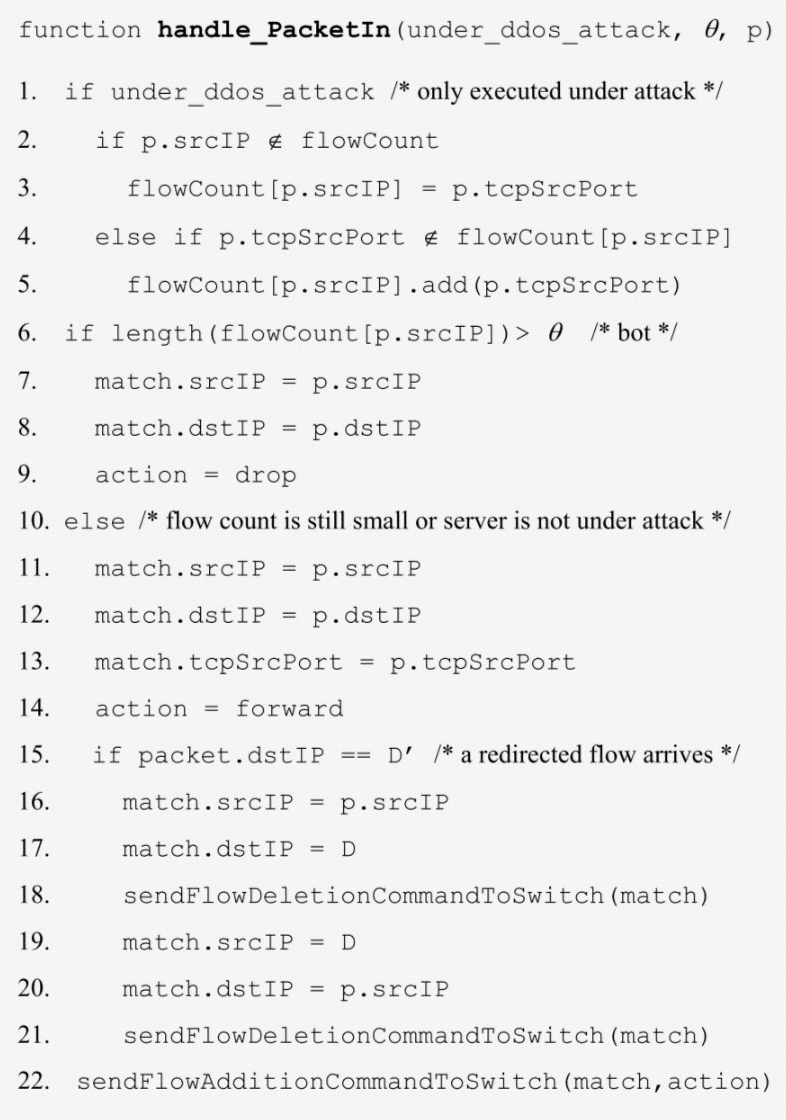
SDN-Oriented DDos Blocking Scheme for Botnet-Based Attacks

In this project we will create a SDN controller for defense against DDoS attacks. Through a REST API system, a web server can notify the begin of a DDoS attack; the controller will create an address change mechanism from D to D’.

In this project we will implement:



## IMPLEMENTATION

We will implement a module inside ddosdefence-floodlight-controller/src/main/java/net/floodlightcontroller/ddosdefence here, there will be tree file: DDoSDefence.java, EnableDefenceResource.java, IDDoSDefenceREST.java.

## ASSUMPTION

* First part of the paper, which establishes the change of server address from D to D ', will not be implemented.
* Gestiremo soltato il caso di indirizzi IPv4
* Number of IP addresses: predefined and sufficient for the purpose of the project. Our project involves the management of the number of IP addresses via a circular list for assignment.
* Our project plans to define a reCAPTCHA service: we will insert an equivalent implemented by us \*
* Gestione di ARP:
  + Change the IP address but not the ARP table, the server will auto-assign the new IP address.
  + This behavior is implemented in the forwarding module that must be included.
* DELETE management: delete the precise rule dictated by the code. If you want to delete more than one, we must manage it keeping track of previous entries or inserting a maximum priority rule that invalidates the others.
* Quando si abilita la seconda volta la difesa (passaggio da D’ a D’’) assumiamo che tutte le connessioni a D sono già terminate o già state filtrate, perché in questo modo possiamo disabilitare il server in ascolto su D quando la protezione è abilitata per una seconda volta.

\* Assumptions are optional and will be implemented according to the difficulties we will encounter in the project

## PARAMTERS

* Timeout ?
* \tehta = 3
* Cmax = 600
* Cattack = 300
* n = number of legitimate users
* k = number of bots
* \lamba norm
* \lamba attack
* \mu

## TESTING SYSTEM

## Our module will be tested through a script that we will produce that emulates the client and server. The architecture chosen for testing is Mininet.

## FINAL DOCUMENTATION AND REPORTING

At the end of this project will be produced:

* Final report: accompanying document illustrating the choices made and the reasons for these choices for the implementation of the module,
* Summary presentation in which the work carried out will be shown,
* Demo to show how the module works.

Questo verrà implementato, cosa si farà, quali sono le interfacce rest che farà il modulo.

come ? Elenco assunzioni

ci sono delle semplificazioni?

Cosa usi? Plug in? Pyotn?

~~Qui non ci sono assunzioni grandi (ignorare la prima parte del paper).~~

~~Soglia parametri – esponi quali parametri ci sono.~~

~~Testato attraverso un applicazione esterna da scrivere noi o uno script o plug in rest.~~

~~Architettura scelta per il testing dell’implementazione (mininet)~~

Proporre topologia di test

~~Assunzione # indirizzi ip (fai una lista circolare)~~

Se ho più hop per indirizzi, -> mantenere la storia dei passi precedente e corrente (va bene un passaggio solo)

Il server non origina del traffico da solo (differenziazione indirizzi e porta).

~~Pool di indirizzi: implementabile se c’è tempo~~

~~Servizio con capcha: inserire un equivalente , ma anche questo se c’è tempo.~~

~~Relazione:~~

~~documento di accompagnamento che illustra scelte fatte per il codice~~

~~Per l’esame:~~

~~una presentazione in cui presentiamo il lavoro e se vogliamo una demo per far vedere come funziona~~

~~ARP~~

1. ~~Cambi ip ma non cambi l’arp -> non funziona perché arp ha un addr table: due ip diversi avrebbero lo stesso mac addr. Per ogni indirizzo ip che usi devi avere un mac addr~~
2. ~~L’indirizzo ip deve essere gestito:~~ 
   1. ~~Server sotto attacco~~
   2. ~~Arp trasparente -> da non gestire perché il server si autoassegna il nuovo indirizzo D’.~~
   3. ~~Questo comportamento è implementato nel modulo forwarding che va incluso~~

Rotta che viene da fuori (forwarding in floodlight)

Switching su modalità autolearning -> aggiunge una entry nella tabela del forwarding

Mettersi prima del learning sw

Testa: sw con priorità massima – learning switch aggiunge regole

Regola in base alla porta sorgente per contrare le connessioni

~~TIMEOUT~~

~~Scegliere un valore noi, fare keep alive sul client per testare meglio.~~

Usa nc o curl 🡪 io qui non ho capito ☹

Modo per tenere traccia del numero di connessioni avute -> hash table

~~DELETE:~~

~~cancellare tutte le regole che hanno match: serve di guardare nella documentazione o provare. La delete cancella la regola precisa che gli dici. Se vuoi cancellare più di una si deve trovare noi il modo ma carlo non sa se si più fare, se non tenendo traccia delle entry già fatte o inserendo una regola a priorità massima che invalida le altre e levi di mezzo il modulo di sotto.~~

Il modulo sotto che fa autolearning cancella

Lo sw lavora per mac addr, se ho su una porta un host e su un'altra un altro allo sw non importa dell’ip