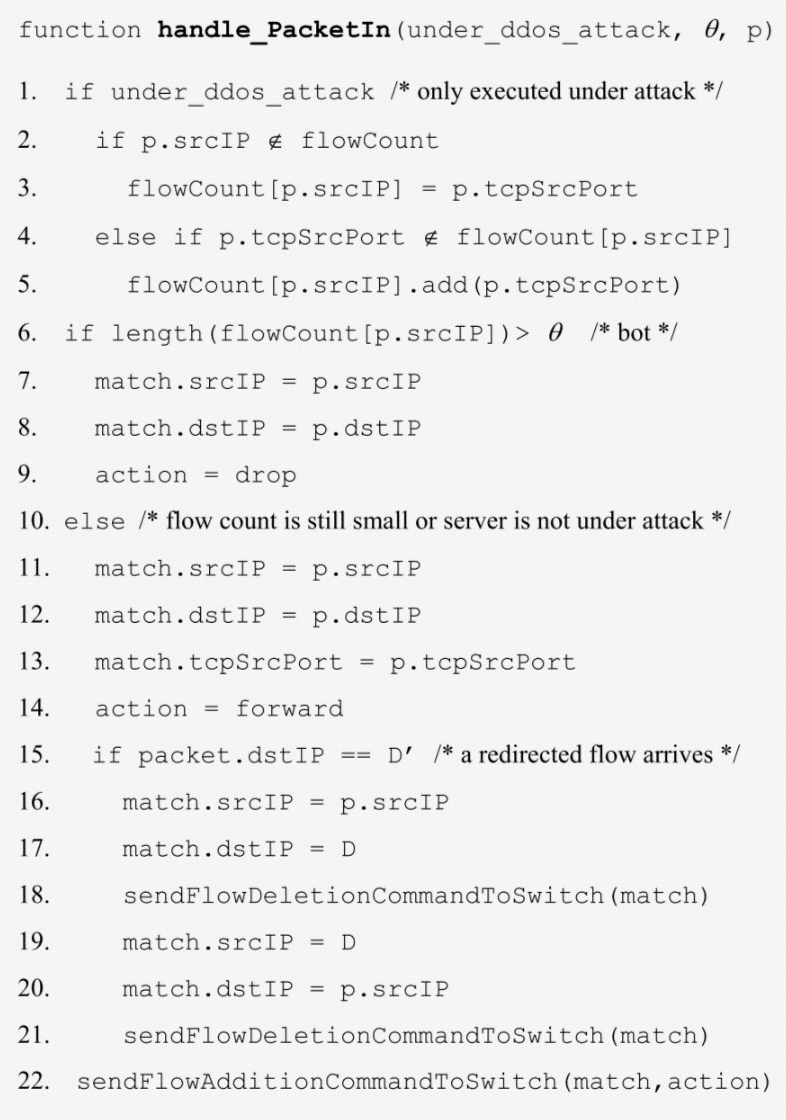
SDN-Oriented DDos Blocking Scheme for Botnet-Based Attacks

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’.

## IMPLEMENTATION

In this project we will implement:



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.

## ASSUMPTIONS

* First part of the paper, which establishes the change of server address from D to D ', will not be implemented.
* 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 \*
* ARP Handling
  + 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.

\* 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 using a set of scripts that we will produce, that emulates the client and server. Mininet will be used to create a virtual network composed by a server and multiple (N) clients.  
  
Server (h1) can use addresses from a /24 subnet, in our case 7.7.7.0/24 will be used.  
Each client (h2-hN) will use an addresses took from the 80.80.80.0/24 subnet.  
The Controller will listen for OpenFlow PACKET\_IN connections on 127.0.0.1:6653.

One OpenFlow switch (s1) will be used to connect all the previous entities. No routing or complex forwarding rules are required because the switch will solve the problem only using L2 Table packet switching. The only rule to use on both server and clients is

“route add -net 0.0.0.0/32 dev <out interface>”

to send all the packet originated from the emulated device through the interface connected to the switch. ARP on the originating packet device will make sure to set the correct destination MAC address.

Clients can be regular service users or malicious ones:

* Bots: will continuously do HTTP requests to a single target, but will not be able to compute any complex forwarding (like CAPTCHA or JavaScript implemented)
* Clients: will perform regular HTTP requests and be able to do forwarding

Both behaviours will be simulated using two separate scripts. Pseudo-code examples could be:

**./start\_bot.sh serverip**

While(true) {

Establish HTTP on server connection using keepalive; exits when connection is closed

}

**./start\_client.sh serverip**  
while(true) {

Establish HTTP on server connection using keepalive; exits when connection is closed

serverip = (if forwarded) ? newaddress : serverip ; # user can process forward

sleep 2; # user will generate less traffic than bots

}

As stated on the paper, the forwarding method must be sufficiently complex to not be executed by the bots. However this represents an issue to ./start\_client.sh script that is, in reality, a bot itself. To overcome this issue, we can initially simplify the forwarding mechanism to let the script do the forward. However the ./start\_bot.sh will ignore, in good faith, the simplified forwarding directive.  
An example of HTTP Page forwarding response could be one not containing an HTML page (we can check if contains <html> or not) but containing the address to forward. This is just for a testing purpose and can be subjected to changes due to chosen web server implementation.

### Checking the correct behaviour for OpenFlow rules:

mininet> dpctl dump-flows  
This command can be used to dump all the switch flow entries.

## 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