ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
SCUOLA DI INGEGNERIA E ARCHITETTURA
DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA
in
Laboratorio di Amministrazione di Sistemi T

ANALISI E SVILUPPO DI FUNZIONI DI RETE VIRTUALI INTEGRATE IN ARCHITETTURE SDN PER APPLICAZIONI DI CONTROLLO REMOTO DI MACCHINE AUTOMATICHE

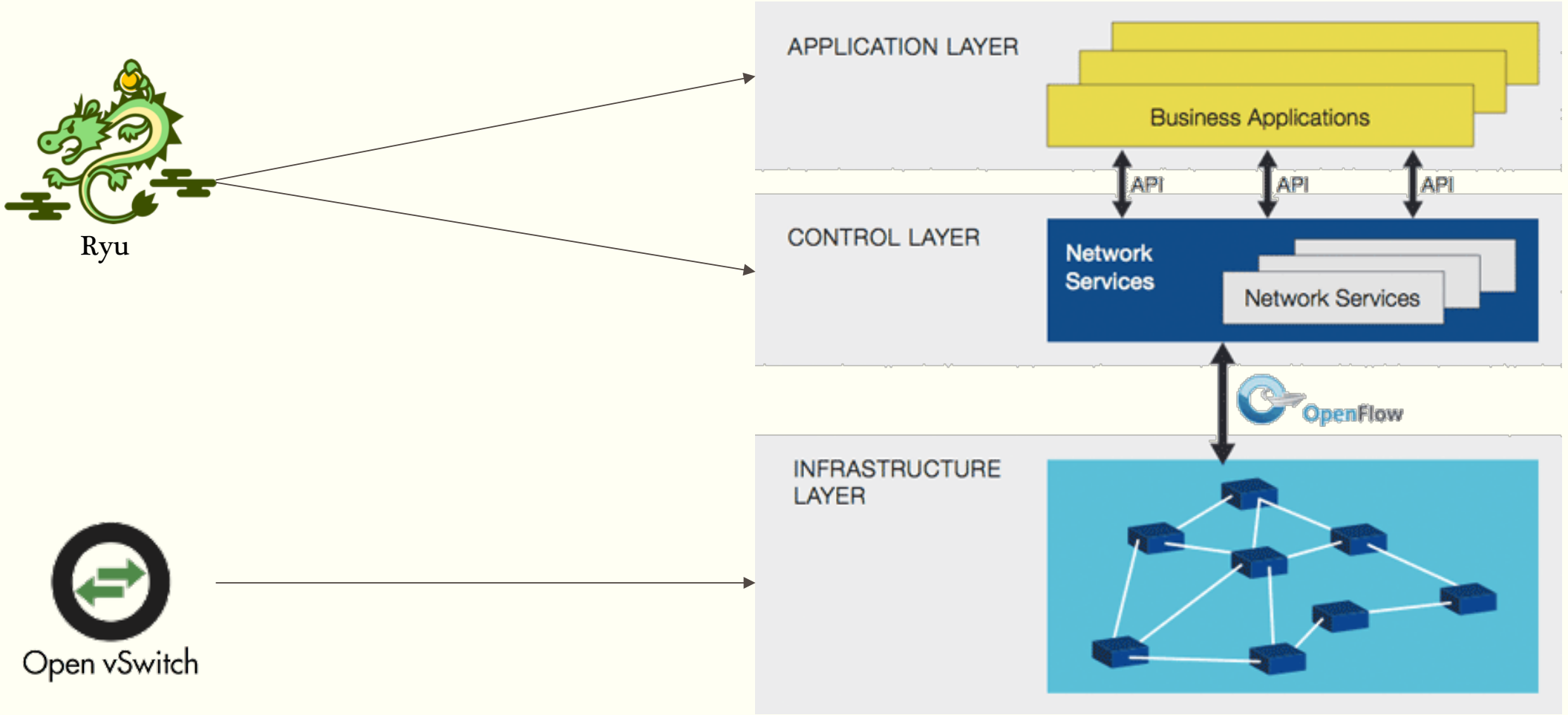
CANDIDATO:
DAVIDE DI DONATO

RELATORE:
CHIAR.MO PROF. MARCO PRANDINI

CORRELATORI:
CHIAR.MO PROF. FRANCO CALLEGATI, DOTT. ING. ANDREA MELIS

Anno Accademico 2015/2016
Sessione III

Contesto


Ryu
Open vSwitch
APPLICATION LAYER
Business Applications
API
API
API
CONTROL LAYER
Network Services
Network Services
OpenFlow
INFRASTRUCTURE LAYER

Obiettivi

- Creare una controller application in grado di:
  - Redirezionare il traffico di interesse in ingresso verso zero o più computer di elaborazione, connessi allo switch stesso, secondo le richieste dell'utente
  - Visualizzare le redirezioni impostate e modificarle, in base a quanto richiesto dall'utente
- Verificare il funzionamento della controller application, attraverso dei test, su una rete SDN virtuale
- Integrare un controller, "guidato" dalla controller application suddetta, in una rete SDN reale, e testarne il funzionamento

Controller application – Analisi dei requisiti

Computer di elaborazione

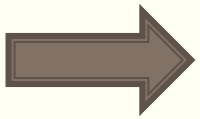
Elaboratori dotati di due interfacce di rete: da una, riceve del traffico, dall'altra, lo riemette senza manipolazioni, dopo averlo analizzato

Redirezione

Inoltro del traffico verso i computer di elaborazione selezionati, uno alla volta; al termine, il traffico viene infine inviato alla destinazione originale

Traffico di interesse

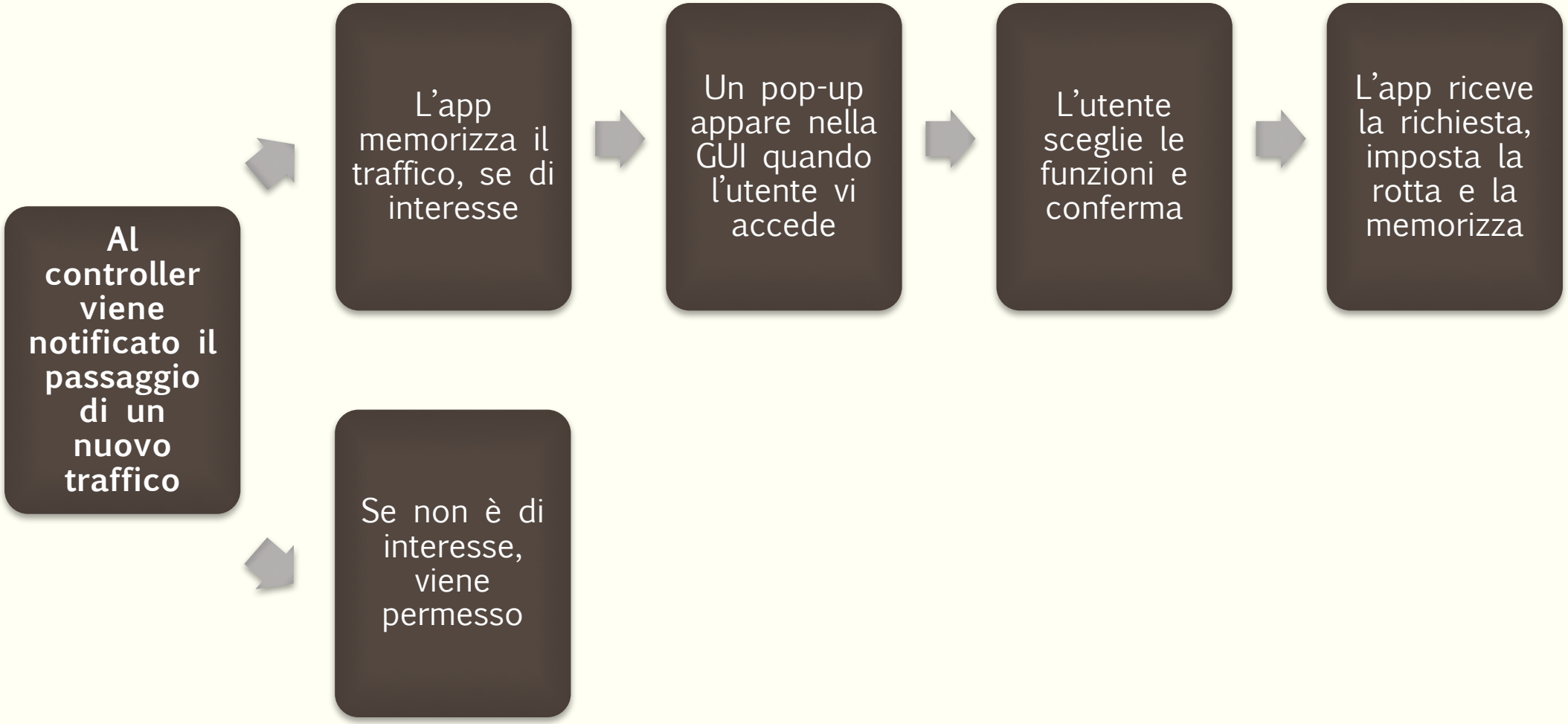
Traffico proveniente da una determinata locazione (sottorete o calcolatore)



Rotta

Entità che identifica, per ogni traffico di interesse, il percorso che esso deve compiere

Controller application – Gestione degli eventi



Controller application – Gestione delle richieste

L'utente richiede di visualizzare le rotte → L'app riceve la richiesta e restituisce le rotte finora memorizzate → La GUI visualizza tutte le rotte esistenti in una tabella

L'utente modifica una rotta → L'app riceve la richiesta e modifica la rotta, memorizzando la nuova rotta ed eliminando la vecchia

Controller application – Architettura realizzata

Core Logic

- Memorizza i traffici di interesse in ingresso
- Memorizza le rotte impostate
- Imposta la rotta per un traffico secondo le richieste

Server ↕ App

- Collega ogni richiesta al rispettivo metodo della Core Logic
- Realizza il server attraverso il quale verrà visualizzata la GUI

Web GUI

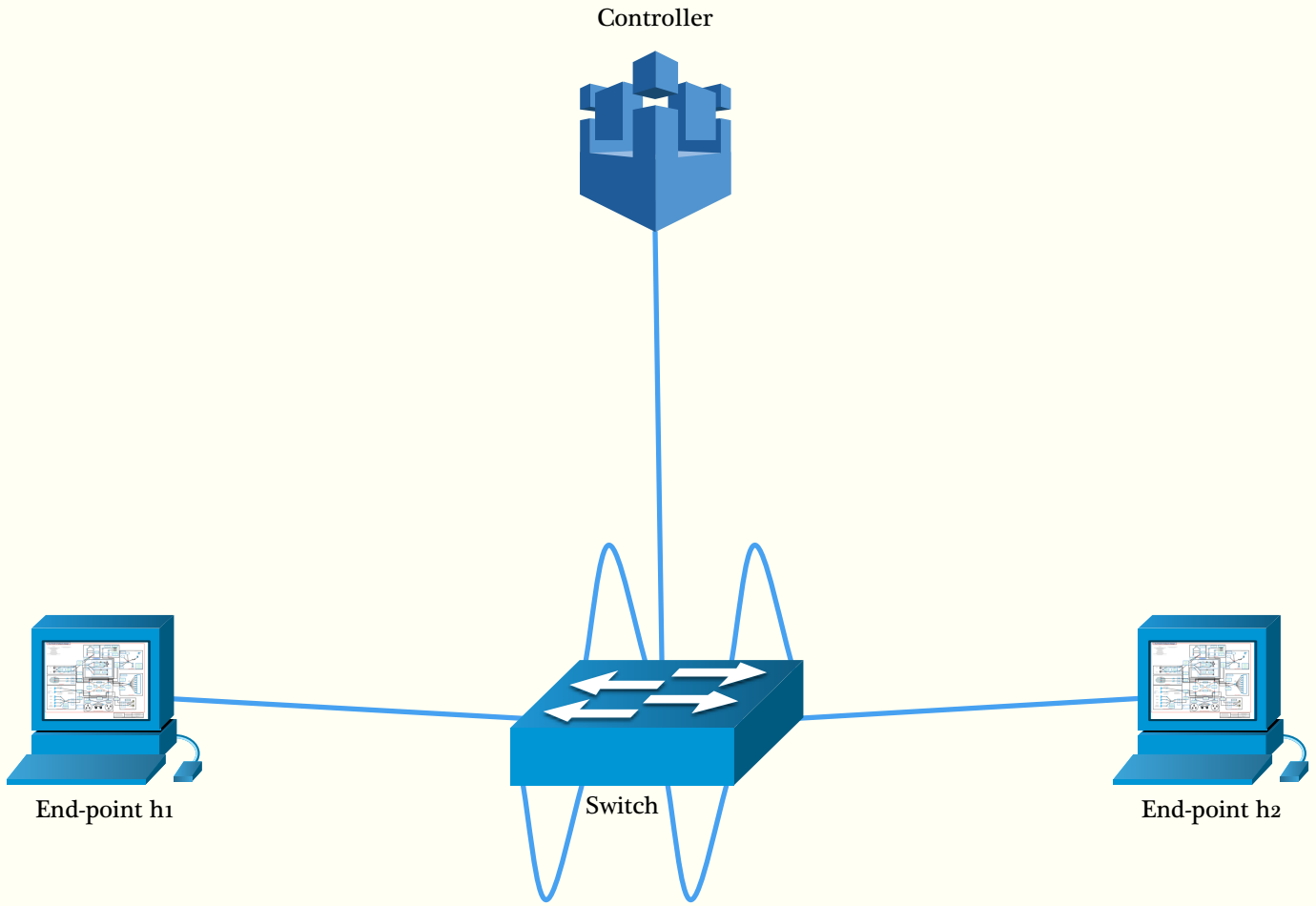
- Realizza la GUI
- Permette di richiedere l'impostazione di una rotta per i traffici di interesse
- Permette di visualizzare le rotte impostate e richiederne la modifica

Rete virtuale

Su macchina virtuale, è stata emulata una rete SDN, usando Mininet e Open vSwitch.

Bisogna verificare che:

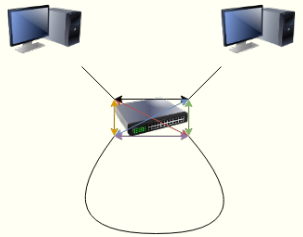
- Tutto il traffico proveniente da h2 passi indisturbato
- Per tutto il traffico proveniente da h1, all'utente venga richiesta l'impostazione di una rotta
- Le rotte esistenti vengano correttamente visualizzate e sia possibile modificarle con successo
- La rotta ridiriga il traffico associato nella maniera richiesta



Gestione dei loop di rete

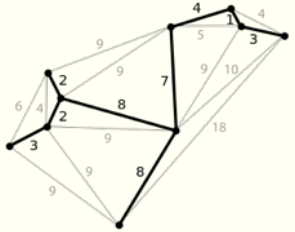
Problema

I loop di rete presenti rendono impossibile la comunicazione: non appena si interconnette la rete, i canali di comunicazione vengono saturati dal continuo fluire dei pacchetti



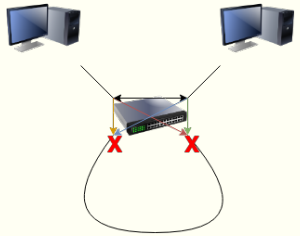
Prima soluzione

STP e RSTP, applicabili direttamente attraverso Open vSwitch, rappresentano solo una parziale soluzione al problema, in quanto permettono la comunicazione nella rete, ma chiudendo una delle due porte dei loop non permettono il fluire del traffico al loro interno



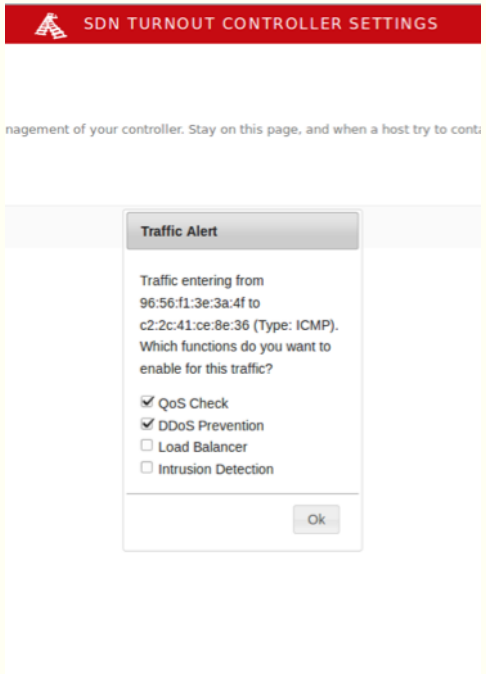
Seconda soluzione

Infine, si è risolto il problema negando la possibilità, ad ogni porta appartenente ad un loop, di fare flood nella rete, il che consente l'eliminazione di gran parte del traffico che si riscontra se non si applica STP, permettendo comunque il fluire del traffico all'interno di un loop

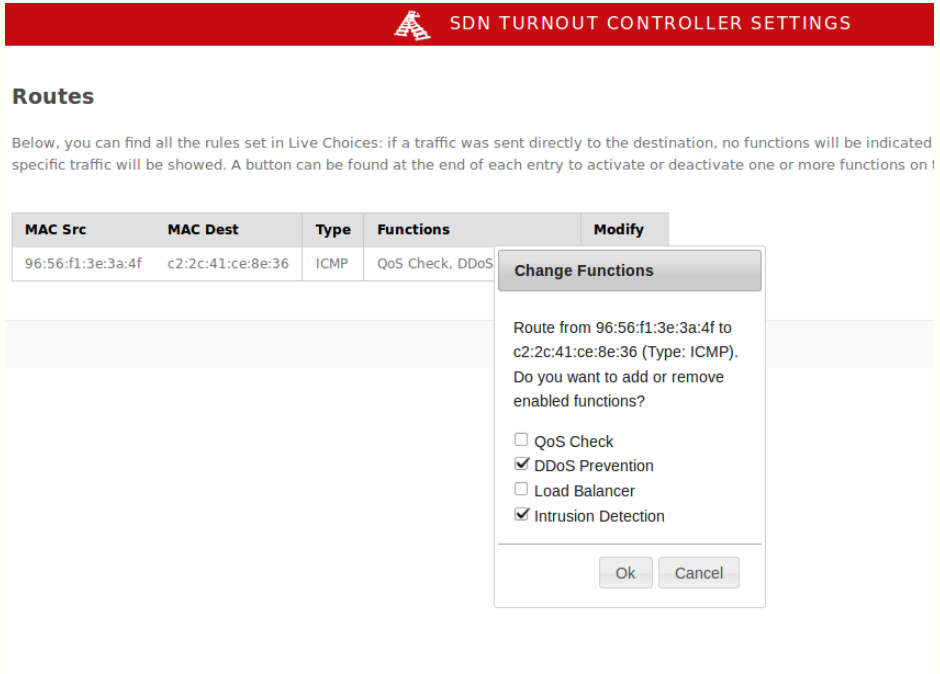


Test della rete virtuale - 1

L'end-point h1 tenta di comunicare con h2: a questo punto, appare un pop-up che avvisa l'utente e gli consente di impostare una rotta per quel traffico



Le rotta impostata viene visualizzata correttamente; inoltre, è possibile modificarla, selezionando o deselezionando a piacimento le funzioni esistenti



Test della rete virtuale - 2

Le flow entry presenti sullo switch, controllate in seguito alla modifica della rotta, risultano corrette

```
root@sdnhubvm:~# ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=23.553s, table=0, n_packets=15, n_bytes=1470, idle_age=4, prio
rity=1,icmp,in_port=1,dl_src=96:56:f1:3e:3a:4f,dl_dst=c2:2c:41:ce:8e:36 actions=outp
ut:5
 cookie=0x0, duration=23.553s, table=0, n_packets=15, n_bytes=1470, idle_age=4, prio
rity=1,icmp,in_port=6,dl_src=96:56:f1:3e:3a:4f,dl_dst=c2:2c:41:ce:8e:36 actions=outp
ut:9
 cookie=0x0, duration=23.553s, table=0, n_packets=15, n_bytes=1470, idle_age=4, prio
rity=1,icmp,in_port=10,dl_src=96:56:f1:3e:3a:4f,dl_dst=c2:2c:41:ce:8e:36 actions=out
put:2
 cookie=0x0, duration=133.268s, table=0, n_packets=35, n_bytes=3430, idle_age=4, pri
ority=1,icmp,in_port=2,dl_dst=96:56:f1:3e:3a:4f actions=output:1
 cookie=0x0, duration=432.560s, table=0, n_packets=81, n_bytes=7726, idle_age=13, pr
iority=0 actions=CONTROLLER:65535
```

Infine, è possibile verificare che il traffico, effettivamente, passa attraverso lo switch, ed arriva alla destinazione prefissata

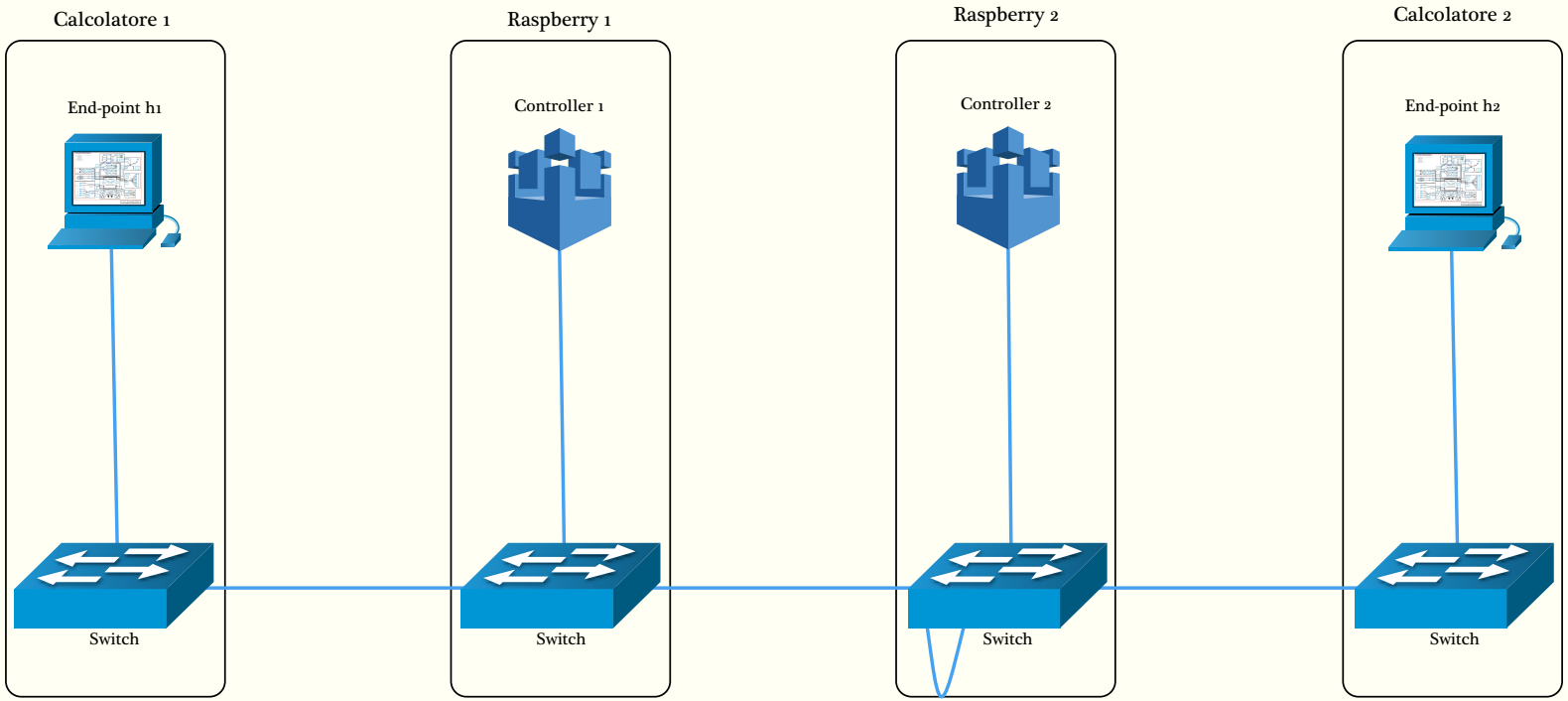
```
2
 cookie=0x0, duration=63.180s, table=0, n_packets=20, n_bytes=1960, idle_age=44, pri
ority=1,icmp,in_port=2,dl_dst=96:56:f1:3e:3a:4f actions=output:1
 cookie=0x0, duration=362.472s, table=0, n_packets=79, n_bytes=7642, idle_age=58, pr
iority=0 actions=CONTROLLER:65535
root@sdnhubvm:~# ip netns exec h1 ping 10.0.0.2
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=0.567 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.181 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.094 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.088 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.091 ms
```

Rete reale

Usando Mininet e Open vSwitch, sono state create diverse reti virtuali, che sono poi state interconnesse con collegamenti fisici.

Bisogna verificare che:

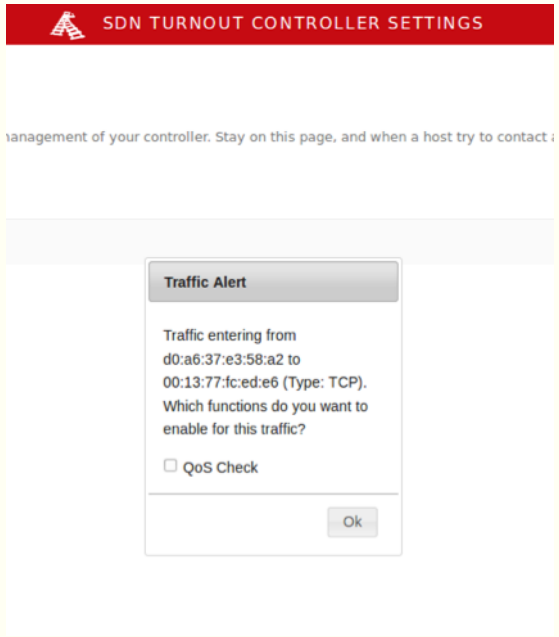
- La separazione delle responsabilità permetta il funzionamento della rete
- Le prestazioni fornite dai Raspberry permettano il deploy degli stessi in contesti più grandi

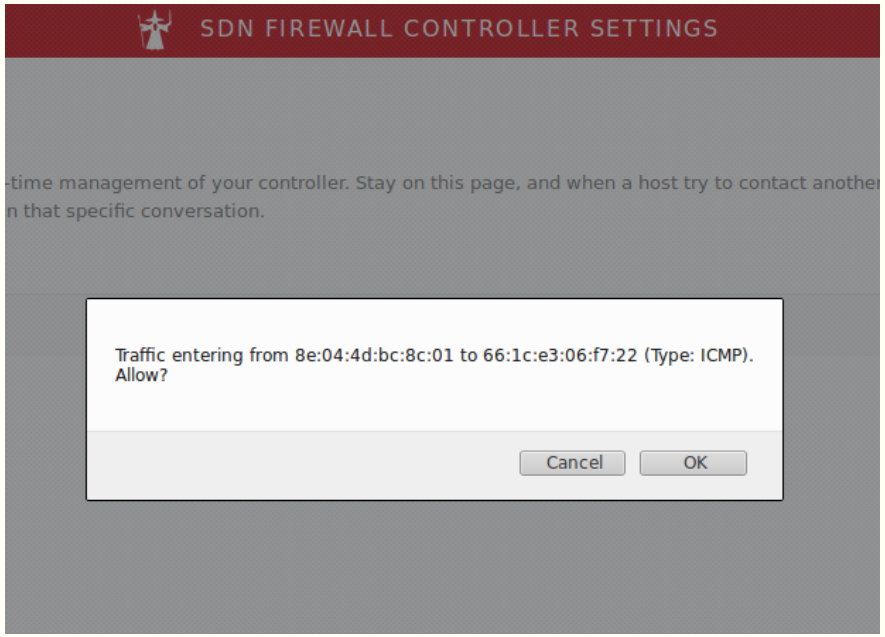


Test della rete reale - 1

Sul Raspberry 2 si verifica il passaggio del traffico tra il controller sul Raspberry 1 e lo switch sul Calcolatore 2: impostando una qualsiasi rotta, viene instaurato il canale sicuro

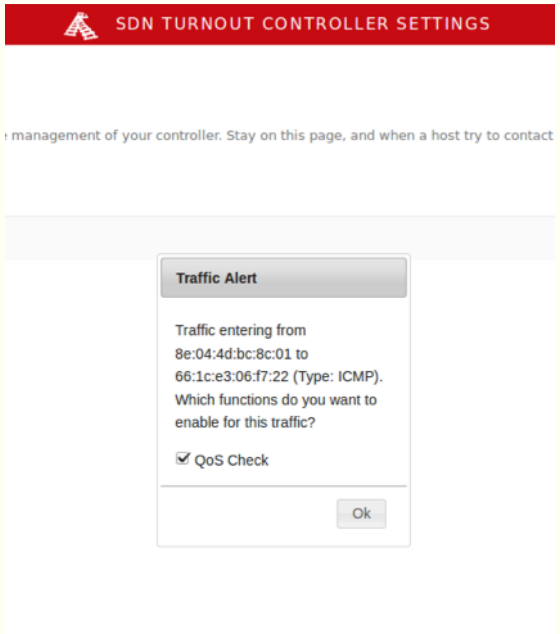
Cercando di contattare h2 da h1, il controller contenuto nel Raspberry 1 farà apparire un pop-up che permetterà all'utente di consentire o meno la comunicazione





Test della rete reale - 2

Ora, sul Raspberry 2, si verifica invece il passaggio del traffico appena consentito: impostiamo la rotta attivando l'unica funzione disponibile



E possiamo riscontrare, infine, che il traffico effettivamente passa; in tutto questo processo, si sono riscontrati picchi di lavoro per i Raspberry intorno al 30%

```
root@sdnhubvm:~# ip netns exec h1 ping 10.0.0.2
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=0.448 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.140 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.142 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.144 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.130 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 0.130/0.200/0.448/0.124 ms
root@sdnhubvm:~#
```

Conclusioni

Risultati ottenuti:

- Sviluppo di una controller application che permetta di creare e gestire rotte per dei traffici in ingresso
- Integrazione di un controller, guidato dalla controller application suddetta, in una rete reale SDN precedentemente realizzata
- Utilizzo del Raspberry come calcolatore di rete contenente sia lo switch che il controller

Sviluppi futuri:

- Scalabilità lato codice al fine di controllare più switch
- Ottimizzazione delle risorse del sistema operativo usato per i Raspberry al fine di incrementarne le prestazioni
- Implementazione di reali computer di elaborazione, al fine di condurre analisi sul traffico e modificare di conseguenza il comportamento della rete

GRAZIE PER L'ATTENZIONE