**Progetto SDN:Blocco traffico**

*Scrivere un programma che, da ogni host, generi traffico TCP iperf a intervalli casuali e con durate casuali verso altri host. Scrivere un programma Ryu che osservi i pacchetti SYN. Se una destinazione riceve più di X nuove connessioni nell’ultimo intervallo T la nuova connessione è bloccata. Siano X e T valori arbitrari.*

Obiettivi del progetto:

Lo scopo è osservare i SYN (solo primo pacchetto per ogni connessione TCP). Anche le connessioni non established sono conteggiate, a prescindere dall’esito positivo. È indipendente dalla topologia di rete (anelli, maglie, lineari, a stella …) e dal numero di host, per questo viene usato l’hop by hop Switched Eth (codice di riferimento lab switch5.py). Per la prova usiamo n host, su ciascuno verrà eseguito codice client e server, entrambi funzionanti sul medesimo con uno script per generare traffico casuale. La funzione è eseguita a scopo dimostrativo.

Per semplicità vengono gestiti i pacchetti facendo passare solamente i SYN in IPv4.

Il traffico gestito è unicamente IPv4 unicast per evitare loop sulla rete, viene evitato in questo modo il broadcast storm. Dato che gli host hanno bisogno di conoscere l’indirizzo mac degli altri host e associare l’ipv4 tramite arp (funzionamento in broadcast) è necessario implementare il proxy arp sul controllore.

Se un dato host riceve più di X Syn negli ultimi T secondi il nuovo SYN non verrà inoltrato nella rete, si può verificare la correttezza vedendo le richieste iperf in errore lato client o con TCP dump.

Dato che il progetto funziona anche su connessioni non estabilshed è possibile filtrare un eventuale attacco TCP SYN Flood (un host riceve tanti syn senza che siano vere connessioni, conseguentemente permane nello stato di timeout).

Topologia usata:

La topologia scelta per l’esperimento teorico a scopo di riferimento è una full mesh (maglia completa), ovvero ognuno dei 4 switch è interconnesso direttamente a tutti gli altri. Il numero di collegamenti in una topologia di questo genere è pari a n\*(n-1)/2. Siccome il fattore di dipendenza della complessità della rete (numero di link) varia come n^2/2 al crescere del numero di switch il numero aumenta quadraticamente. Nella nostra topologia sono presenti 4 host connessi ai rispettivi switch diversi. Gli switch implementano OpenFlow mentre il controllore implementa ryu. In vista dell’esperimento in Laboratorio i mac degli host non sono assegnati progressivamente (funzione - - mac), ma tramite proxy arp come specificato sopra.

Funzionamento:

* Il controllore apprende la topologia della rete osservando con il protocollo LLDP (Link Layer Discovery Protocol), apprende i mac address degli host e dove sono connessi, in questo modo ricava il primo tra i cammini minimi. Il passaggio dei dati avviene sui cavi control plane (gialli in lab), i quali hanno lo scopo di controllare la rete e gestire le packet out
* inizialmente tutti i pacchetti vengono inviati direttamente al controllore con priorità più alta di tutte le regole che istalleremo si esegue una flow mod per matchare i syn tcp in IPv4
* ricevuti i primi pacchetti si installa la Flowmod per instradare direttamente sul dataplane col percorso minimo calcolato prima
* Quando un host vuole parlare con un altro inizialmente il pacchetto verrà inoltrato al controllore il quale saprà lo switch sul quale inoltrare il pacchetto, lo inoltra (tramite packetout) e comunicherà allo switch il camminino verso quel determinato host tramite una flow mod. In questo modo la prossima richiesta di comunicazione verrà direttamente inoltrata senza passare dal controllore con l’accortezza che i pacchetti syn verranno sempre inoltrati al controllore per la priorità che possiedono.