:03	1	0.00	291.20	56.00	499.20	240	0	499
	00:00:00:00:00:00:03	2	0.00	291.20	56.00	499.20	20	0	499
	00:00:00:00:00:00:03	3	0.00	291.20	56.00	499.20	220	0	499
	00:00:00:00:00:00:03	4	224.80	60.00	432.80	160.75	0	0	499
	00:00:00:00:00:00:03	local	0.00	0.00	0.00	0.00	499	499	499
	00:00:00:00:00:00:04	1	0.00	238.40	56.00	4,937.60	20	0	499
	00:00:00:00:00:00:04	2	56.00	182.40	56.00	4,937.60	0	0	499
	00:00:00:00:00:00:04	3	1,635,374.40	2,394,235,568.00	10,877,139.20	11,324,207,761.60	0	0	499
	00:00:00:00:00:00:04	4	2,394,235,445.60	1,635,490.40	11,324,207,695.20	10,877,199.20	0	0	499
	00:00:00:00:00:00:04	local	0.00	0.00	0.00	0.00	499	499	499

Figura 5.9

Un risultato corretto si ottiene anche inviando byte tra h1 ed h2, i quali sono collegati allo stesso switch, in figura 4.1.1 vista in precedenza, si nota come il throughput sia lo stesso tra l'interfaccia 1 di s2 in ricezione (collegata ad h1) e l'interfaccia 2 di s2 in trasmissione (collegata ad h2).

È inoltre possibile testare se il throughput misurato corrisponde al throughput reale, inviando dei byte ad esempio tra h1 ed h9, si vede in figura 5.1.1, come il throughput indicato nel terminale di h1 sia quasi uguale al throughput misurato nella colonna "transmit throughput" relativa all'interfaccia 3 di s4, il valore non è esattamente uguale dato che potrebbe esserci un leggero ritardo tra l'istante in cui iperf misura il throughput e l'istante in cui vengono aggiornate le informazioni.

switch id	port number	receive throughput (bps)	transmit throughput (bps)	max throughput recv (bps)	max throughput tran (bps)	inactive time recv (sec)	inactive time tran (sec)	duration (sec)
00:00:00:00:00:00:01	1	13,117,177,866.40	13,224,496.80	13,117,177,866.40	13,224,496.80	0	0	1193
00:00:00	"Node: h1" - + x			6	"Node: h9" - + x			
00:00:00	ether 00:00:00:00:00:01 txqueuelen 1000 (Ethernet)			1	ether 00:00:00:00:00:09 txqueuelen 1000 (Ethernet)			
00:00:00	RX packets 188 bytes 15328 (15.3 KB)			1	RX packets 188 bytes 15328 (15.3 KB)			
00:00:00	RX errors 0 dropped 130 overruns 0 frame 0			0	RX errors 0 dropped 130 overruns 0 frame 0			
00:00:00	TX packets 15 bytes 1226 (1.2 KB)			1	TX packets 16 bytes 1292 (1.2 KB)			
00:00:00	TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0			1	TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0			
00:00:00	lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536			1	lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536			
00:00:00	inet 127.0.0.1 netmask 255.0.0.0			2	inet 127.0.0.1 netmask 255.0.0.0			
00:00:00	inet6 ::1 prefixlen 128 scopeid 0x10<host>			1	inet6 ::1 prefixlen 128 scopeid 0x10<host>			
00:00:00	loop txqueuelen 1000 (Local Loopback)			1	loop txqueuelen 1000 (Local Loopback)			
00:00:00	RX packets 0 bytes 0 (0.0 B)			1	RX packets 0 bytes 0 (0.0 B)			
00:00:00	RX errors 0 dropped 0 overruns 0 frame 0			0	RX errors 0 dropped 0 overruns 0 frame 0			
00:00:00	TX packets 0 bytes 0 (0.0 B)			1	TX packets 0 bytes 0 (0.0 B)			
00:00:00	TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0			1	TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0			
00:00:00	root@osboxes:~# iperf -s 10.0.0.9 -f b			9	root@osboxes:~# iperf -s			
00:00:00	Client connecting to 10.0.0.9, TCP port 5001			1	Server listening on TCP port 5001			
00:00:00	TCP window size: 87380 Byte (default)			6	TCP window size: 85.3 KByte (default)			
00:00:00	[ 33] local 10.0.0.1 port 54148 connected with 10.0.0.9 port 5001			0	[ 34] local 10.0.0.9 port 5001 connected with 10.0.0.1 port 54148			
00:00:00	[ ID] Interval Transfer Bandwidth			1	[ ID] Interval Transfer Bandwidth			
00:00:00	[ 33] 0.0-10.0 sec 17394696192 Bytes 13914899796 bits/sec			1	[ 34] 0.0- 9.9 sec 16.2 GBytes 14.0 Gbits/sec			
00:00:00	root@osboxes:~#			1	root@osboxes:~#			
00:00:00:00:00:00:00:04	2	0.00	160.00	114.49	1,264.00	272	0	1193
00:00:00:00:00:00:00:04	3	13,225,070.40	13,126,975,988.80	13,225,070.40	13,126,975,988.80	0	0	1193
00:00:00:00:00:00:00:04	4	13,126,293,991.20	13,225,130.40	13,126,293,991.20	13,225,130.40	0	0	1193
00:00:00:00:00:00:00:04	local	0.00	0.00	0.00	0.00	1193	1193	1193

Figura 5.1.1.

## 5.3 Deployment

Nel caso in cui si voglia utilizzare il progetto occorre effettuare diversi passaggi in sequenza, per prima cosa occorre utilizzare una macchina virtuale Ubuntu su cui eseguire l'applicazione, all'interno di essa occorre disporre di un server per poter eseguire l'applicazione, durante lo sviluppo è stato utilizzato Xampp, uno strumento disponibile in rete, il quale dispone di alcuni servizi, tra i quali il server Apache. Una volta scaricato viene salvato di default nella cartella **opt/lampp**, per attivare il server da terminale occorre entrare in tale cartella ed eseguire il comando **sudo ./xampp start**, questo comando attiva anche altri servizi che non interessano l'applicazione.

Una volta attivato il server, occorre inserire la cartella con i file del progetto all'interno della cartella **htdocs** di Xampp, successivamente è possibile creare una rete da terminale utilizzando Mininet come visto in precedenza.

Dopo aver creato la rete occorre eseguire il controllore, in questo caso l'editor utilizzato per l'esecuzione è Eclipse Oxygen, nel quale è possibile avviare il controllore SDN, una volta attivato il controllore l'ultimo step consiste nell'aprire la pagina principale dell'interfaccia inserendo nel browser il seguente url: **http://localhost/nome\_cartella/index.php**, dove nome\_cartella indica una cartella contenente l'elenco dei file del progetto precedentemente elencati.

# Bibliografia

## Articoli consultati

- Floodlight:  
<https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/what-is-sdn-controller/openflow-controller/what-is-floodlight-controller>
- REST API:  
<https://www.redhat.com/it/topics/api/what-is-a-rest-api>
- SDN attivo e proattivo:  
<https://www.ionos.it/digitalguide/server/know-how/software-defined-networking>

## Manuali consultati

- Mininet:  
<https://github-wiki-see.page/m/mininet/mininet/wiki/Introduction-to-Mininet>
- iPerf:  
<https://iperf.fr/iperf-doc.php>