於 P4 實現基於 Euclid 的DDoS 防禦策略

DDoS mitigation strategy based on Euclid and implemented with P4

專題組員：余紹桓、陳明閎

指導教授：張燕光

DDoS與SDN簡介

DDoS(Distributed Denial of Service)，又稱分散式阻斷服務攻擊，
利用傳送大量**偽造封包**到目標電腦的網路，
使其系統資源耗盡，造成**服務中斷**，以此達到攻擊的目的。

而近年來興起的**SDN(Software Defined Network)**，
也就是軟體定義網路，將交換器的Control Plane與Data Plane分離，
解決了傳統交換機無法隨時更新與集中控管的困境，
降低了時間與金錢成本，
同時也非常適合用來研究DDoS攻擊的解決辦法。

P4簡介

P4正是一種能夠實現SDN的程式語言，語法格式和C語言很像

P4語言的特點有：

- Protocol Independent （可相容任何通訊協議）
- Target Independent （支援任何平台）
- Field Reconfigurable （隨時都能修改）

我們可以在.p4檔撰寫如以下的行為：

- 提取封包裡的指定header
- 進行運算+判斷，將封包送往指定出口，或是丟棄
- 對封包的header進行修改

可參考官方github： P4 tutorial

▲P4 ICON

P4簡介

在使用p4檔之前，會需要先安裝 **p4c** 與 **bmv2** ：

p4c：p4的編譯器，

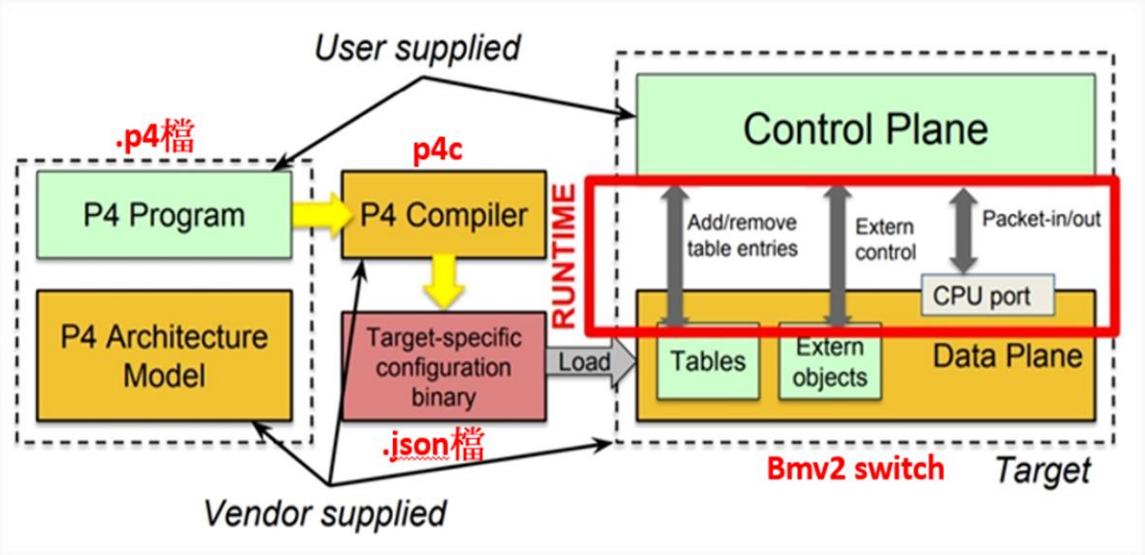
會將檔案編譯成.json檔和其他東西，

json檔定義了switch的行為。

bmv2：支援p4語言的一種軟體switch，

讀入json檔後就能模擬switch的行為，

可設定每個port所對應的網路卡。



▲ p4c與bmv2

可參考官方github：p4c , bmv2

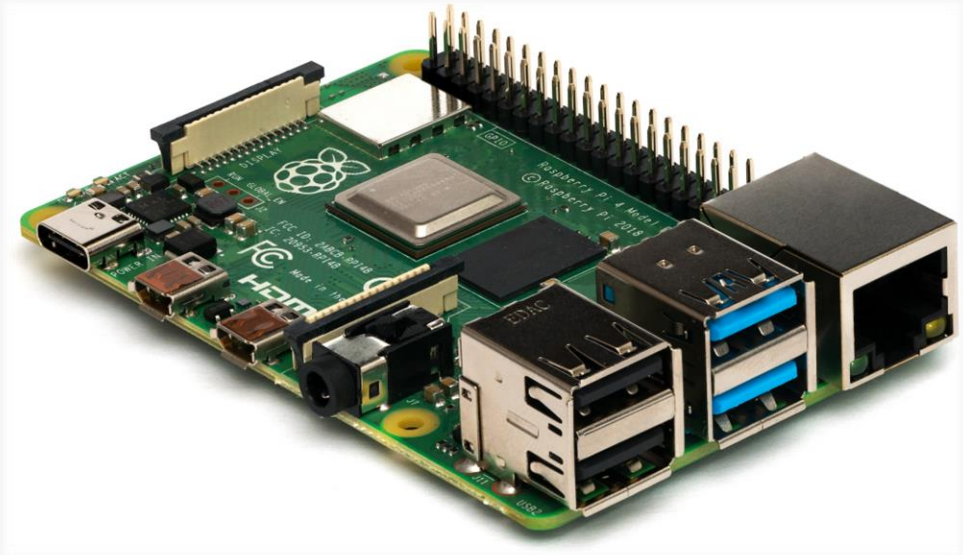
介紹P4PI

P4PI就是將P4實現在**raspberry pi(樹莓派)**裝置上 ，

讓樹莓派成為一台實體switch，

參考自P4官方團隊的[P4PI github]

裡面有提供P4 Raspberry Pi Imager檔、後續安裝與操作上的教學。



▲樹莓派

▲P4PI 操控介面示意圖

介紹Euclid

Euclid策略的步驟可以簡化成以下幾點：

1. 每次蒐集**1萬個封包** (數字可改)，當成是**一個Window**
2. 對這1萬個**來源IP (src_IP)與目標IP(dst_IP) 分別計算entropy**，判斷是否受DDoS攻擊
3. 確認受到攻擊後，開始進行防禦。
4. 防禦的過程中，也會對每個Window計算Entropy，視情況停止防禦。

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 計算Entropy

假設現在有**DDoS攻擊**，根據Entropy公式：

目標IP(Dst_IP)：受害者IP的出現次數會**暴增**，因此Dst_IP所得的entropy值會**非常低**，趨近於0

來源IP(Src_IP)：任何IP的出現次數都會**很低**，因此Src_IP所得的entropy值會**非常高**，趨近於$\log_2(m)$

因此，根據這兩者所得的entropy值，就可以用來**判斷是否遭遇DDoS攻擊**。

$$H(X) = \log_2(m) - \frac{1}{m}\sum_{x=1}^{N} f_x \log_2(f_x)$$

X：Window的編號

m：Window Size的值

fx：某IP的出現次數

N：共有幾個不同IP

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 計算Entropy

而Euclid中，大致也是這樣去判斷DDoS攻擊的，

但不是直接用Entropy值去判斷，而是只**保存部分比例**，並更新給某個變數

例如num = 0.3*H(X) + (1-0.3)*num，再用num**值**去當作**判斷攻擊的依據**。

好處是能夠避免短期震盪，像是當**正常流量出現短期DDoS攻擊特徵**的時候，

就比較不會被**誤判**成是遭遇DDoS攻擊。

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 如何防禦

Euclid策略會使用上一個被判斷成DDoS攻擊的**Last Window**資訊，

以及先前記錄的**Safe Window**資訊，來判斷是否阻擋目前收到的封包：

$$V_{src} = f_{src,last} - f_{src,safe} \text{ and}$$
$$V_{dst} = f_{dst,last} - f_{dst,safe}$$

$$V = V_{dst} - V_{src}$$

$f_{src,last}$：該封包的src_IP 於 **Last** window中的出現次數。

$f_{src,safe}$：該封包的src_IP 於 **Safe** window中的出現次數。

$f_{dst,last}$：該封包的dst_IP 於 **Last** window中的出現次數。

$f_{dst,safe}$：該封包的dst_IP 於 **Safe** window中的出現次數。

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 如何防禦

當DDoS攻擊發生時，

攻擊者產生的封包，目標IP(dst_IP)會是受害者IP

而受害者IP於Last window中的出現次數會**暴增**，在Safe window中就是**正常次數(很少)**

因此攻擊者發送的封包，所得的$\mathbf{V_{dst}}$**會很大**。

$$V_{src} = f_{src,last} - f_{src,safe} \text{ and}$$
$$V_{dst} = f_{dst,last} - f_{dst,safe}$$

$$V = V_{dst} - V_{src}$$

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 如何防禦

另外在DDoS攻擊時，

Last window中任何src_IP的出現次數都會**很低**，

並且攻擊者的src_IP基本上也不會出現在safe window中。

因此攻擊者發送的封包，所得的**V_{src}會很小**。

$$V_{src} = f_{src,last} - f_{src,safe} \text{ and}$$
$$V_{dst} = f_{dst,last} - f_{dst,safe}$$

$$V = V_{dst} - V_{src}$$

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 如何防禦

因此**DDoS產生的封包**，所得到的**V值會明顯很高**

我們就能利用這個特性，對每個封包去計算V值，

當**V值超過某個閥值(T)**後，就可以判斷目前的封包為**DDoS攻擊封包**，

就能選擇**將封包丟棄**，或是轉送到指定出口，用於後續分析。

$$V_{src} = f_{src,last} - f_{src,safe} \text{ and}$$
$$V_{dst} = f_{dst,last} - f_{dst,safe}$$

$$V = V_{dst} - V_{src}$$

V <= T ： 判定為正常封包

V > T ： 判定為DDoS攻擊封包

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

介紹Euclid – 補充說明

P4語言並不支援諸如**乘**、**除**、**取餘數**或是**log運算**，甚至是**for loop**等等...

因此要達到Euclid中的運算，必須用其他投機取巧的方式來配合

這點在Euclid的論文中有更詳細的介紹。

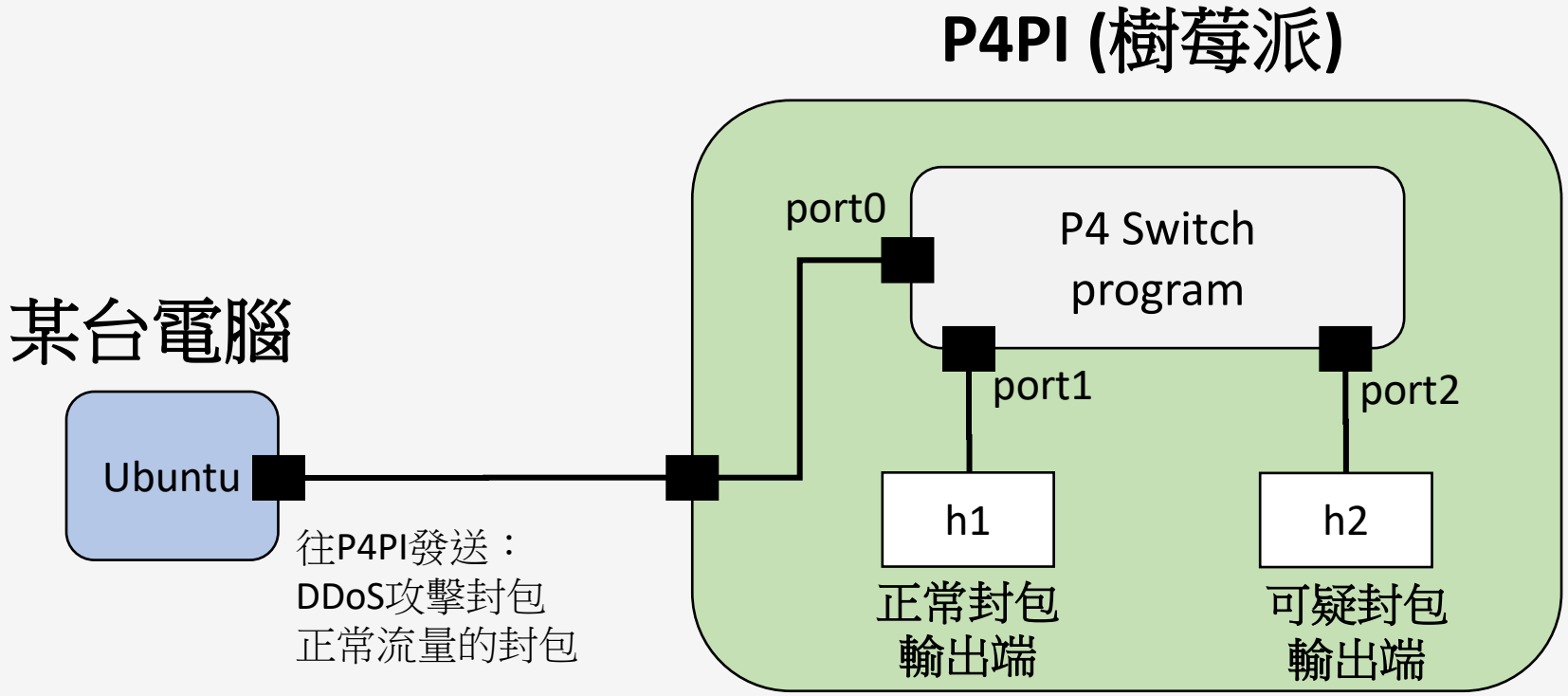
而Euclid團隊所使用的bmv2與p4c環境，是有自行改寫並重編譯的，

而我們P4PI中的bmv2與p4c是**乾淨原始的環境**，

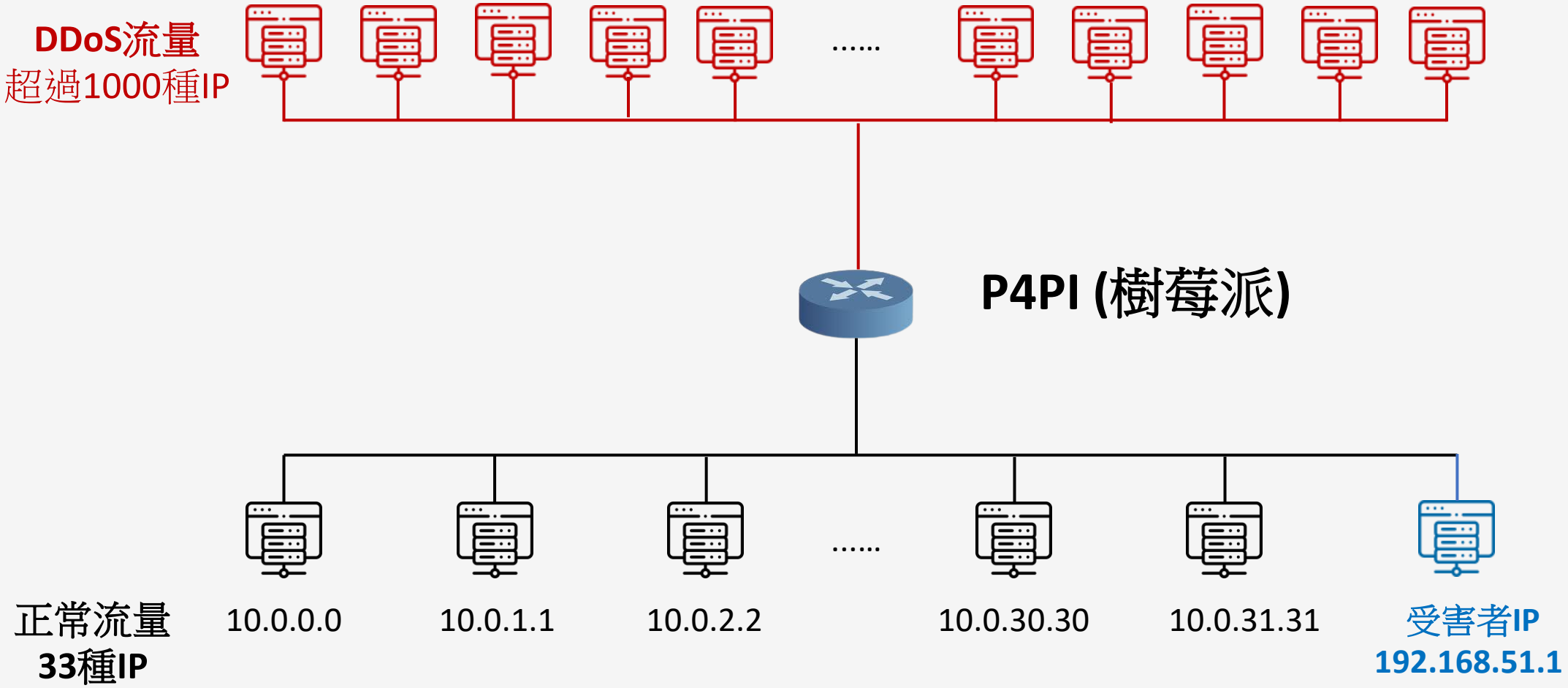
因此為了能夠執行，我們有將Euclid改寫一部分程式碼，達到相同效果。

取自 Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation

測試說明 – 實際環境


P4PI (樹莓派)
port0
P4 Switch
program
port1
port2
某台電腦
Ubuntu
h1
h2
往P4PI發送：
DDoS攻擊封包
正常流量的封包
正常封包
輸出端
可疑封包
輸出端

測試說明 – 網路拓樸


DDoS流量
超過1000種IP
.....
P4PI (樹莓派)
.....
正常流量
33種IP
10.0.0.0
10.0.1.1
10.0.2.2
10.0.30.30
10.0.31.31
受害者IP
192.168.51.1

測試說明 – DDoS攻擊流量

攻擊流量的pcap有一部分取自：

(CAIDA DDoS 2007 Attack Dataset)

同時會有超過1000種src_IP

去攻擊受害者IP (192.168.51.1)。

ICMP封包：72.78%

TCP封包　：9.14%

UDP封包　：18.08%

No.	Source	Destination	Protocol
491241	202.107.62.252	192.168.51.1	ICMP
491242	51.111.201.218	192.168.51.1	ICMP
491243	167.24.138.224	192.168.51.1	ICMP
491244	195.34.25.201	192.168.51.1	ICMP
491245	194.8.39.30	192.168.51.1	ICMP
491246	60.13.50.80	192.168.51.1	TCP
491247	167.24.138.224	192.168.51.1	ICMP
491248	202.107.62.252	192.168.51.1	TCP
491249	54.108.148.19	192.168.51.1	ICMP
491250	200.78.254.138	192.168.51.1	ICMP
491251	208.121.6.92	192.168.51.1	ICMP
491252	193.1.97.200	192.168.51.1	ICMP
491253	167.243.30.187	192.168.51.1	ICMP
491254	199.192.55.223	192.168.51.1	ICMP
491255	57.222.217.116	192.168.51.1	TCP
491256	196.196.205.211	192.168.51.1	ICMP
491257	196.103.74.152	192.168.51.1	ICMP
491258	133.65.48.92	192.168.51.1	TCP
491259	193.216.28.75	192.168.51.1	ICMP

▲DDoS攻擊流量示意圖

測試說明 – 正常流量

正常流量的pcap是自行用scapy生成的

裡面共有33種IP，隨機抽兩個互傳封包。

10.0.1.1 , 10.0.2.2 , 10.0.3.3 …

10.0.31.31 , 10.0.32.32 ,

受害者IP (192.168.51.1)

No.	Source	Destination	Protocol
6778	10.0.19.19	10.0.9.9	ICMP
6779	10.0.9.9	10.0.19.19	ICMP
6780	10.0.24.24	10.0.31.31	ICMP
6781	10.0.31.31	10.0.24.24	ICMP
6782	10.0.30.30	10.0.20.20	ICMP
6783	10.0.20.20	10.0.30.30	ICMP
6784	10.0.18.18	10.0.9.9	ICMP
6785	10.0.9.9	10.0.18.18	ICMP
6786	192.168.51.1	10.0.8.8	ICMP
6787	10.0.8.8	192.168.51.1	ICMP
6788	10.0.27.27	10.0.5.5	ICMP
6789	10.0.5.5	10.0.27.27	ICMP
6790	10.0.25.25	10.0.26.26	ICMP
6791	10.0.26.26	10.0.25.25	ICMP
6792	10.0.24.24	10.0.29.29	ICMP
6793	10.0.29.29	10.0.24.24	ICMP
6794	10.0.20.20	10.0.2.2	ICMP
6795	10.0.2.2	10.0.20.20	ICMP
6796	192.168.51.1	10.0.0.0	ICMP

▲正常流量示意圖

測試結果 (TPR 與 FPR)

我們這次的測試，Window的封包總數都是用2^13

也就是8192個封包。

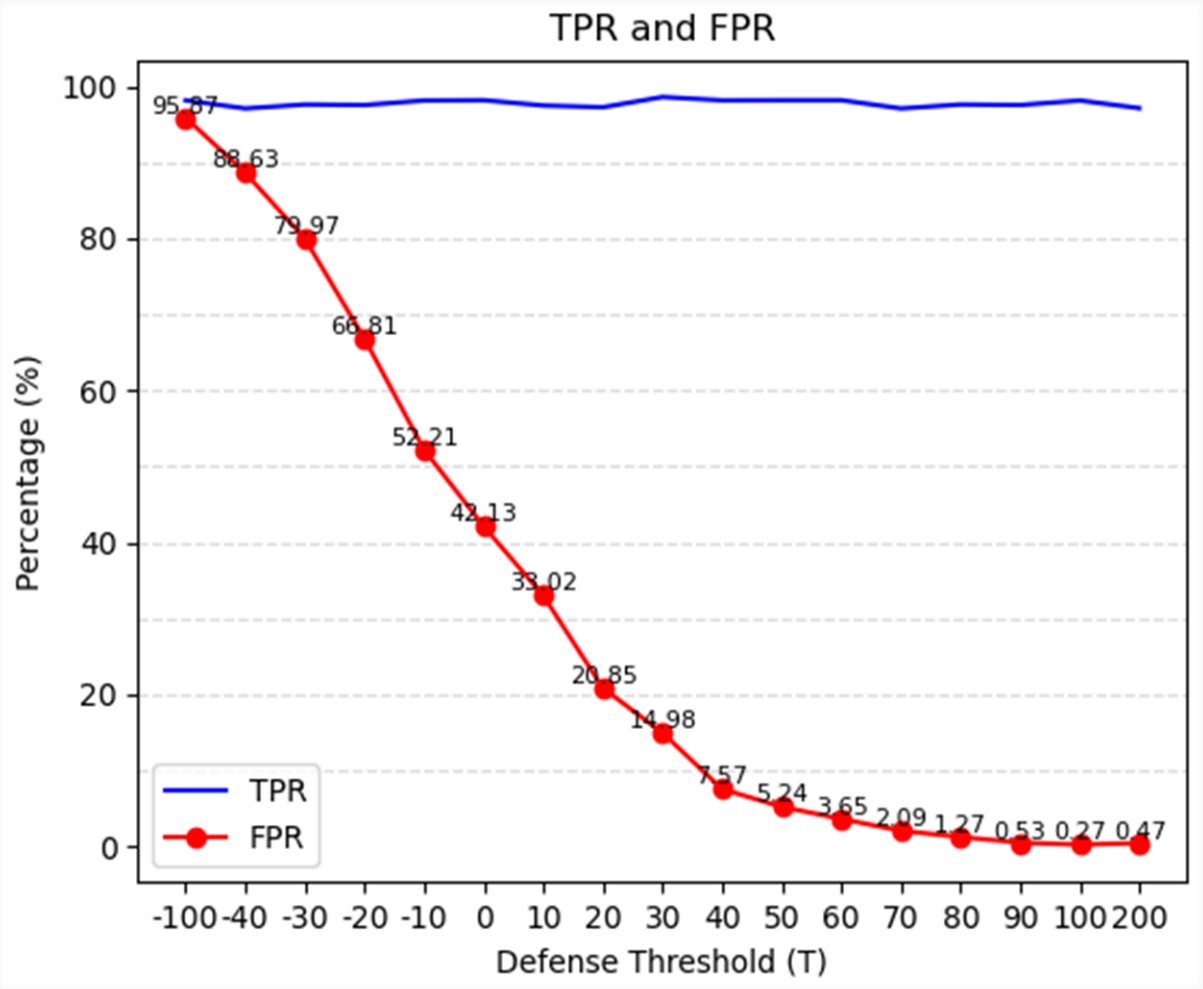
總體來說，當閥值T過低的時候，

會導致連正常流量的封包，也會被當成攻擊封包

FPR就會變高，因此T要設成適當的值，才不會誤擋。

TPR (True Positive Rate) ：攻擊流量的成功阻擋比例

FPR (False Positive Rate)：正常流量的誤阻擋比例



▲TPR與FPR 折線圖

測試結果 (TPR 與 FPR)

當T值過大時，會造成攻擊封包被當成正常封包，

導致TPR急速下降，因此閥值T也不能設太大。

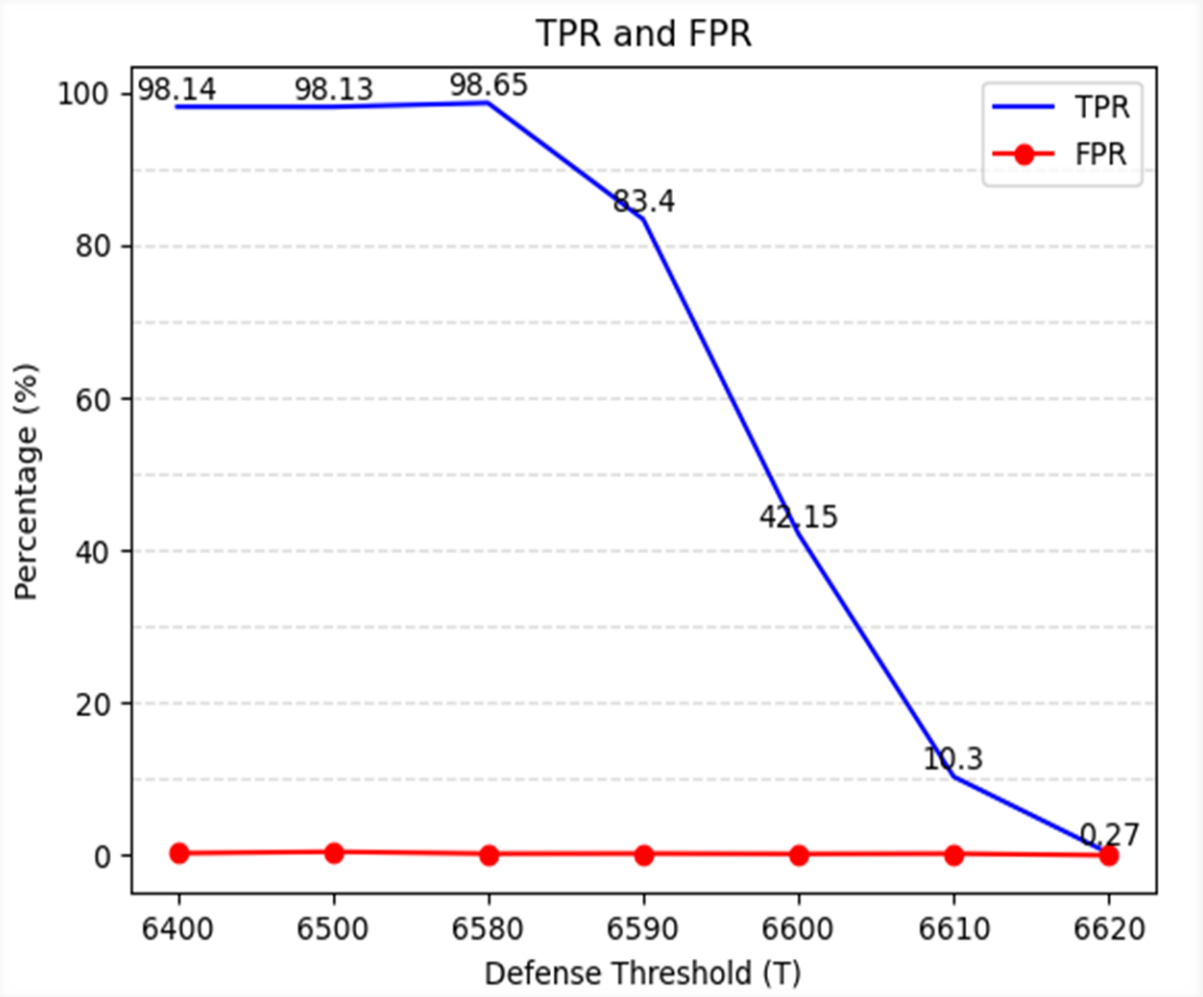
這個大小會和**DDoS流量速率占比、Window Size**有關，

譬如這次測試：DDoS流量佔了80%流量

因此T值大約在 8192 * 80% ≒ **6554** 之後TPR會迅速下降。

TPR (True Positive Rate) ：攻擊流量的成功阻擋比例

FPR (False Positive Rate)：正常流量的誤阻擋比例



▲T值設太高的情況

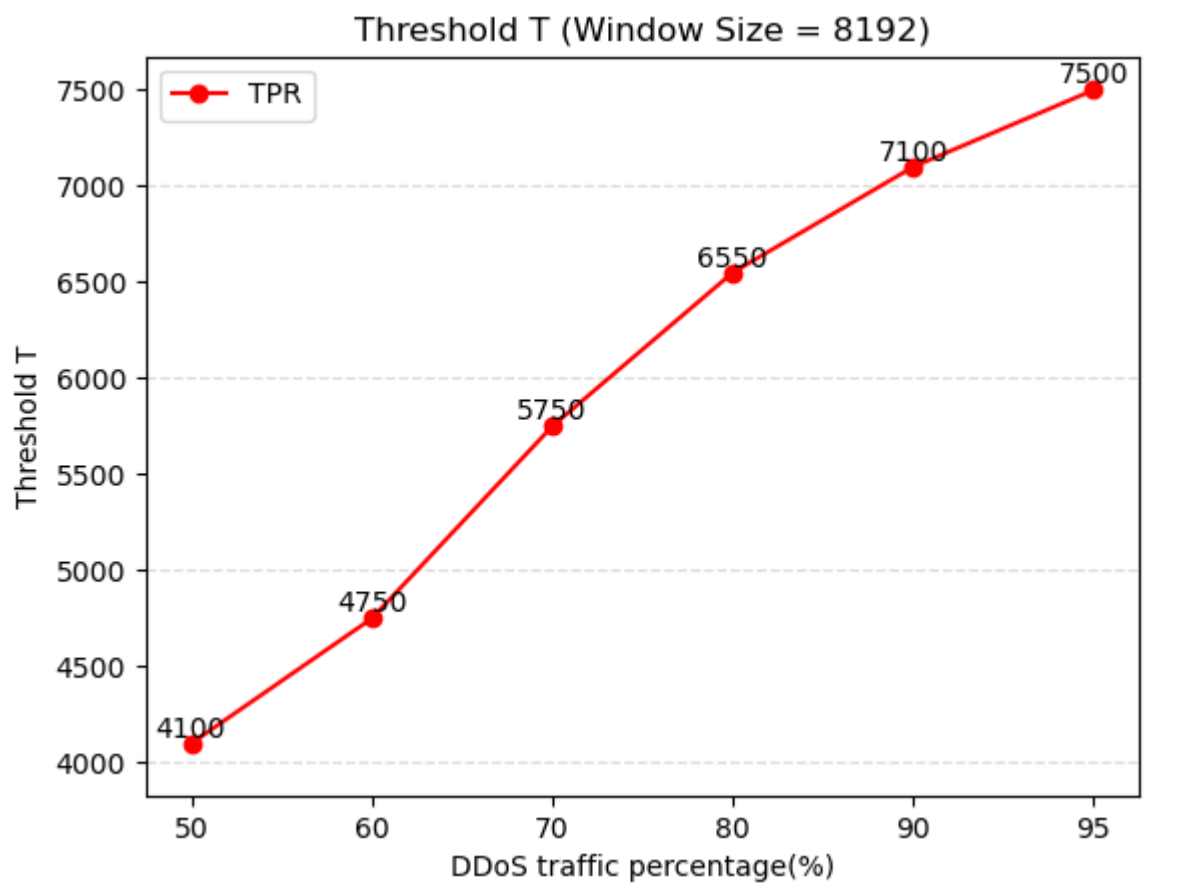
測試結果 (T值與DDoS流量之關係)

我們針對不同的DDoS流量百分比，

去記錄了當閥值T設在約多少後，TPR會迅速下降。

可以發現這個值會和DDoS流量百分比有關

當DDoS的流量佔越多百分比時，T值的上限會提高。



▲T值與不同DDoS流量之關係

測試結果 (Defense Delay)

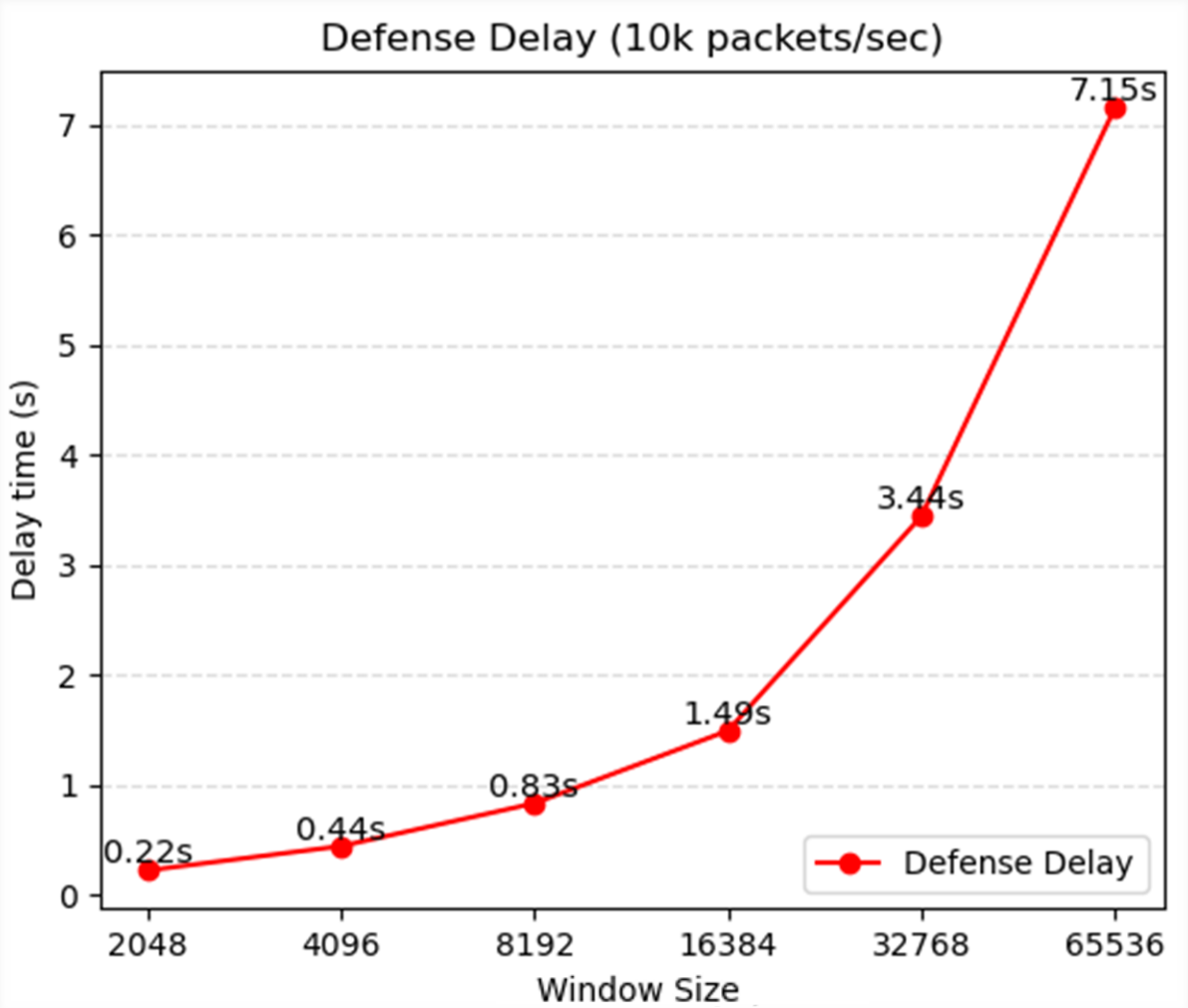
每次啟動防禦的時機，

只可能發生在**Window滿後**，計算Entropy後。

因此**Defense Delay** ，會和**Window**大小成正比

Window若設太大，會導致**Defense Delay**大幅提升。

但若Window設太小，也可能導致DDoS偵測上會不準。



▲Defense Delay折線圖 (10kbps)

測試結果（處理數量）

Euclid在未受攻擊的**狀態**下，

會有維護Window資訊、計算Entropy偵測攻擊的動作，

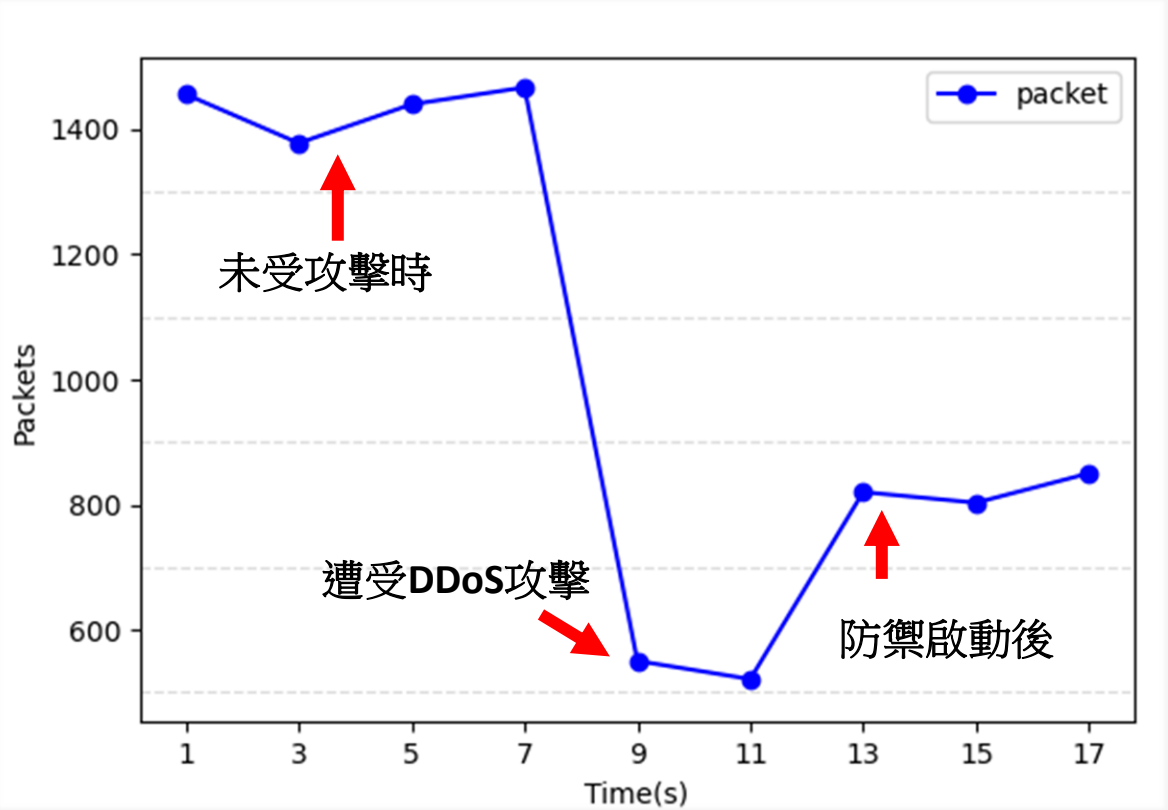
這種階段中最多能通過1400個正常封包。

當**DDoS攻擊發生**，並且**防禦未啟動**時，

每秒通過的正常封包數就只剩下500個左右，

在**防禦啟動**後，Euclid會開始抵擋DDoS封包，

這時每秒通過的正常封包數就會回復到800個左右。



▲三種狀況的每秒正常封包處理量

(註：攻擊流量約為正常流量的2倍)

結語

Euclid只使用到IP資訊，是一種general的作法，可以解決許多種DDoS攻擊。

Euclid使用到大量複雜計算，每秒可處理封包數很低，switch效能可能會成為瓶頸。

(P4PI switch在僅有**ipv4轉發**功能時，能大約每秒處理**2萬個封包**

運行Euclid後每秒只能處理**1400個封包**，效能低了很多。)

不過其價值在於TPR與FPR的表現良好，能確保大部分DDoS封包不會進入網路群中

可以避免host的資源被DDoS封包耗盡，導致服務中斷。

當未來switch的效能不會成為瓶頸時，這種策略就會有更高的價值。

感謝觀看