Intel® Ethernet Switch FM6000 Series - Software Defined Networking

Recep Ozdag
Intel Corporation

Software Defined Networking (SDN) promises to provide a flexible and simple management approach to traditional, heavily protocol driven networks by separating the control and data forwarding planes and abstracting the control and management functions into a software based controller that presents a centralized view of the network. OpenFlow, which is one method to enable SDN, provides software control of the flow tables that instruct switches in the network how do direct traffic within a network. While both SDN and OpenFlow provide real benefits to users, the move from traditional networking will not happen overnight. In this paper we present Intel's low latency, high performance and extremely programmable, 10GbE FM6000/FM7000 Ethernet switch silicon as a key enabler for network administrators to gradually migrate to OpenFlow and SDN. The FM6000 series hybrid Ethernet switch provides superb support for OpenFlow and SDN with its unique FlexPipe® frame processing pipeline architecture while it continues its role as a traditional data center switching solution.

SDN

Server and network infrastructure virtualization are of tremendous help to data center administrators, who have to address increased demand for computing resources on a constrained budget. However, the increased growth of both physical and virtual devices in the data center as well as the demand to seamlessly move virtual machines (VM) also results in added complexity for these same administrators who manage dynamic networks that need to quickly adapt to user needs. Unfortunately, management and control of these virtualized and complex networks have not kept pace with the changes in the data center and largely remain unchanged.

Traditional networks rely on IP addresses to locate servers and each distributed switch device relies on one or more instances of a L2/L3 control plane. Traditional switches and routers using legacy networking protocols can take too long to converge for today's networks, which need to be faster and more flexible. Traditional networking methods and protocols were acceptable for yesterday's static networks. But managing complex and dynamic virtual networks has become extremely labor-intensive, far too time consuming and expensive to remain feasible and competitive in modern networks.

Dominated by virtual device mobility and multi-tenancy, modern networks need to quickly adapt to business needs. Furthermore, traditional data center networks heavily borrow from the enterprise with complex network protocols that have not been concerned with scalability or server migration.

Administrators need to be freed from the physical infrastructure to simplify the task of managing large and virtualized networks. Software Defined Networking (SDN) is a new networking paradigm that separates, abstracts and centralizes the control information of the network from the underlying distributed data forwarding infrastructure. This centralized view of the network, as illustrated in the figure below, combined with splitting the control and data planes creates a far more dynamic, flexible, automated and manageable architecture that also results in increased reliability and reduced costs. SDN provides the opportunity for the networks to be managed through intelligent orchestration software that supports virtualized networking and on-demand resource allocation. SDN provides a global network wide data flow control to the administrator, allowing administrators to define network flows that meet the requirements of end users. The SDN controller, which implements the control plane functionality, logically centralizes and

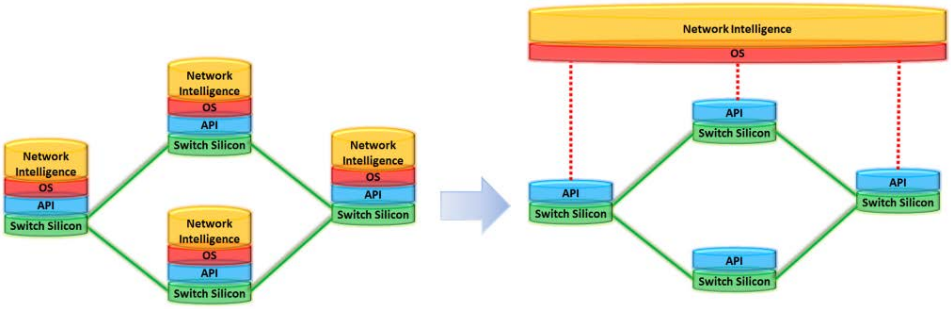

Figure 1: SDN provides a centralized view of the network

manages switches and routes packets through the network.

Some of the high level goals of SDN include the following:

- Flexible control of flows passing through the network
  - Forwarding of packets according to software defined rules
  - Load balancing of packets according to software defined heuristics combined with hardware flow hashing
  - Flexible modification of the frame (NAT, Tunneling, tag rewrites) to aid in interfacing between hosts, local networks, and external network interfaces
  - Intelligent flow classification (security, virtual domaining, etc.) that is software defined and is done in parallel with forwarding and frame modifications
- Vendor independent interface to the switching elements
- Ability to work beside existing protocols (hybrid switch)
- Ability to overlay SDN intelligence using tunneling

OpenFlow

The OpenFlow protocol is one method to enable SDN. OpenFlow provides software control of the flow tables that instruct switches how do handle traffic within an SDN based network. OpenFlow provides access to the data plane of the network and allows software to determine the path that data packets or flows will take.

OpenFlow depends on switches with internal flow tables and an interface to add, remove and manipulate flow entries that can be controlled via software running on an external and decoupled control plane. The controller and the switch communicate via the OpenFlow protocol. OpenFlow uses a number of packet header fields to define a flow. Each entry in the flow table contains a set of packet fields against which incoming traffic is matched and an associated action is performed on the matched flow. When the switch encounters a flow that it cannot match, the packet is sent to the controller to determine how to handle the packet. The controller may define a new flow and a new action based on this unmatched packet, and populate the switch's flow table accordingly.

While OpenFlow provides a new networking paradigm that is gaining traction, it must co-exist with traditional networks - the transition to SDN or OpenFlow will not happen overnight. For the transition to be smooth, OpenFlow must be supported by hybrid switches that also support traditional L2/L3 switching and IP routing.

The Intel® Ethernet FM6000 Series Hybrid Switch

The Intel® Ethernet FM6000 series hybrid switch provides the most practical way to transition to a flow-processing based OpenFlow type protocol while also supporting traditional switching and routing protocols. The highly programmable parser of the FM6000 allows for incoming traffic to be parsed and to be directed to the OpenFlow processing pipeline or to the traditional pipeline. This enables an infrastructure in which both traditional networks and OpenFlow can be simultaneously supported with extremely low latency and high performance.

Programmable Parsing

The Intel® Ethernet FM6000 series switch provides unprecedented parsing flexibility to support the ever growing portfolio of networking protocols, making the switch an ideal building block for administrators that want to define and introduce their own protocols and standards. The key building block that provides such programmability while maintaining line rate performance is a TCAM/SRAM/MUX structure as illustrated in Figure 2.

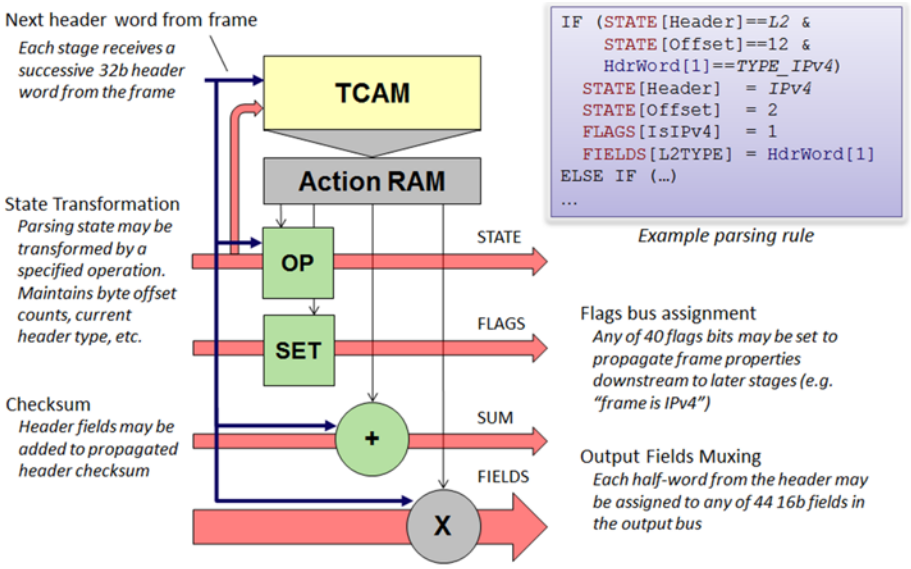

Figure 2: TCAM/RAM/MUX structure used in the parser

Incoming frame header fields are looked up in the TCAM, which allows partial matches with the usage of wildcards. Matches in the TCAM point to an entry in the RAM, which controls various operations to be performed in the incoming frame header fields as well as extracting and propagating these fields to subsequent stages.

The FM6000's loop unrolled and fully pipelined parsing state machine is illustrated in Figure 3. The parser processes each frames incoming byte sequence according to programmed rules, mapping the frames conditionally formatted header fields into fixed hardware channels. At the hardware level, very little of this flexible functionality is specific to Ethernet or any particular standard supported by the device. The parser can be viewed as an iterative state machine that consumes successive four byte words of the frame in each iteration. In response to incoming frame contents and its internal state, the parser maps the frame data to specific fields of an output bus that ripples through the frame processing pipeline. In addition, the parser records properties of interest about the frame such as whether the frame is a unicast, multicast, IP frame, by setting specific bits of a 40-bit output flags vector.

In order to maintain a fully pipelined operation, the parser implementation is physically unrolled in hardware, with each *Parser Slice* representing one iteration of the parsing state machine as shown above. Each successive slice receives the next word of frame data and the prior slices' 32-bit state output. These

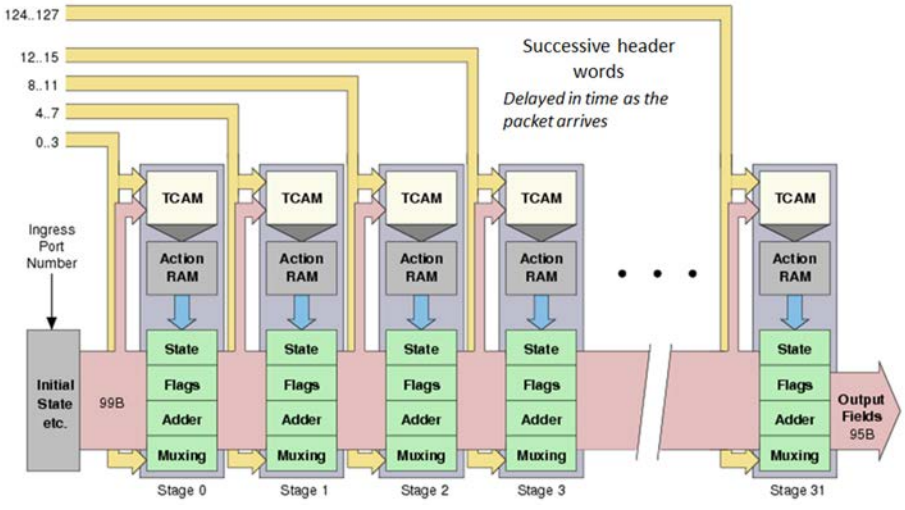

Figure 3: FM6000's programmable and pipelined parser

two 32-bit quantities are combined into a key that is looked up in the TCAM to determine an action specifying how to update the state vector, what flags to set and how the frame data should be extracted and mapped to the output channel. This very powerful structure provides the maximum flexibility in detecting and parsing ever changing Ethernet standards and protocols providing a future proof investment.

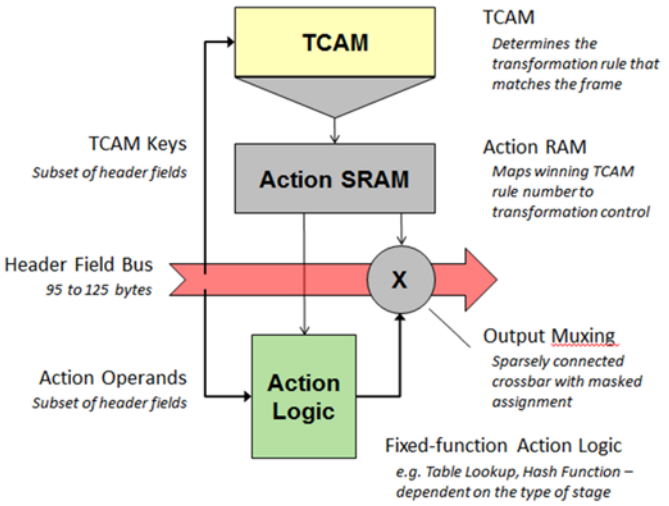

Figure 4: TCAM/RAM/LOGIC structure

The Frame Forwarding Unit and Pattern Matching

The Intel® Ethernet FM6000 series switch provides a wide array of pattern matching capabilities to support traditional networking, SDN and OpenFlow. The key building block of the Frame Forwarding Unit (FFU) that provides such flexibility and high performance is a TCAM/SRAM/LOGIC structure that is illustrated in Figure 4.

A subset of the original or modified frame header that is passed along from the Parser to the FFU is fed into the TCAM that determines the transformation rules that are to be applied to the frame. The winning entry in the TCAM points to the corresponding action that must be applied among all the possible actions that are stored in the Action RAM. The Action Logic takes both the input header fields and data from the Action RAM as operands to perform a particular action. This circuit structure provides a very powerful mechanism to selectively match a portion of a field then select an action to be applied out of a set of actions, and can be pipelined to provide a very flexible pattern matching construct to be used for access control lists (ACLs) or matching to the widest aggregate field dictated by the latest OpenFlow specification. The resulting FFU structure is illustrated in Figure 5.

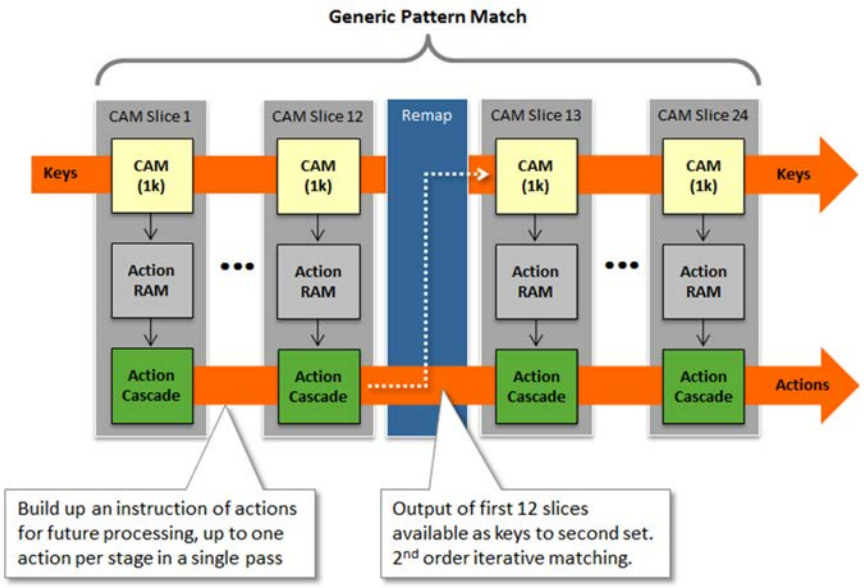

Figure 5: FM6000's FFU pipeline used for generic pattern matching

Once the incoming packet is parsed, the fields are mapped on to keys to be fed into the FFU that consists of 24 slices with each slice capable of matching 1K entries, providing a total of 24K TCAM entries, with each entry being 36-bits. Slices can be programmed to match and act on 36-bit keys or can be grouped together to form a much wider

entry to match on wider keys, such as OpenFlow's 12-tuple construct. Furthermore, the outputs of the first 12 slices can be mapped as keys to the input of the second 12 slices providing recursive flow forwarding with no performance degradation penalty as illustrated in Figure 5.

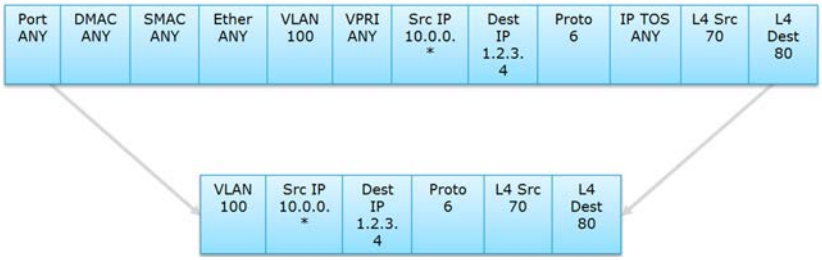

Figure 6: Rule compression to efficiently use TCAMs

A Binary Search Tree (BST) for IP Lookup

The Intel® Ethernet FM6000 series switch supports both IPv4 and IPv6 routing. While IP addresses can be looked up using the TCAMs available in the FFU, the FM6000 series also provides a more efficient longest prefix search based Binary Search Tree (BST) block that can match a key with up to 128-bits with capacity to compare up the 64K rules against any frame. Because of the flexible TCAM based structure, the BST can be used to lookup both IP addresses in a traditional networks as well as provide pattern matching capabilities for OpenFlow based usage.

Rule Compression

To support hundreds of thousands of flows that carriers and massive data centers may require, the FM6000 provides a "compression" service at the API level. This service compresses and maps these flows to rules to populate the TCAMs and is orders of magnitude more efficient than simply using the flow fields without modification. This powerful compression allows the

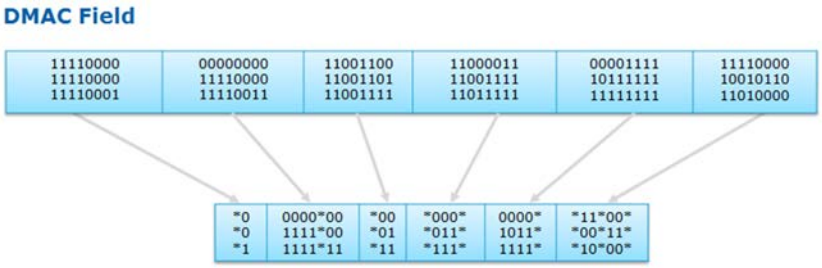

Figure 7: Compressing a single field

finite, on-chip TCAMs to be utilized very efficiently supporting most customers' needs while providing a combination of 10GbE and 40GbE throughput.

Consider a flow that is defined using the 12 fields in Figure 6. An OpenFlow controller that adds this flow, without optimizing it will use up a very large TCAM entry to fit all the fields. However, it is clear that 6 fields are not involved in defining the flow, thus, such an approach will waste valuable resources. Intel's OpenFlow compatible switch will compress this rule by removing the non-identifying information and will compact the flow identifier into a very narrow TCAM entry. Even if all the fields were used, compression is still possible.

Consider the DMAC field illustrated in Figure 7. Three flows with different DMAC addresses are intended to be added to the switch's flow table. Rather than using all 48-bits, Intel's compression API can reduce the field width down to 20-bits, with no loss of information or functionality.

Multi-Stage Architecture

OpenFlow heavily relies on matching packets against multiple tables in the pipeline, with the output of one table being able to modify the contents of the inputs to the next table as illustrated in Figure 8. Not only are modified or unmodified packets sent from one table to the next, but metadata is also generated and passed on between tables. This enables a wide range of actions that are

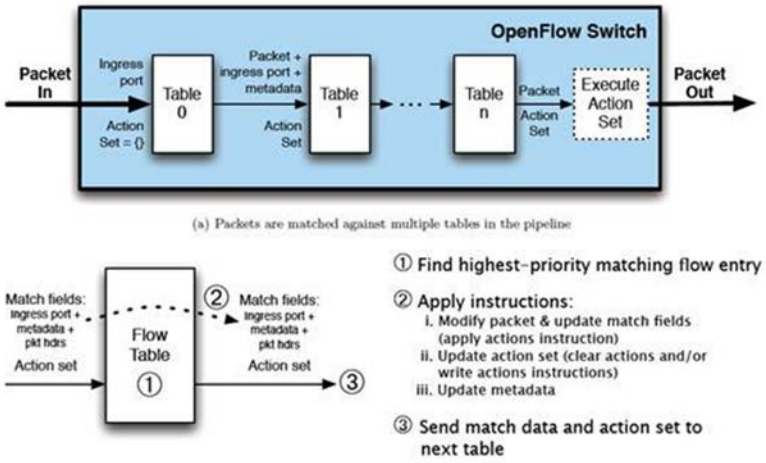

Figure 8: OpenFlow multi-state pattern matching

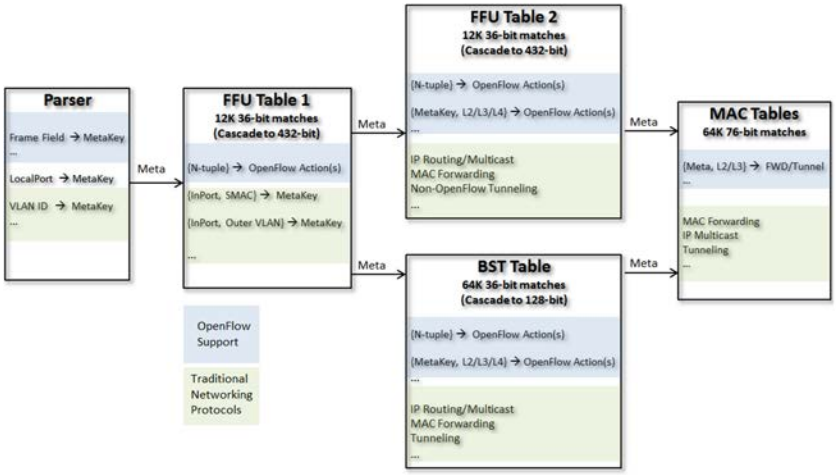

Figure 9: FM6000's multi-stage FlexPipe® architecture

accumulated and executed on the packet at the end of the pipeline before the frame leaves the switch.

The Intel® Ethernet FM6000 series switch provides a similar and very powerful multi-stage architecture. Not only does the FFU consist of two blocks with a remap stage between them but the FM6000's microcode also allows the hybrid switch to reallocate its Parser, MAC and Binary Search Tree (BST) tables from legacy protocols such as support for IPv4, IPv6, IP Multicast and tunneling to flow matching as illustrated in Figure 9.

The FM6000's forwarding tables can be used to support OpenFlow and traditional networking protocols simultaneously at full line rate.

Beyond SDN and OpenFlow

While SDN and OpenFlow are making great strides in addressing the needs of today's data center and continue to evolve in the right direction, they are still far from providing complete solutions. In this section we provide a few use cases in which various protocols and features are not necessarily fully supported by these emerging concepts but can be easily addressed with the Intel® Ethernet FM6000 and FM7000 series switches. The Intel FM7000 Ethernet switch builds upon the FM6000 features to not only support SDN, but also several advanced tunneling protocols.

NAT Processing

Network Address Translation (NAT) allows seamless connection between local IP addresses in the private network and global IP addresses in the public network. In order to maintain several concurrent connections between local computers and outside global IP addresses, NAT uses the following header fields.

- IPv4 source address
- TCP or UDP source port number
- IPv4 destination address
- TCP or UDP destination port number

This section provides a high-level overview of how these fields are

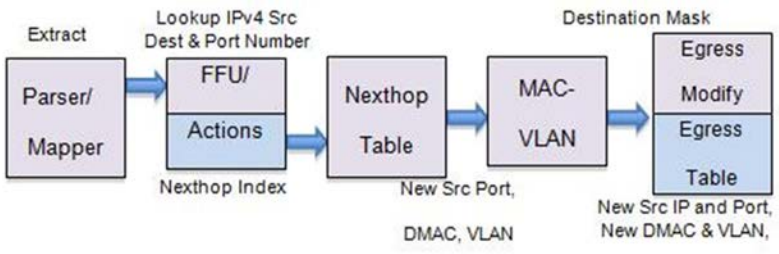

Figure 10: NAT processing into the public network using the Intel FM7000 switch

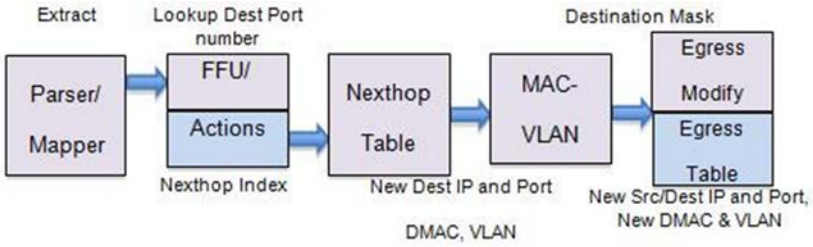

Figure 11: Intel FM7000 switch NAT processing from the public network

translated as traffic moves in and out of the private network and how NAT is supported in Intel FM7000's FlexPipe® frame processing pipeline.

Outgoing Traffic

Figure 10 shows how frames are processed as they transition from the private network into the public network in a NAT router, such as the FM7000.

In this direction, the FFU/BST match various IPv4 fields of the flow to create an index into the Nexthop table. The Nexthop table provides the normal DMAC, VLAN used for forwarding, plus any additional information that will be used to process response packets as they are returned from the destination. Egress Modify is a programmable unit that updates the DMAC and VLAN fields, while replacing the SMAC with the address of the router.

Incoming Traffic

Figure 11 shows how frames are processed as they transition from the public network into the private network in a NAT router.

In this direction, the Egress Modify unit updates the DMAC and VLAN, while replacing the SMAC with the address of the router.

Load Balancing

In applications such as server load balancing, the Intel FM6000's and FM7000's ECMP feature can be used to distribute loads across private network IPv4 addresses after translation. The FM6000 and FM7000 contain an L3 hashing mechanism that can use various L3/L4 header fields as a hash key as shown in the following table.

Field	Bytes	Masking Support
L3_SIP	15:0	Per-byte
L3_DIP	31:16	Per-byte
QOS.L3_PRI	32	Per-bit
L3_FLOW	35:33	Per-bit
ISL_USER	36	Per-bit
L3_PROT	37	Per-bit
L4_SRC	39:38	Per-bit
L4_DST	41:40	Per-bit
FIELD16{A..D}	49:42	Per-byte
RANDOM	55:50	Per-byte

Three 16-bit hash values are produced that are statistically independent allowing support of applications such as hierarchical ECMP. Symmetrical hashing is also supported. The L3 hashing of the FM6000/FM7000 switch allows the balancing of flows across multiple entries.

IP Tunneling

The Intel® Ethernet FM7000 series hybrid switch provides the most practical way to move to IP-in-IP tunneling. At the switch ingress, the outer and inner IP headers are presented to the FFU TCAM and routing tables. The microcode can be configured to either store the outer IP header in the standard IP header fields (Destination IP, Source IP, Protocol, etc), or store the outer header in the header expansion fields when a tunnel header is detected. At the switch egress, a tunnel header is added to the frame's IP header.

IP-in-IPv4 Encapsulation

Figure 12 illustrates a high-level view of how the frame-processing pipeline can be configured to support IP-in-IPv4 encapsulation using the Nexthop table. In this case, the parser will present the

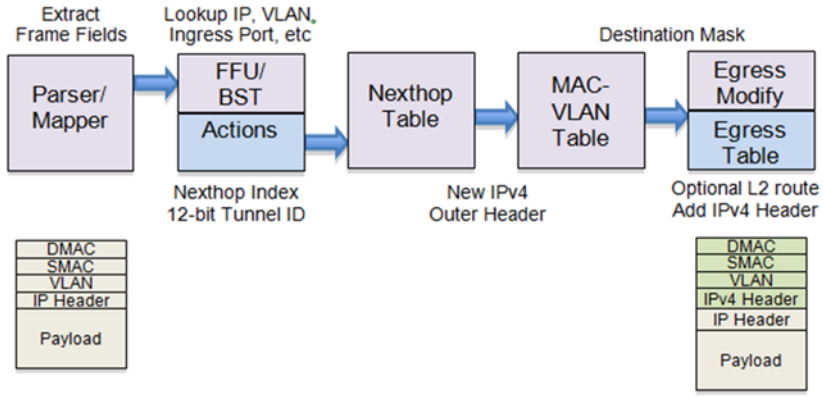

Figure 12: IP-in-IP Encapsulation in the FM7000 series Ethernet switch

frame fields to the ACL and routing tables in the FFU/BST. As a result of a frame match in these tables, the Nexthop table may specify that the frame should be IP-in-IPv4 encapsulated or de-encapsulated.

IP-in-IPv6 Encapsulation

Encapsulating for IPv6 works similarly to adding an outer IPv4 header, with the exception that up to 96 bits of encapsulating IPv6 destination IP address is sourced from the Egress Modify tunnel ID table. The pipeline treats this tunneling similarly to adding an IPv4 header.

Conclusion

New technologies such as SDN and OpenFlow meet the goal of helping users develop applications that manage and control the network independent of the underlying topology or switch vendor while providing access to resources on a per-flow basis. The Intel® Ethernet FM6000 series and FM7000 series switch silicon provides full support for these technologies while filling the gap that these specifications lack for supporting IPv6, NAT, load balancing, QoS, traffic shaping and many more fundamental features in today's data center. The FM6000/FM7000 provides the perfect mix of programmability with its FlexPipe® frame processing pipeline and support for traditional networking, allowing users to support both of these old and new technologies at the same time.

For more information on Intel® Open Network Platform, visit **www.intel.com/go/ethernet**