HOW TURBO ACL'S WORK

Basic information to know :

- The normal way IOS matches traffic to ACL's is that.
  - When a packet is received (in case of an input acl) , the IOS checks if there are any ACL's applied to that interface
  - If yes, it takes the first ACE and checks the ACE's conditions against the packet
  - If there is no match, it moves to the next ACE. This process goes on untill there it reaches the implicit DENY ANY ANY ACE at the end of the ACL.
  - The thing to remember here is that , the ACL entries are taken up one-by-one and the packet in question is matched to it.
- The way Turbo ACL's work
  - As soon as you create the acl's and you type in " **access-list compliled ",** the IOS creates a lookup table which contains the ACL data. The table is built which contains the following coloums :

    TURBO TABLE

Field	Source IP (MS)	Source IP (LS)	Dest IP (MS)	Dest IP (LS)	IP Flags	L-3 Protocol field + L-4 Flags field	Source Port	Destination Port
Field Value	This contains the 16 bits of the source IP	This contains the LS 16 bits of the source IP	This contains the 16 bits of the dest IP	This contains the LS 16 bits of the dest IP	L-3 Flags	This is used to identify the L-4 protocol + the L-4 protocol flags	L-4 Source Port	L-4 Destination Port
Size (bits)	16	16	16	16	8	16	16	16

  - The green fields in the TURBO Table represents information found in the L3 Header of the packets. The red fields represent the information which will be present in the L-4 header of the packets.

  - Combined this table forms a formidable force which can be used to match any incoming packet to a particular ACE , and the ACE's action can be taken for that packet (permit, deny, log , etc)
- Now we know the table structure. So when we create the ACL's and when we type in "**access-list compiled**" , the following table is populated using the ACE entries.
  - Ex : Suppose we create 4 ACL's
    - **#access-list 101 deny tcp 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255 eq telnet**
    - **#access-list 101 permit tcp 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255 eq http**
    - **#access-list 101 deny tcp 192.168.1.0 0.0.0.255 192.168.3.0 0.0.0.255 eq http**
    - **#access-list 101 deny icmp 192.168.1.0 0.0.0.255 200.200.200.0 0.0.0.255**
  - The table gets populated as follows

    <u>EQUIVALENCE TABLE</u>

INDEX	Source IP (MS)	Source IP (LS)	Dest IP (MS)	Dest IP (LS)	IP Flags	L-3 Protocol field + L-4 Flags field	Source Port	Destination Port
0	192.168	1.0	192.168	2.0	*	TCP	*	23
1	192.168	1.0	192.168	2.0	*	TCP	*	80
2	192.168	1.0	192.168	2.0	*	TCP	*	80
3	192.168	1.0	200.200	2.0	*	ICMP	*	*

  - * reperesent fields which are not filled for that particular entry
  - Each row in the Table represents one ACE in the ACL 101 , in the exact order as configured
- Now that the entries are populated in the table, we can now look into another type of table called "Equivalence tables", which contain fields called "**Bitmaps**".
  - Each field in the TURBO Table (above) would have one equivalence table generated for it.
  - So totally there would be as 8 equivalence tables generated .
  - The Equivalence table for the Source IP (MS) field would look like this

Index	Value/Mask	ACL Entries in Bitmap
0	192.168/255.255	1 1 1 1

  - This says that, the source MS field contains only "1" UNIQUE IP address i.e 192.168 with a mask of 255.255 , and the BITMAP field value of "1 1 1 1" means that all the 4 ACE's have this IP as their SOURCE IP(MS). Suppose we encounter a case where only the first three ACE's had a source ip (MS) field equal to 192.168 , and the 4th ACE had a SOURCE IP(MS) field equal to 172.16, then the equivalence table would look like this

Index	Value/Mask	ACL Entries in Bitmap
0	192.168/255.255	1 1 1 0
1	172.16/255.255	0 0 0 1

  - I hope this is clear now . Understanding this is very crucial so please take your time in understanding this.
- Ok, we’ve got the 2 tables defined now , now it’s time to get into the protocol matching.
- TURBO ACL’s take a maximum of 5 lookups to match any packet!
- Suppose you have an ACL with about 500 ACE’s , while conventional ACL’s could take about 500 lookups to match a packet, TURBO ACE would take only 5. Isn’t that awesome.
- This lookup can reduce depending on the match condition (ex : protocol, etc) .But it would take a maximum or total of 5 lookups.
- At the end of the 5th lookup, we would have the ACE match .

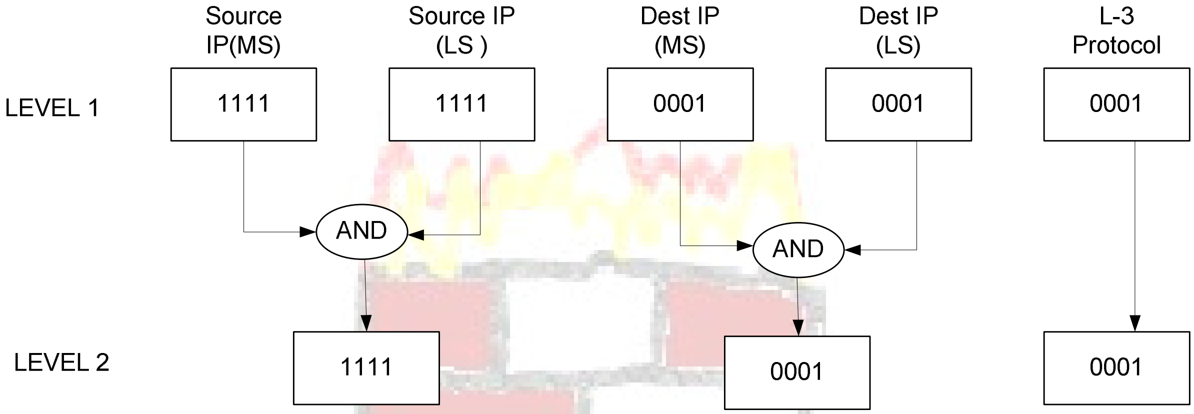
PROCEDURE

- The important thing to note here, at every level processing is done Paralelly . So all the activity per level happens at the same time, not sequentially. But each level is processed sequentially.
- Let’s get into the levels now.
- LEVEL 1 :
  - The equivalence tables of each field in the TURBO table is already built and present in the system memory
  - Now as soon as a packet arrives, the values corresponding to the TURBO TABLE fields (ex : Source IP MS, Source IP LS , Dest port , etc) are removed and are matched against the equivalence table of the particular table to get the Bitmap variable value.
  - Ex : Suppose we get a packet from source IP 192.168.1.0/24, after level one, the BITmap it would retrieve from the Equivalence table of the Source IP (MS) and the Source IP (LS) would be “ 1 1 1 1” , this is because the IP 192.168.1.0 appears in all the 4 ACE’s. (As indicated by the equivalence tables).
  - For all the fields that are defined in the TURBO table, the packet will get a list of Bitmaps
  - Ex : A packet comes in with the following characteristics
    - Source IP (MS) : 192.168
    - Source IP (LS) : 1.1
    - Destination IP (MS) : 200.200
    - Destination IP (LS) : 200.1
    - L-3 Protocol field + L-4 Flags : 0001 (ICMP)
  - After matching with the equivalence tables , it’s Level 1 output would look like this

	Source IP(MS)	Source IP (LS)	Dest IP (MS)	Dest IP (LS)	L-3 Protocol
LEVEL 1	1111	1111	0001	0001	0001

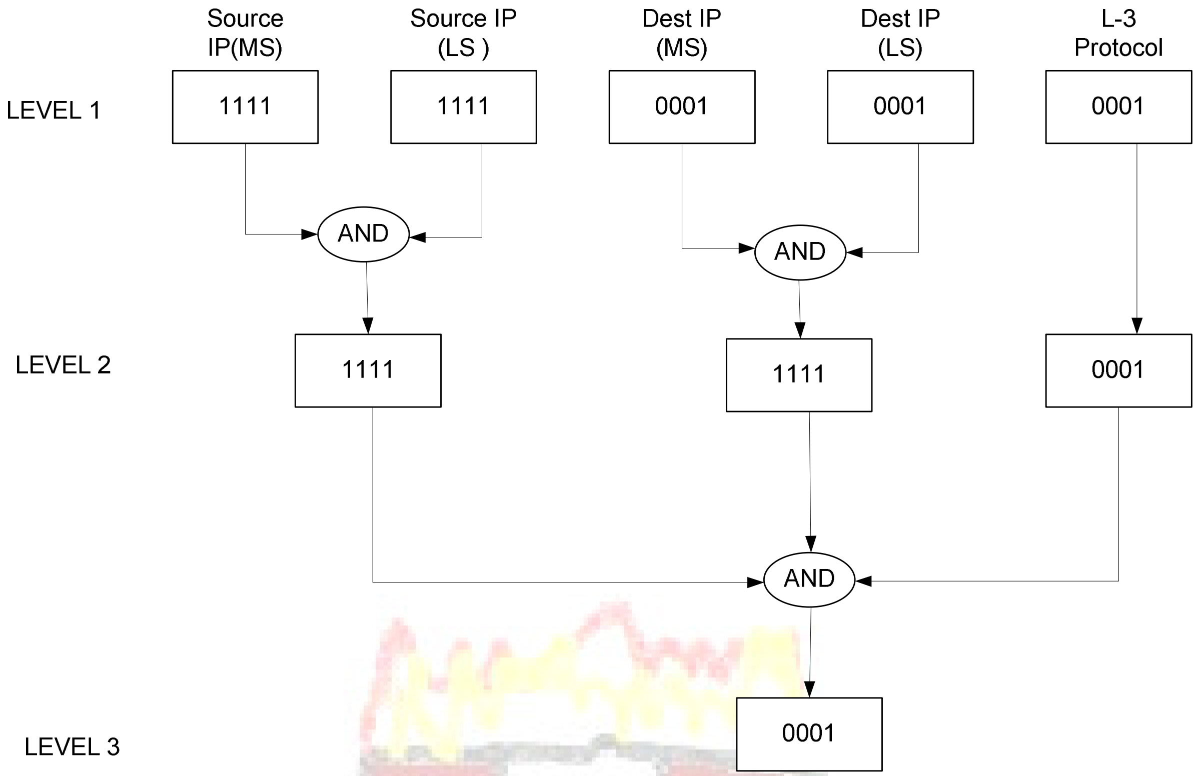
- Take your time to understand this. The Destination IP fields have a value of “0001” because the destination IP values of the incoming packet only is present in the 4th ACE. I hope I’m clear

- LEVEL 2 :
  - We now drill down to a much lower level .
  - We still notice that we have separate fields for the MS and the LS parts of the IP’s, and the Destination ports also.
  - So what we do in the next step is we “AND” the bitmap values which we have obtained as a result of Level 1.
  - The source IP blocks are ANDED , the destination IP blocks are ANDed and we get a much “NARROWED” down result after L2



  - Now because of this Processing and creating of Level 2 bitmaps, we now know that
    - the SOURCE IP of “192.168.1.0/24” is found in ACE entries 1,2,3 and 4
    - The DESTINATION IP of “200.200.200.0/24 “ is found only in ACE #4.
  - We still haven’t arrived on a match yet , so we will go down one more layer

- LEVEL 3 :
  - Here we AND the values of the SOURCE IP and the DESTINATION IP along with the L-3 protocol bitmap . So the result of this LEVEL 3 lookup, is we finally ARRIVE on the matching ACE entry.



- Now we finally come up with a BITMAP of “0001”, which means that our entry matches ACE # 4 , which if you look back is “**#access-list 101 deny icmp 192.168.1.0 0.0.0.255 200.200.200.0 0.0.0.255**”

TURBO ACL’s are inefficient if we are using a small number of ACL’s (~ 5 ACE’s), because for small numbers sequential matching is less processor intensive. But for large ACL’s (~ >10) which are being used in an ISP environment ,etc Turbo ACL’s save a lot of processor cycles.

The only disadvantage of turbo ACL’s is that additional space is required to store the TURBO and the EQUIVALENCE tables in the router memory.