Ternary Content Addressable Memory (TCAM)

In traditional routing, ACLs can match, filter, or control specific traffic. Access lists are made up of one or more access control entities (ACEs) or matching statements that are evaluated in sequential order. Evaluating an access list can take up additional time, adding to the latency of forwarding packets.

In multilayer switches, however, all of the matching process that ACLs provide is implemented in hardware. TCAM allows a packet to be evaluated against an entire access list in a single table lookup. Most switches have multiple TCAMs so that both inbound and outbound security and QoS ACLs can be evaluated simultaneously, or entirely in parallel with a Layer 2 or Layer 3 forwarding decision.

The Catalyst IOS Software has two components that are part of the TCAM operation:

- **Feature Manager (FM)**—After an access list has been created or configured, the Feature Manager software compiles, or merges, the ACEs into entries in the TCAM table. The TCAM can then be consulted at full frame forwarding speed.
- **Switching Database Manager (SDM)**—You can partition the TCAM on Catalyst switches into areas for different functions. The SDM software configures or tunes the TCAM partitions, if needed.

TCAM Structure

The TCAM is an extension of the CAM table concept. Recall that a CAM table takes in an index or key value (usually a MAC address) and looks up the resulting value (usually a switch port or VLAN ID). Table lookup is fast and always based on an exact key match consisting of two input values: 0 and 1 bits.

TCAM also uses a table lookup operation but is greatly enhanced to allow a more abstract operation. For example, binary values (0s and 1s) make up a key into the table, but a mask value is also used to decide which bits of the key are actually relevant. This effectively makes a key consisting of three input values: 0, 1, and X (don't care) bit values—a three-fold or *ternary* combination.

TCAM entries are composed of Value, Mask, and Result (VMR) combinations. Fields from frame or packet headers are fed into the TCAM, where they are matched against the value and mask pairs to yield a result. As a quick reference, these can be described as follows:

- **Values** are always 134-bit quantities, consisting of source and destination addresses and other relevant protocol information—all patterns to be matched. The information concatenated to form the value is dependent upon the type of access list, as shown in Table 3-2. Values in the TCAM come directly from any address, port, or other protocol information given in an ACE.

- **Masks** are also 134-bit quantities, in exactly the same format, or bit order, as the values. Masks select only the value bits of interest; a mask bit is set to exactly match a value bit, or not set for value bits that don't matter. The masks used in the TCAM stem from address or bit masks in ACEs.
- **Results** are numerical values that represent what action to take after the TCAM lookup occurrs. Where traditional access lists offer only a *permit* or *deny* result, TCAM lookups offer a number of possible results or actions. For example, the result can be a permit or deny decision, an index value to a QoS policer, a pointer to a next-hop routing table, and so on.

Table 3-2 *TCAM Value Pattern Components*

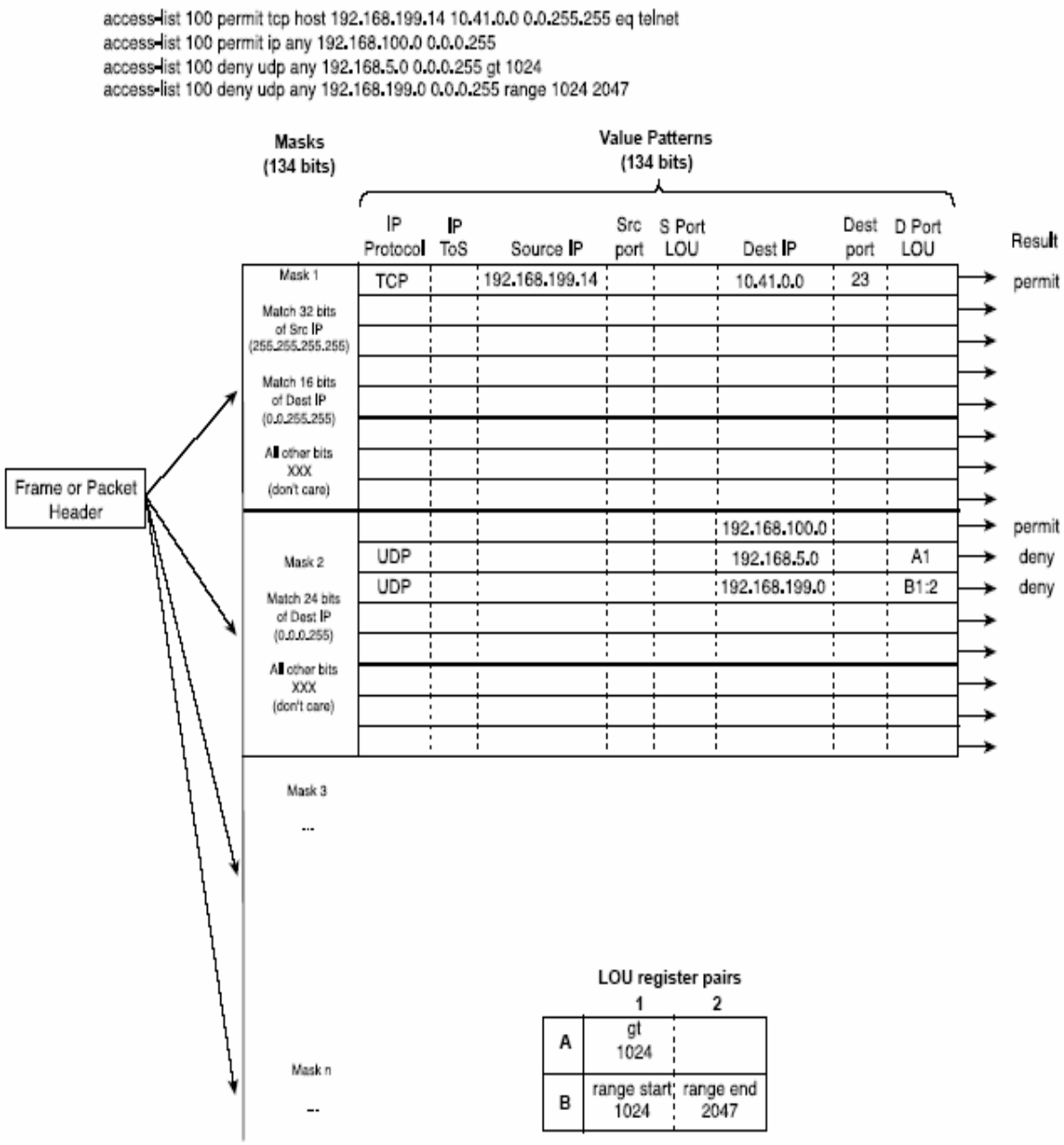
Access List Type	Value and Mask Components, 134 Bits Wide (Number of Bits)
Ethernet	Source MAC (48), destination MAC (48), Ethertype (16)
ICMP	Source IP (32), destination IP (32), protocol (16), ICMP code (8), ICMP type (4), IP type of service (ToS) (8)
Extended IP using TCP/UDP	Source IP (32), destination IP (32), protocol (16), IP ToS (8), source port (16), source operator (4), destination port (16), destination operator (4)
Other IP	Source IP (32), destination IP (32), protocol (16), IP ToS (8)
IGMP	Source IP (32), destination IP (32), protocol (16), IP ToS (8), IGMP message type (8)
IPX	Source IPX network (32), destination IPX network (32), destination node (48), IPX packet type (16)

The TCAM is always organized by masks, where each unique mask has eight value patterns associated with it. For example, the Catalyst 6500 TCAM (one for security ACLs and one for QoS ACLs) holds up to 4096 masks and 32,768 value patterns. The trick is that each of the mask-value pairs is evaluated *simultaneously,* or in parallel, revealing the best or longest match in a single table lookup.

TCAM Example

Figure 3-5 shows how the TCAM is built and used. This is a simple example, and might or might not be identical to the results that the Feature Manager produces. This is because the ACEs might need to be optimized or rewritten to achieve certain TCAM algorithm requirements.

Figure 3-5 *How an Access List Is Merged into TCAM*



The example access list 100 (extended IP) is configured and merged into TCAM entries. First, the mask values must be identified in the access list. When an address value and a corresponding address mask are specified in an ACE, those mask bits must be set for matching. All other mask bits can remain in the "don't care" state. The access list contains only three unique masks: one that matches all 32 bits of the source IP address (found with an address mask of 255.255.255.255 or the keyword **host**), one that matches 16 bits of the destination address (found with an address mask of

0.0.255.255), and one that matches only 24 bits of the destination address (found with an address mask of 0.0.0.255). The keyword **any** in the ACEs means match anything or "don't care."

The unique masks are placed into the TCAM. Then, for each mask, all possible value patterns are identified. For example, a 32-bit source IP mask (Mask 1) can be found only in ACEs with a source IP address of 192.168.199.14 and a destination of 10.41.0.0. (The rest of Mask 1 is the destination address mask 0.0.255.255.) Those address values are placed into the first value pattern slot associated with Mask 1. Mask 2 has three value patterns: destination addresses 192.168.100.0, 192.168.5.0, and 192.168.199.0. Each of these is placed in the three pattern positions of Mask 2. This process continues until all ACEs have been merged.

When a mask's eighth pattern position has been filled, the next pattern with the same mask must be placed under a new mask. A bit of a balancing act occurs to try and fit all ACEs into the available mask and pattern entries without an overflow.