20160407

* Locality產生方式:抽一個rule，drain一個packet出來，locality只的是這個packet要重複連續擺幾次
* Scope:wildcard數量，現在預期wildcard數量越多，cache效果應該要越好
* 可以做的實驗:產生一個高locality的packet，然後把他打亂，越亂封包相異度越高，cache成績應該越差
  + 所以這裡的locality可定義改變了，指的是，進來的封包有多少種可能
  + 我認為這才是locality真正的作用，所以packet產生出來後，應該先打亂，可以把他稱為dynamic程度，來區分不同亂度
  + Dynamic越低，如果是packet-based的，hit ratio一定越高
  + 可是這裡的實際cache hit數量，應該要乘上一個stream通常的連續封包大小，所以會高很多
    - 因此cache miss ratio算法，
      * 原本是CACHE MISS/TOTAL ROUND
      * 變成 CACHE MISS/(TOTAL ROUND \* stream size)
* 本來在投影片做的實驗code\_v2下，不同locality可能意義不大，因為等於說有兩個變數應該要控制
  + 因此才會造成，好像相鄰封包的相異程度，幾乎就是cache miss程度，現在預期應該是high locality的 cache miss ratio 應該更糟
* 實驗:把三個打的一樣亂，然後去看cache miss ratio，最後可以乘上stream size 來把miss ratio 變好看
  + 預期 low locality應該還是會好一點
  + 預期BF表現應該一樣
* 預期結論應該是只要是locality高的，cache hit ratio都會比要好，even在rule-based也很容易想像
* 實驗: 應該固定在高locality，打亂後，調整不同scope，rule-based在scope比較高的時候，會比較賺
* 實驗: 和packet-based做比較，為了達到同要的cache hit ratio, packet based需要比較大的cache空間
  + Ex. If 只有10條rule，但scope很大，所以packet-based需要超大空間，rule based 只要十條 就可以達到100%
* 後來的這些實驗結果 CACHE HIT RATIO可能都不到10%我認為是合理的，所以這時候BF的重要性才會出來
* Packet-base 的實驗幾乎不用做，因為他就幾乎是決定於後來我們乘的stream size
* 檔案說明
  + Code v2就是專門做packet base，code v3也有packet base差別在於hash function 不同
  + D槽的code和data是最完整的
  + Test\_data是在high locality下，不同seed的code v2的測試
  + Test\_data\_for\_different\_scope是，在code\_v3下，想看不同scope的rule會不會賺
* 實驗設計:用code v3
  + 產生十條RULE 兩種scope，用high locality，產生100000個packet，只給四條cache， 看hit rate
* 實驗設計:用 code v2
  + 產生100條rule，產生10000packet，給10條CACHE，用high low locality 打亂後測試hit rate

20160411

* Slides 201604\_v5 實驗數據算法說明
  + TCAM power model 都是使用0.1nm 製程(1000 104 0.1)
    - 最小只能到0.1，不能到0.09nm!!
  + 1000\*104 (320pJ)tcam cache, 10000\*104 (3244pJ) tcam lookup table
  + Lookup table 當作是有加filter的，根據以前的數據，大約會有5%open，filter bit 數取5 bit(10000\*5 -> 498pJ)
* 3244\*0.05+498=660 (filter lookup table power)
* BF overhead 20pJ，用7000\*5的search，乘上5
* Power=(Power\_cache\*(1-P)+Power\_cache\*P\*FALSE\_POS+Power\_lookup\*P+BF\_overhead)