Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches

Paul T. Congdon, Prasant Mohapatra, Matthew Farrens and Venkatesh Akella

***Abstract*—The Ethernet switch is a primary building block for today's enterprise networks and data centers. As network technologies converge upon a single Ethernet fabric, there is ongoing pressure to improve the performance and efficiency of the switch while maintaining flexibility and a rich set of packet processing features. The OpenFlow [1] architecture aims to provide flexibility and programmable packet processing to meet these converging needs. Of the many ways to create an OpenFlow switch, a popular choice is to make heavy use of Ternary Content Addressable Memories (TCAMs) [2]. Unfortunately, TCAMs can consume a considerable amount of power, and when used to match flows in an OpenFlow switch, put a bound on switch latency.**

In this paper we propose enhancing an OpenFlow Ethernet switch with per-port packet prediction circuitry in order to simultaneously reduce latency and power consumption without sacrificing rich policy-based forwarding enabled by the OpenFlow architecture. Packet prediction exploits the temporal locality in network communications to predict the flow classification of incoming packets. When predictions are correct, latency can be reduced, and significant power savings can be achieved from bypassing the full lookup process. Simulation studies using actual network traces indicate that correct prediction rates of 97% are achievable using only a small amount of prediction circuitry per port. These studies also show that prediction circuitry can help reduce the power consumed by a lookup process that includes a TCAM by 92% and simultaneously reduce the latency of a cut-through switch by 66%.

***Index Terms*—Ethernet networks, packet switching, software defined networking**

I. Introduction

ETHERNET and IP communications have become the most popular means of computer communications. As Ethernet moves beyond 10 Gbps speeds and prices continue to decline, there is a strong desire to use this commodity technology in as many diverse computing environments as possible. Unfortunately each environment demands a different set of requirements and a one-size-fits-all approach (to packet forwarding) does not adapt to all needs. The OpenFlow architecture for Ethernet switches [1, 3] is receiving considerable attention within the research community as a means for catalyzing the exploration of new applications for Ethernet switches. OpenFlow allows the forwarding plane of switches to be reprogrammed by an external controller, making it possible to adapt traditional Ethernet switching to different environments. However, certain environments have specific requirements, such as extreme low latency and reduced power consumption, which cannot be completely addressed through simple reprogramming.

Methods for implementing fast and efficient lookups have been extensively studied [4-6]. Traditional switching chips often incorporate multiple tables to support simple layer-2 and layer-3 lookups using address hashing and various data structures stored in SRAM. OpenFlow switches leverage these tables for basic forwarding, but when processing rules related to a specific flow, they delay forwarding until a packet flow identifier can be assembled and presented to a classification engine. The classification engine is often implemented using a Ternary Content Addressable Memory (TCAM) [2, 7, 8]. TCAMs are popular because they allow the switch to support a rich set of policy based features while maintaining line rate forwarding across a large set of 10 or 100 Gbps Ethernet ports. TCAMs, however, are expensive, power hungry devices, and their power consumption increases linearly with the size of the structure; as much as 25% of the total power required for a switching ASIC (see Fig. 5).

Many environments for this new class of OpenFlow enabled Ethernet switch demand the lowest possible latency in addition to rich packet processing. Unfortunately these two objectives can be at odds with one another. Packet classification (to at least the flow level) enables rich features such as firewalling, intrusion prevention, connection rate metering and load balancing. OpenFlow switches that combine flow-based packet classifiers with TCAMs have the flexibility to support these features. However, to meet strict latency requirements, network managers often avoid these features and deploy customized equipment with specialized cut-through designs. Cut-through switches provide low latency by allowing a packet to begin transmission on an egress port before the packet has been completely received at the ingress port. This approach is not as effective on an OpenFlow switch that must wait for multiple protocol headers

Manuscript received July 9, 2011.
P. T. Congdon is with Tallac Networks, Rocklin, CA 95765 USA (phone: 916-765-4056; e-mail: paul.congdon@tallac.com).
P. Mohapatra is with the Department of Computer Science, University of California Davis, Davis, CA 95616 USA (e-mail: prasant@cs.ucdavis.edu).
M. K. Farrens is with the Department of Computer Science, University of California Davis, Davis, CA 95616 USA (e-mail: farrens@cs.ucdavis.edu).
V. Akella is with the Department of Electrical and Computer Engineering, University of California Davis, Davis, CA 95616 USA (e-mail: akella@ucdavis.edu)

to arrive in order to assemble a flow identifier. There is simply no time in a cut-through switch for a rich set of features to inspect transport or application level fields before the packet is switched. Two current state-of-the-art low latency Ethernet switches from Arista Networks [9] and Cisco Systems [10] achieve latencies of 350ns and 190ns respectively. While these latencies are impressive, they are still nearly three times the latency of a minimum size Ethernet packet at 10Gbps.

This paper proposes enhancements to simultaneously reduce both switch power consumption and switch latency, beyond that of cut-through switches, while maintaining the flexibility and rich policy based forwarding of an OpenFlow switch. The enhancements apply the architectural techniques of value prediction and speculative execution to the problem of packet switching by exploiting the temporal locality of network traffic. Unique per-port circuitry predicts the flow membership of a packet in order to bypass traditional flow-based or TCAM lookups, and when possible, begin speculative forwarding, thus simultaneously reducing both switch latency and power consumption.

We demonstrate the effectiveness of the techniques by simulating an enhanced OpenFlow switch using a variety of workloads captured in actual network traces from different parts of the network topology. The results show that correct prediction rates of 97% are achievable and that it is possible to reduce the power consumption of flow-based lookups between 92% and 98% depending upon the size of a TCAM. The results also show that latency can be reduced by a factor of 3 over a conventional cut-through switch.

This paper makes the following contributions:

1) Improves upon the OpenFlow switch architecture by providing a per-port optimization that simultaneously reduces both switch latency and power consumption.
2) Provides a detailed architectural-level power model for a flow-based Ethernet switch.
3) Provides a latency model for a flow-based Ethernet switch that uses prediction to speculate on forwarding operations.
4) Defines low-latency methods of predicting flow membership for packets as they are received.

The remainder of this paper is organized as follows. Section II describes the switch model used and a description of the prediction circuitry, while Section III describes how bypassing the TCAM can reduce power and Section IV explains how speculative forward can reduce latency. Section V discusses the evaluation environment. Section VI describes the results of those simulations. Section VII describes related work and Section VIII provides the conclusion.

II. Switch Architecture

A. *Switch Architecture*

There are many ways to create an Ethernet switch [11]. The OpenFlow Ethernet switch architecture used in this paper assumes line cards with physical media ports connected to a switched backplane fabric. The architecture is similar to one of many described in [11], but with an emphasis on flow-based switching where the logic is implemented in a single chip on the line card and takes advantage of a TCAM for flow matching. The line cards are equipped with separate input and output memory, lookup logic, backplane fabric interfaces and per-port prediction circuitry. The lookup and policy logic may involve multiple layer-2 and layer-3 address tables in addition to a TCAM used to support per-flow forwarding features. Fig. 1 shows the high-level switch architecture.

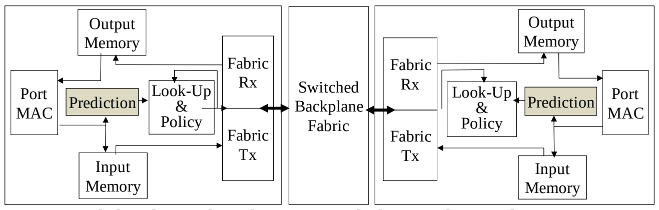

Fig. 1. High-level switch architecture including packet prediction circuitry at the input port

Packets are received at line rate from the physical media ports and put into the input memory. While the packet is being received into memory, a block of combinational logic within the Media Access Controller (MAC), called the packet parser, is extracting important fields from the packet to generate a flow-key. A flow-key is the fundamental structure used to lookup and determine how to forward the packet according to OpenFlow rules. The flow-key and the forwarding architecture of the switch are consistent with the definition of a “Type 0” and “Type 1” OpenFlow switch [3].

A flow-key is a concatenation of critical fields from the packet that uniquely identify the packet as being part of a flow. It can be generalized as an n-tuple that is defined by a set H = {H_1, H_2, …, H_n} of fields from the packet. All packets that are part of a flow are subject to the same policy and treatment by the switch. For a typical routing switch that performs layer-2 bridging, layer-3 routing and transport level filtering, a flow-key can be represented by a 9-tuple that includes the following fields: VLAN ID, destination MAC address, source MAC address, ethertype, IP protocol number, source IP address, destination IP address, TCP/UDP source port number and TCP/UDP destination port number. A flow-key, in a large OpenFlow switch, includes an additional field that indicates the source port of the received packet.

A flow table is a large database of flow-keys that is searched by the lookup process in addition to traditional layer-2 and layer-3 tables. This structure may be implemented in software using SRAM and a fast network processor, or more often, implemented in hardware by a TCAM. TCAMs are an expensive, power hungry, high performance resource that allows rapid wildcard searching. The TCAMs may be shared by multiple input ports on the same line card, and consequentially, may be subject to contention and further arbitration delays. The process of classifying the packet and searching the flow table has been well studied [12, 13], and is known to be a time consuming and critical stage in the switch pipeline.

The results of the lookup steps tell the switch where to forward the packet across the switch fabric, and optionally,

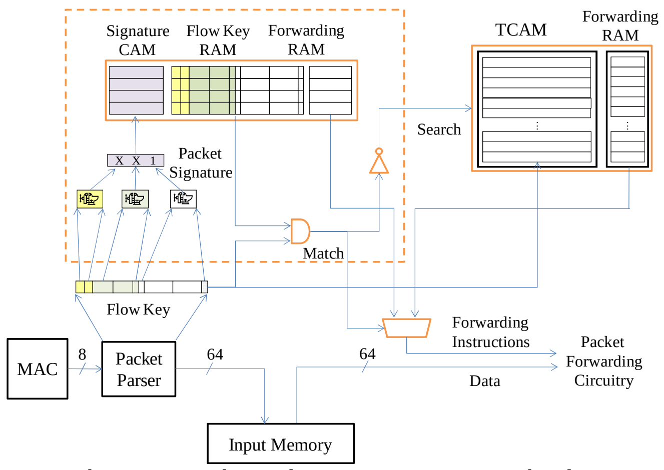

Fig. 2. The per-port packet prediction circuitry snoops packet data as it is received by the input MAC to create a compressed packet signature and a flow-key for validation

what modifications to the packet may be required. In our switch model, the receiving line card makes all necessary modifications to the packet and initiates a transfer of the packet across the fabric to the output memory on another line card. Alternative switch architectures may make modifications to the packet at the egress port or other parts of the pipeline. The speed of the backplane fabric interface is usually faster than the speed of the input port and in our model the backplane does not represent a bottleneck. Once the packet is received in the output memory it may immediately begin transmission on the egress port.

In heavily loaded networks, packets may experience queuing delays at the ingress and/or egress memory depending upon the backplane transfer protocol and the aggregation of traffic from multiple ingress ports. This queuing will reduce the impact of our latency improvements; however power savings can still be achieved. Under heavy load, even state-of-the-art cut-through switches [9, 10] are unable to further reduce latency due to queuing delays. In latency sensitive environments, techniques such as congestion control [14] and adaptive routing [15] are used to manage latency and mitigate contention at hotspots across the network. These approaches can be combined with uncongested individual switch latency improvements, such as the one proposed here, to establish the lowest possible switch latency.

B. Packet Prediction Circuitry

The switch architecture of Fig. 1 includes packet prediction circuitry on each ingress port. The goal of the circuitry is to predict the flow-key of a received packet as quickly as possible, without requiring the use of the lookup process and TCAM. Fig. 2 shows how a packet signature is created by circuitry that snoops on the input memory bus as the packet is streamed into memory. While the flow-key is being assembled, significant bits from the packet are extracted and used to generate a packet signature according to a prediction method (see II.C). The packet signature is searched in a local per-port prediction cache that is comprised of a signature CAM, flow-key RAM and forwarding RAM. If exactly one match is found, the packet is assumed to be part of the same flow as a previously received packet with the same signature and forwarding can begin immediately. This constitutes speculative packet forwarding which reduces switch latency. Such forwarding by the prediction circuitry involves applying the same set of packet operations that would have been obtained from the full lookup process. Since the prediction circuitry is required on each input port of the switch, it is worthwhile to find the smallest and most efficient implementation possible.

The flow-key RAM in Fig. 2 holds the flow-key for the most recent packet that has a matching signature. The flow-key in the RAM is compared against the flow-key that has been assembled by the packet parser in order to confirm if there has been a match. This comparison is necessary to avoid invoking the full lookup process on every packet, thus saving power. The flow-key RAM is fundamentally a level 1 cache for the master TCAM, referencing the most recent forwarding instructions for packets of the matching flow.

Once a complete flow-key has been received and assembled by the packet parser, there are three possible scenarios that may occur with respect to the prediction logic and speculative forwarding:

1) Prediction Hit: The flow-key matches a flow-key found by the prediction logic. In this case, a correct prediction has occurred and there is no need to take any further action. No lookup or search is required of the master TCAM and the power required to perform that search is saved. The lowest possible latency is achieved because packet forwarding has already started and the speculation was correct.
2) Incorrect Prediction: A signature is found, but the flow-key does not match. In this case, an incorrect prediction has occurred and the current speculative transfer must be aborted. The master TCAM must be searched to determine the correct forwarding instructions and the local prediction cache must be updated. The power for the prediction cache searches and the partial packet transfer is wasted.
3) Prediction Miss: No flow-key was found by the prediction logic. In this case, the prediction cache did not find a match and the full lookup process must be invoked. The local prediction cache must be updated. The power required for searching and updating the prediction cache is wasted.

The prediction circuitry is effective because it exploits the temporal locality within the stream of network packets. All packets in a stream that are members of the same flow require the same forwarding treatment by an Ethernet switch. The observation that network communications exhibit strong locality and that this may be used to optimize resource utilization is not new [16]. There are differences of opinion as to whether the temporal locality of Internet traffic is sufficient to enable optimized forwarding using caching [17-19]. However, different parts of the network topology are exposed to a smaller number of flows and are expected to have a greater degree of locality than previously discovered by studies of core Internet traffic. Recent data center traffic, for example, has been observed to exhibit an ON-OFF pattern with strong temporal locality among the packet trains [20]. A brief study of the temporal locality in the network traces used in this work is described in Section V.

C. *Prediction Methods*

There are many ways to construct a packet signature, but since the prediction circuitry exists on each port of the switch, finding the balance between implementation cost and complexity is important. The basic approach is to compress significant (i.e. frequently changing) bits of the received packet into a signature used to search a prediction cache. The significant bits may come from predefined offsets in the packet or well known fields in the packet headers. Fig. 3 indicates which bit offsets in a flow-key vary the most between subsequent flow-keys in the Server Trace (see V.B.2). The other trace files obtained from different parts of the network topology have a similar frequency distribution. Combinations of these bits may be hashed to form portions of the signature or they may be directly mapped. This paper considers two different methods that tradeoff implementation complexity for accuracy.

1) *Direct Map*

The extremely simple Direct Map method extracts bits from predefined locations in the packet as it is arriving. The offset locations have been chosen to include bit fields that vary the most between subsequent flows as seen in Fig. 3. There is no logic that parses the packet and adjusts the offsets according to frame encapsulation or protocol. Bits are blindly extracted at pre-determined offsets. As a consequence, bit offsets that would normally align with the TCP port fields in an untagged Ethernet packet will be unaligned if the packet is VLAN tagged. Similarly these bit offsets may point to random payload data if the packet is an IP fragment (which contains no TCP header). For a practical implementation of the Direct Map method a different set of offsets should be considered based upon the port configuration.

As the bits arrive from fixed offsets, the circuitry builds a partial signature to present as a key to the fully associative prediction cache. Missing bits that have not yet arrived are marked as don't care conditions for the match. If no matching entries are found, there are clearly no previous elements from this flow in the cache and the packet must wait for the full flow lookup to complete. If there is precisely one entry found, then there is a chance that this entry is an exact match and the speculative forwarding of the packet may start immediately. This method forwards the packet as soon as possible, but can experience a higher misprediction rate than methods that perform more intelligent parsing.

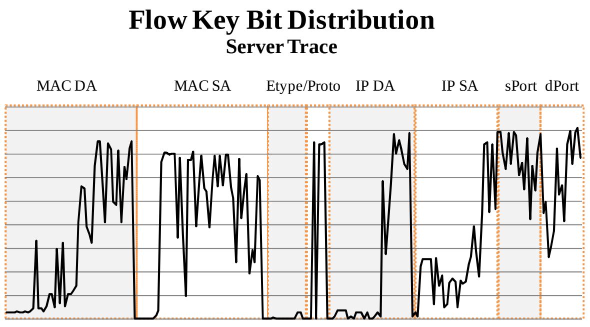

Fig 3. Frequency of bit variance in subsequent flow-keys showing which bits are most significant to incorporate into a packet signature

2) *Sub-Field Hash*

The Sub-Field Hash method intelligently parses incoming packets to extract precise sub-fields of packet headers and uses a simple hashing algorithm to construct segments of the packet signature. The goals of this method are to minimize the number of incorrect predictions, but also to support low latency by including a relatively aggressive eager approach to searching the prediction cache. The popular DJB hash function [21] is applied to sub-fields of the 29-byte flow-key as they arrive. The DJB hash function was chosen because of its simplicity, efficiency and distribution characteristics over small fields. The small hash results are combined to create partial signatures.

Similar to the Direct Map method, the prediction cache is searched as soon as a partial signature has been formed; where missing bits of the signature are marked as don't care conditions. The number of times the prediction cache is searched depends upon the length of the packet signature and the result of previous searches. A number of different packet signature sizes have been chosen to evaluate this sensitivity. The Sub-Field Hash method generates signatures of lengths 8, 16, 24 and 32 bits. For an 8-bit signature, the flow-key is divided into two parts; the MAC header is hashed to create a 4-bit partial signature and the IP and TCP headers are used to create another 4-bit quantity. These two quantities are combined to create an 8-bit signature with which the prediction cache is searched at most two times per packet. Longer signature types allow a greater number of partial signatures and thus may be more aggressive at speculating the forwarding of the packet. However, they will also search the cache more frequently, consuming more power. For example, in the implementation the 16-bit signature is made up of 5 hashes and in the worst case will search the cache 5 times. The 24-bit signature includes 9 hashes and the 32-bit signature includes 11 hashes.

While long signatures have the potential to invoke a search of the prediction cache a greater number of times and consume more power, they are generally more accurate at predicting the flow membership of a packet because there are a greater number of bits in the signature to distinguish one flow from another. It is important to minimize the number of incorrect predictions to avoid wasting power from incorrect speculative forwarding.

Other prediction methods are possible, but may require further trade-offs between complexity and cost at each port of the switch. For example, additional logic could enable application specific prediction algorithms or complex history traces. The Direct Map and Sub-Field Hash methods were chosen because they are stateless and can be implemented with simple combinational logic.

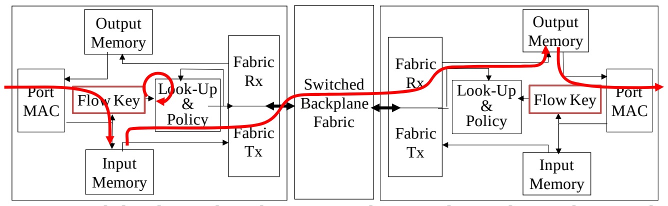

Fig. 4. High-level switch architecture without prediction logic, showing the four stages of the forwarding pipeline

III. Power Reduction from TCAM Bypass

The proposed enhancements reduce power by avoiding TCAM lookups when prediction and speculation are correct.

A. *Switch Power Model*

A power model is needed to evaluate the proposed enhancements. Various switch power models have been developed to support the analysis of power consumption at system-wide or network-wide levels [22-24]. Since the proposed enhancements are intended to be implemented in a switch ASIC, a power model that includes bit-level transfer energy is more appropriate [25]. In our model, we focus on the complete port-to-port switching path from the ASIC perspective. The forwarding process is divided into pipeline stages and each stage is analyzed individually assuming that packets are flowing at full line rate. We first consider a switch without the prediction logic as seen in Fig. 4. Later we replace the traditional lookup phase with a phase that includes the prediction logic and bypass circuitry.

The four stages of the forwarding pipeline are identified as follows:

1) Packet Receive (Rx): The Rx stage involves receiving the packet from the physical media, extracting important fields to build a flow-key and streaming the packet into the input memory system.
2) Lookup: The Lookup stage involves searching the TCAM for the received flow-key and retrieving the forwarding instructions.
3) Packet Transfer (Xfer): The Xfer stage involves reading the packet from the inbound memory, all of the logic required to initiate a transfer across the fabric, driving the fabric connections and crossbar, as well as writing the packet into the remote outbound memory.
4) Packet Transmit (Tx): The Tx stage involves reading the packet from the outbound memory and transmitting it on the physical media.

The energy required to progress a packet through the switch is simply the sum of these stages.

$$E_{pkt} = E_{rx} + E_{lookup} + E_{xfer} + E_{tx} \qquad (1)$$

Each of the stages can be looked at independently with its own set of individual components. The values used in the individual equations are summarized in Table I.

$$E_{rx} = pkt\_size * (E_{MAC} + E_{buf\_wr}) \qquad (2)$$

$$E_{lookup} = E_{TCAM\_search} + P_{in} * E_{data\_rd} + (1\text{-}P_{in}) * (E_{TCAM\_wr} + E_{data\_wr}) \qquad (3)$$

$$E_{xfer} = pkt\_size * (E_{buf\_rd} + E_{fab} + E_{buf\_wr}) \qquad (4)$$

$$E_{tx} = pkt\_size * (E_{MAC} + E_{buf\_rd}) \qquad (5)$$

The energy to receive (E_{rx}) and transmit (E_{tx}) an individual packet is dependent upon the packet size. In the Rx and Tx stages, the per-bit energy of the MAC and Serializer/Deserializer (SerDes) is given by E_{MAC}. The SerDes on an Ethernet link are continuously running and consuming power regardless of frame transmission. New efforts within the IEEE 802.3 working group are focused on reducing this consumption of power, however a typical 10GbE SerDes interface running at 3.125 Gb/s consumes about 150mW per channel [26]. Power is also consumed while streaming the received packet into and out of the buffer memory. The per-bit energy to read and write the buffer is given by E_{buf_rd} and E_{buf_wr}.

The energy consumed by the lookup stage (E_{lookup}) is of most interest since the prediction circuitry specifically aims to reduce this component. This stage is dependent upon whether a search of the TCAM is successful or not. If the flow-key is found in the TCAM, then the associated data memory is read to get the forwarding instructions. If the flow-key is not found, then both the TCAM and associated data memory must be updated. Let P_{in} be the probability that a flow-key is found in the TCAM given that it is not found in the prediction cache. The determination of this probability is actually quite complex and dependent upon many factors related to the characteristics of the traffic (e.g. temporal locality, duration of the flow, number of flows). While this probability is quite difficult to estimate, the significance of it is negligible when compared to the power consumed by the TCAM search itself. This search is always performed in a switch which does not have the prediction optimization. E_{TCAM_search} and E_{TCAM_wr} are the energy required to search and update the TCAM, and E_{data_rd} and E_{data_wr} are the energy required to read and write the associated TCAM flow memory and forwarding instructions, respectively.

The energy consumed by the Xfer stage is important to understand because it represents the cost penalty for incorrect speculation. This energy can be significant because it involves the use of several pairs of high speed SerDes and an external crossbar fabric chip (E_{fab}). The external fabric chip alone can consume between 15W and 20W [27].

B. *Switch Power Model with Prediction*

The packet prediction circuitry reduces latency through speculative switching and reduces power consumption by avoiding the lookup stage entirely. The power saved by avoiding searches of the TCAM may be offset by the power wasted from incorrect speculation. If correct prediction rates are high enough, the savings can be substantial. To determine the energy required to progress a packet through the switch architecture with prediction circuitry, the lookup stage is replaced with a new prediction stage. The per-packet energy is now given by:

$$E_{pkt} = E_{rx} + E_{predict} + E_{xfer} + E_{tx} \qquad (6)$$

The prediction phase energy is dependent upon the accuracy of prediction and caching schemes. Let P_{hit} be the probability of a prediction cache hit, $P_{incorrect}$ be the probability of an incorrect prediction cache hit and P_{miss} be the probability of a prediction cache miss. The energy consumed by the prediction phase ($E_{predict}$) is given by the following formula:

$$E_{predict} = P_{hit}{*}E_{cache\_hit}+P_{incorrect}{*}E_{cache\_incorrect}+P_{miss}{*}E_{cache\_miss} \quad (7)$$

E_{cache_hit}, $E_{cache\_incorrect}$ and E_{cache_miss} are the energy required to search and maintain the prediction cache when an entry is correctly found, incorrectly found or not found at all, respectively. They are calculated using the following equations:

$$E_{cache\_hit} = E(S) * E_{cache\_search} + E_{cache\_rd} \quad (8)$$
$$E_{cache\_incorrect} = E(S)* E_{cache\_search} + E_{cache\_rd} + E_{xfer\_min} + E_{tx\_min} + E_{lookup} + E_{cache\_wr} \quad (9)$$
$$E_{cache\_miss} = E(S)* E_{cache\_search} + E_{lookup} + E_{cache\_wr} \quad (10)$$

With eager prediction methods the prediction cache is searched with partial signatures multiple times until exactly one or zero entries are found. Let S be a random variable indicating the number of times the prediction cache is searched for a particular packet. Then E(S) is the expected number of searches in the prediction cache per packet. If an entry is found (E_{cache_hit}), then the associated flow-key RAM is read to retrieve the forwarding instructions. E_{cache_search} is the energy required to search the signature CAM and E_{cache_rd} is the energy required to read the associated flow-key and forwarding RAM.

In the unfortunate scenario where an incorrect prediction occurs ($E_{cache\_incorrect}$), speculative forwarding begins because a partial signature matches the prefix of a full signature in the prediction cache. In this case, the energy for the prediction and partial transfer is wasted. Let E_{xfer_min} be the energy needed to transfer the beginning of the packet across the fabric and into the output memory. This energy is the same as (4), but where the packet_size is set to the beginning portion of the packet up to and including the offset for the last field of the flow-key. In a similar fashion, let E_{tx_min} be the energy needed to transmit the partial fragment of the packet on the actual output link before being aborted. The energy of a full TCAM lookup from (3) is required in this scenario as well as the energy required to update the prediction cache denoted as E_{cache_wr}. If no packet signature is found (E_{cache_miss}) in the prediction cache, then a full TCAM lookup and cache update are required.

The parameters to the formulas (1) – (10) can be categorized into four main functional components: TCAMs, memory arrays, logic and I/O. The contribution to overall power consumption by these components in a particular chip depends upon the target configuration. For example, low-end systems implemented with a single chip may have more power consumption due to logic than I/O or a TCAM because of the lack of a backplane and limited functionality. Higher-end systems with high-performance backplanes, high-speed links and large TCAMs may see a greater amount of energy dissipated by the SerDes and the TCAM. Fig. 5 shows the measured distribution of power for three different switching ASICs in the HP ProCurve family of Ethernet switches. The amount of power consumed by the TCAM becomes a significant portion of the switch chip as the performance and feature set address the high-end of the market.

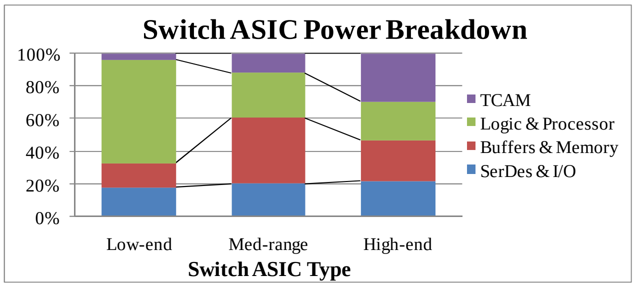

Fig. 5. Functional component percentage breakdown of total ASIC power measured in three different HP ProCurve switching ASICs ranging from low-end, single chip switches to high-end multi-chip chassis designs

C. *Source of Power Modeling and Estimation*

Evaluation parameters for the power model come from a variety of sources. The energy estimates for the memory structures, CAMs and TCAMs are performed using existing power modeling tools. The energy for the SerDes, fabrics and Ethernet MAC logic are taken from existing switch chip implementations. Table I shows representative values for the power model parameters.

The SerDes used for the MAC and backplane are the biggest contributors to overall power consumption and cannot be reduced by the proposed enhancements. Future switch designs incorporating optical interconnects have the potential to lessen this part of the power equation. Searching the TCAM is the next significant power consumer and reducing this contribution, along with overall latency, is the focus of this work.

1) *CAMs and TCAMs*

The model for power consumption of the array structures and CAMs is based on the well known cache and memory

TABLE I
POWER MODEL PARAMETERS

Parameter	Description	Model Source	Representative Values (nJ)
E_{cache_search}	Searching prediction CAM	CACTI	0.003759 nJ (32 rows x 24 bits)
E_{cache_wr}	Updating flow key, forward RAMs and cache signature	CACTI	0.062691 nJ (32 rows x 24 bits, 87 bytes)
E_{cache_rd}	Reading the flow key and forwarding RAMs	CACTI	0.062217 nJ (32 rows x 240 bits)
E_{TCAM_search}	Searching the TCAM	Sherwood	7.141493 nJ (8K rows x 240 bits)
E_{TCAM_wr}	Updating the TCAM on miss	Sherwood	0.931197 nJ (8K rows x 240 bits)
E_{data_wr}	Updating the forwarding instruction RAMs	CACTI	0.365080 nJ (8K rows x 58 bytes)
E_{data_rd}	Reading the forwarding instruction RAMs	CACTI	0.387645 nJ (8K rows x 58 bytes)
E_{MAC}	1GbE MAC and standard SERDES	Reference Implementation	97 nJ per 64 byte packet
E_{buf_wr}	Writing to the packet buffer memory	CACTI	4.4993 nJ for 64 byte packet
E_{buf_rd}	Reading the packet buffer memory	CACTI	4.5048 nJ for 64 byte packet
E_{fab}	Driving two ports of the crossbar fabric and SERDES	Reference Implementation	96 nJ per 64 byte packet

systems modeling tool CACTI [28]. CACTI is an integrated cache and memory access time, cycle time, area, leakage, and dynamic power model. A TCAM power modeling tool developed by Sherwood and Agarwal [29] augments CACTI and is used to calculate the power for TCAM searches, reads and writes.

In the switch architecture there are two different fully associative content-addressable memories that need to be modeled; the signature CAM and the master TCAM. The signature CAM is a very small fully associative memory with 2, 4, 8, 16 or 32 rows and a tag width of 8, 16, 24 or 32 bits. The small CAM depths were chosen to keep the per-port cost of the implementation low while exploring the sensitivity of flow caching. The CAM width is dependent upon the signature size, which was also chosen to be small in order to limit costs and explore the sensitivity of representing a flow key in as few bits as possible.

The master TCAM is a large structure that uses a 29 byte flow-key as the search key. The key size is fixed by the concatenation of standard fields from the packet header. The TCAM can be segmented to reduce power consumption during searches as described in various optimization schemes [30-32]. The size of the TCAM can vary based upon the number of simultaneous flows that the switch is attempting to support. Depths of 8K, 16K, 32K, 64K and 128K were chosen to cover a span of common implementations found in practice today. A power consumption model for E_{TCAM_search} is calculated using the Sherwood TCAM tool. The results from the Sherwood TCAM tool are validated against implementations from NetLogics Inc. and the HP ProCurve 5400 family ASIC and were found to be with 15% of nominal values.

2) *Memory Arrays*

The flow-key RAM associated with the prediction cache is 29 bytes wide in order to support the standard concatenation of packet header fields that make up a flow-key. The width of the forwarding instruction RAMs are implementation specific and a width of 54 bytes was chosen to match an existing known implementation. There is a shallow forwarding instruction RAM associated with the prediction cache and a deep RAM associated with the master TCAM. In all cases these memories are modeled as having a single bank with a single read/write port.

3) *Fabrics and MACs*

The calculation of E_{fab} includes the power consumed by an external fabric chip and the SerDes interfaces on the network chip required to interface the line card to the backplane. There are many different switch fabric architectures and a detailed analysis of the power consumption of these architectures is provided by Ye, Benini and Micheli [25]. This analysis focuses mainly on the internal architecture of the fabric and includes the power consumed by node switches, internal buffers and interconnects wires. Reference implementations [27], however, indicate that over 50% of the power for an external chip is associated with the high-speed SerDes. For example, 7.7W of the 15W total described in [25] are due to the sixty four 3.25 Gbps SerDes. Using this reference as an estimate and including the same SerDes estimate on the line card, a measure for E_{fab} is found to require approximately 96 nJ per minimum sized packet. Similar numbers are calculated from information available on fabric chips from Dune Networks [27].

The calculation of E_{MAC} includes the power consumed by the combinational logic needed to implement the Ethernet MAC and the power needed by the SerDes. An Ethernet MAC implementation will have variations across chips, but in general consumes a small amount of energy in comparison to the SerDes. A typical 1GbE SerDes implementation will consume about 125 mW, or approximately 84 nJ per minimum sized packet. An example Ethernet MAC in the HP ProCurve mid-range ASIC consumes about 19 mW or approximately 13 nJ per minimum sized packet. Together these provide an estimate for E_{MAC} of 97 nJ per minimum sized packet.

IV. Latency Reduction from Speculative Cut-Through Switching

The prediction enhancement speculates on the flow membership of the next received packet allowing the forwarding engine to apply a rich set of forwarding policies to the packet while it is still being received. This allows switch forwarding to begin with the lowest possible latency

A. Switch Latency Model

To understand how the prediction enhancement improves latency, it is first necessary to develop a model for switch latency. The switch architecture defined in Section II is applicable to either a store-and-forward switch or a cut-through switch. In a store-and-forward switch, the entire packet is completely received on the ingress port before lookup operations begin. In a cut-through switch, the lookup begins as soon as enough of the packet has been received to assemble a flow-key.

To increase throughput, the process of switching a packet can be pipelined. While the lookup process is working on a packet, the next packet can be copied from the ingress port to the input memory, and the previous packet can be modified and transferred to the output memory of the egress port. Fig. 6 shows the pipeline diagram for the store-and-forward switch.

In order for the switch to maintain line rate forwarding, no stage of the pipeline can exceed the time it takes to receive a packet from the wire. On a 10 Gbps Ethernet port there are potentially 14.88 million minimum size packets arriving per second; therefore, no stage can exceed 67.2 ns. A generalized way to look at the minimum required pipeline stage length is to normalize the stage to received bit times. A minimum size Ethernet packet is 64 bytes, and is therefore received in 512 bit times as determined by the speed of the ingress link.

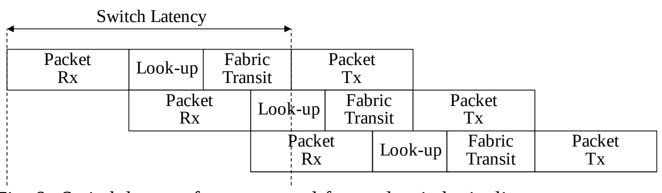

Fig. 6. Switch latency for a store-and-forward switch pipeline

The duration of the packet Rx and packet Tx stages of the pipeline are directly tied to the physical media line rate. The fabric transit and packet modification stage is faster than the physical media line rate. Therefore, to forward at line rate, the lookup stage and the fabric transit stage must be no longer than the time it takes to receive a minimum sized packet. To simplify the calculation of switch latency we assume the lookup stage time will be a constant and equal to the amount of time it takes to receive a minimum size packet.

Let K_{sf} be the number of bits in a minimum size Ethernet packet (which is the constant 512). Let R_p be the received port line rate in bps and let L be the length of the packet in bits. Let R_f be the fabric interface transfer rate in bps and assume that $R_f > R_p$. Switch latency is the amount of delay a packet experiences inside the switch, and will be measured as the amount of time between when the first bit of a packet is received on the ingress port and the time the first bit is transmitted on the egress port. The formula for store-and-forward switch latency is then:

$$\text{Store and Forward Latency} = (L / R_p)+(K_{sf} / R_p)+(L / R_f) \quad (11)$$

This formula represents the time taken to receive the packet plus the time to perform the lookup stage plus the time to make any modifications and transfer the packet across the fabric. (Packet transmission on the egress port is assumed to start immediately once the packet is in the output memory.)

To improve store-and-forward switch latency, two things must change. First, the lookup process and packet modification with transfer across the backplane must begin before the current packet has been completely received. Second, the transmission of the packet on the egress port must also be allowed to begin before the current packet has been completely received.

In the cut-through model of a switch without any prediction the lookup process can begin no sooner than after the last bit of the packet needed to construct a flow-key has been received. Let $D = \{D_1(H_1), D_2(H_2), \ldots, D_n(H_n)\}$ be the set of functions in the classification process that return the starting bit displacement for the flow-key fields in H. Then $D_n(H_n)$ is the starting bit offset for the last field necessary to create the n-tuple needed for the lookup. If K_{ct} is defined as the number of bits that must be received to construct the flow-key for the lookup stage to begin then K_{ct} is determined as:

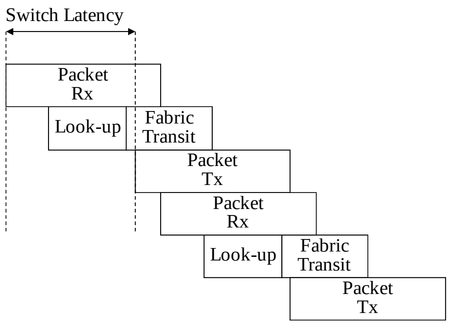

Fig. 7. Switch latency for a cut-through switch pipeline

$$K_{ct} = D_n(H_n) + |H_n| \quad (12)$$

Assuming that packet modification is part of the fabric transfer stage and the transmission of the received cut-through packet may begin as soon as the first bit has arrived in the output memory, then the following formula for cut-through switch latency is:

$$\text{Cut-Through Latency} = (K_{ct} / R_p) + (K_{sf} / R_p) + 1/R_f \quad (13)$$

This formula represents the time taken to receive enough of the packet to construct the flow-key plus the time to perform the lookup stage in order to maintain line rate plus the time to optionally modify the packet and transfer the first bit of the packet across the fabric. Packet transmission on the egress port is assumed to start immediately once the first bit of the packet is in the output memory. Fig. 7 shows the pipeline diagram for a cut-through switch.

1) Further reductions to Switch Latency

The switch latency of both store-and-forward and cut-through switches can be reduced by exploiting two key concepts from computer architecture; value prediction and speculative execution [33]. Value prediction attempts to remove the limits on parallelism imposed by true data dependencies, while speculation seeks to reduce the latency of obtaining computed results. In an Ethernet switch, the lookup stage is a data dependent operation that requires the reception of enough packet data to construct a flow-key, and the operations applied to the packet once the lookup completes must endure the lookup latency before forwarding can begin. Packet prediction and speculative switching remove this barrier by exploiting the temporal locality of network traffic to predict the data being received, allowing speculative packet operations for the flow to begin before the lookup has completed.

Switch latency for a packet predicting speculative switch is limited by the time it takes to generate enough of the packet signature to confirm a match in the prediction cache. Let $S = \{S_1, S_2, \ldots S_m\}$ be a packet signature of length m that consists of a set of bits that have been derived from the fields in H. There are a set of functions F that derive bits S_i through S_j of S from the fields in H as they arrive from the link. Let K_p be the number of bits received to form enough of S to find a match in

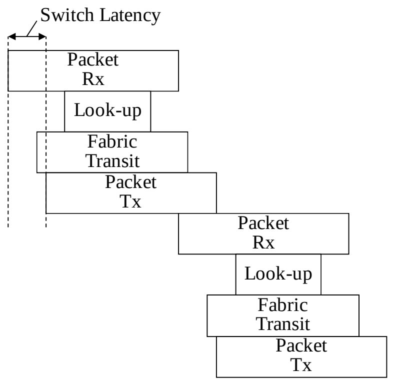

Fig. 8. Switch latency for a cut-through switch with packet prediction in the forwarding pipeline

the prediction cache. Then the latency for a packet predicting speculative switch is as follows:

$$\text{Prediction with Speculation Latency} = (K_p/R_p)+1/R_f \qquad (14)$$

This formula represents the time taken to receive enough of the packet to construct enough of the packet signature to find a match in the prediction cache, plus the time to transfer the first bit of the packet across the fabric. This results in the pipeline diagram shown in Fig. 8.

B. *Implementation Considerations*

As with any pipeline employing speculation, there are several complications and clean-up steps that may be required. Once the entire flow-key has been received it is possible to confirm the correctness of the speculation. Since it is possible for the first bit of the packet to be transmitted on the egress port before the results of the flow-key match are complete, the current egress transmission may be incorrect. Such an incorrect speculation requires that the packet transmission be aborted in some cases. To accomplish this, the packet must be corrupted by the transmitter before the last bit is sent. (However, this may not always be necessary - in the case of a layer 2 bridged network, a packet that is forwarded through the wrong port would simply appear as an extraneous flood and typically not cause an error.)

The reduction in latency for a given configuration of cache size and signature size is directly related to the number of concurrent flows and amount of temporal locality in those flows. To maintain effectiveness, the scheme will require more resources if applied to aggregated links in the core of the network because there will likely be a greater number of concurrent flows on those links. Similarly, the general lookup process for switches attached to aggregated links will require larger table sizes and larger TCAMs to support increased concurrent flows, so the prediction scheme has similar scaling properties.

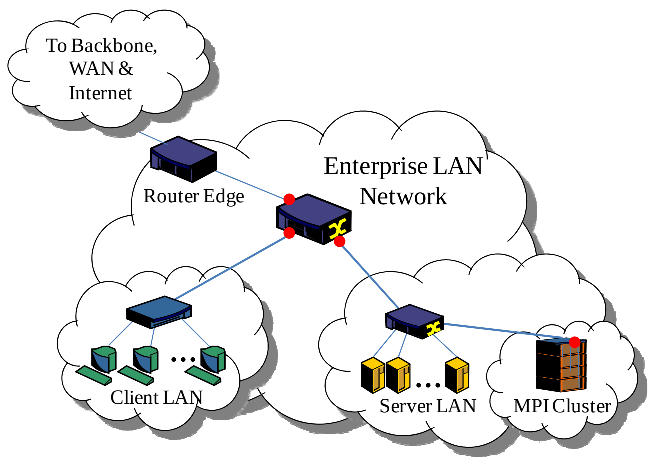

Fig. 9. Network topology containing representative trace capture points

TABLE II
TRACE DATA SET ANALYSIS

Trace File	Packet Count	Average Packet Size	TCP	UDP	other	P_{in} (24x32)	E(S) (24x32)
MPI-BT	237K	1161	100%	0%	0%	0.98	6.51
MPI-CG	237K	1243	100%	0%	0%	0.00	2.67
MPI-EP	1141	150	83%	15%	2%	0.43	4.25
MPI-FT	237K	1160	100%	0%	0%	0.00	4.52
MPI-IS	236K	1194	100%	0%	0%	0.97	6.42
MPI-LU	239K	899	100%	0%	0%	0.00	4.79
MPI-MG	237K	1169	100%	0%	0%	0.00	4.49
MPI-SP	236K	1238	100%	0%	0%	0.00	4.98
Router	2.2M	344	96%	2%	2%	0.80	5.93
Server	490K	198	55%	44%	1%	0.53	6.24
Client	250K	151	38%	40%	22%	0.83	5.89

V. EVALUATION

A. *Experimental Set-up*

In order to evaluate the effectiveness of the prediction approach, a program was written that consumes actual traces of network traffic and simulates the behavior of the proposed OpenFlow switch architecture. Each packet received is assumed to be exposed to the full flow-matching logic of the OpenFlow switch.

B. *Traffic Analysis on Representative Traces*

The traces contain packet data from different network environments and different parts of the network topology as seen in Fig. 9. The highlighted ports in the figure show the representative locations where trace files were captured. Network ports that are closer to individual stations have fewer multiplexed flows, and network ports that are in the core of the network or at the Internet edge are likely to have a greater number of multiplexed flows. Prediction methods are expected to be most effective in the data center, near clusters of message passing servers, where the total number of flows is expected to be relatively small and low latency cut-through switching will be most beneficial.

Four different trace datasets, described below, were used in the simulations and a summary analysis of the trace data is shown in Table II.

1) *Router Traces from LBNL*

Lawrence Berkeley National Laboratory (LBNL) maintains 11 GB of anonymized packet header traces from October 2004 through January 2005, which are available for download from http://www.icir.org/enterprise-tracing/download.html. These traces include enterprise campus LAN traffic from subnet links connected directly to the site router. A thorough analysis of these traces is available in [34].

2) *Server Traces*

The Server trace files were captured from the LAN backbone of a network-engineering department at the Hewlett-Packard Company in May 2008. The trace selected contains only inbound traffic to a core switch with a backbone 10GbE port connecting the engineering development servers. The outbound traffic is not included in the trace, which more accurately represents the type of traffic the prediction logic would be exposed to in an implementation of the architecture

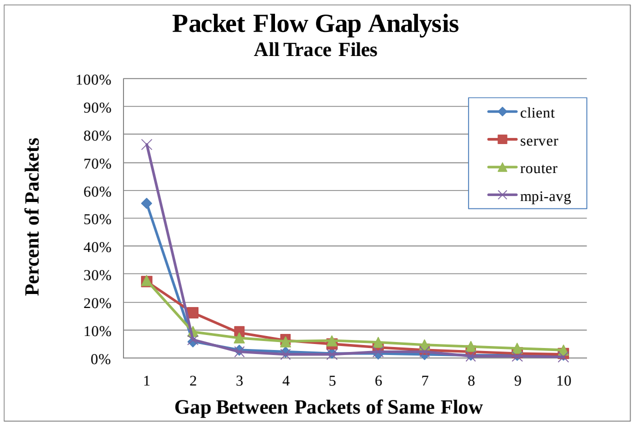

Fig. 10. The packet gap analysis shows the distribution of spacing between consecutive packets of the same flow up to a spacing of 10 packets.

in Section II.

3) Client Traces

The Client trace files were captured from a link to a workgroup switch in the same engineering department. The trace selected for analysis captures only the inbound activity of a small number of engineering users, and therefore the source addresses of the packets are predominantly client stations. This trace has the highest percentage of traffic that is neither TCP nor UDP as seen in Table II.

4) MPI Traces

The National Air and Space Administration (NASA) maintain a set of benchmarks developed by the Numerical Aerodynamic Simulation (NAS) organization in order to analyze the performance of parallel computer systems. These tools are called the NAS Parallel Benchmarks (NPB). The benchmarks are recognized in the industry as a representative suite of parallel applications. The traces collected for this study are the ingress capture of an individual 1 GbE port connected to one of the 16 compute nodes in a Linux Rocks cluster. Individual trace files were captured for each benchmark in the suite, and the inter-cluster communication used was MPI over Ethernet. Complete details of the NPB suite may be found at [35].

C. Temporal Locality of Network Traces

Fig. 10 illustrates the temporal locality of the trace data sets by comparing the gap between consecutive packets of the same flow. The figure shows the percentage of packets that have a particular spacing between a previous packet of the same flow. The figure only shows the distribution of packet spacing up to a gap of 10 packets, which covers approximately 75% of all packets in the traces. The remaining ~25% of the packets lie in the long tail of the distribution. The measured distribution of the packet flow gap in the trace datasets closely matches the results observed in [16].

Traces that have been acquired from links that aggregate fewer flows and are physically closer to end-stations have the highest temporal locality (MPI and Client). The ability to predict flow membership with a small per-port cache is expected to be most effective on these traces.

VI. Results

A complete set of simulation results for both power and latency reductions were obtained for each combination of cache size, signature size and trace file. In the following figures a signature size of 32 bits is commonly used for consistency and because it highlights notable aspects of the proposed enhancements.

The ability to reduce latency and power is strongly dependent upon the rate of correct predictions generated by a prediction method. Fig. 11 shows P_{hit}, $P_{incorrect}$ and P_{miss} for the Sub-Field Hash method when run with all trace files. The figure confirms that the prediction circuitry is more effective when placed closer to servers in the data center. The MPI traces have very high temporal locality, resulting in P_{hit} rates nearing 99%.

Fig. 12 compares the different prediction methods with the most diverse Router trace using 32-bit signatures. The figure

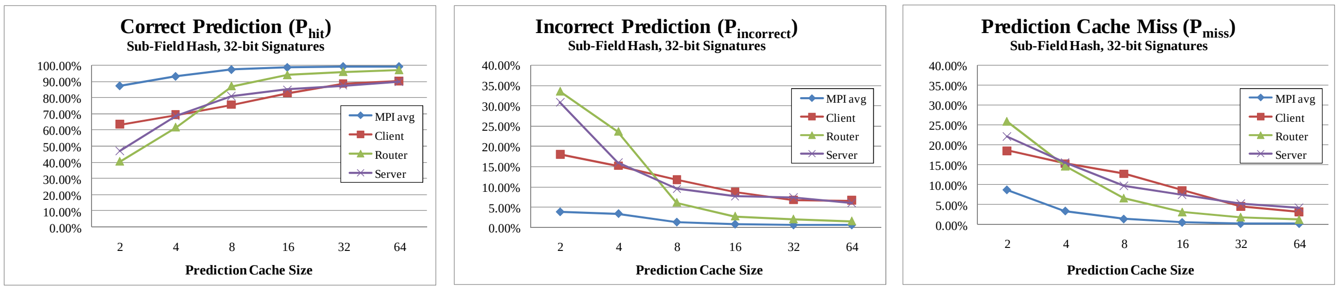

Fig. 11. Prediction rates for each trace data set using the Sub-Field Hash method with 32-bit signatures and varying prediction cache size

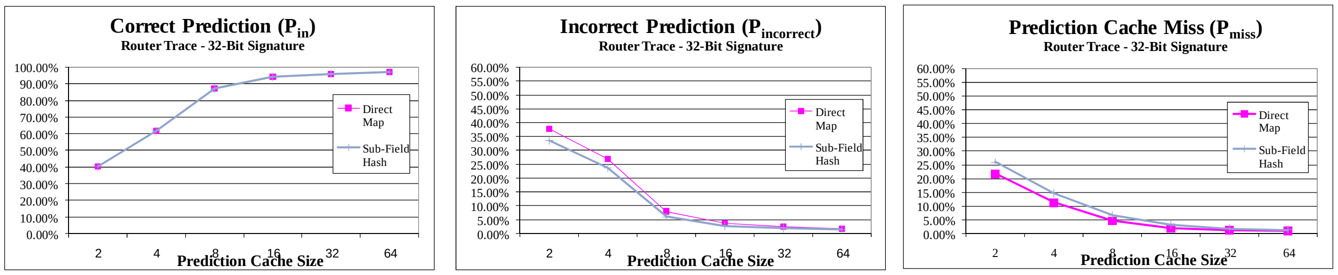

Fig. 12. Prediction rates for each prediction method run against the Router trace with 32-bit signatures and varying prediction cache size

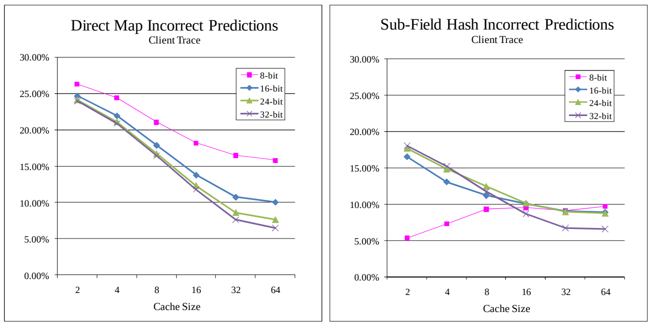

Fig. 13. Difference between Direct Map and Sub-Field Hash methods when making incorrect predictions on the Client trace.

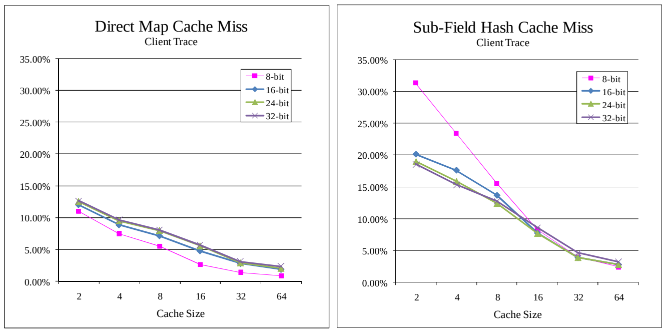

Fig. 14. A comparison of prediction cache misses on the Client trace. Cache misses are preferred to incorrect predictions which need speculation clean-up.

clearly shows that correct prediction rates nearing 97% are possible even as the technique is placed deeper in the network topology. It also shows that the methods have a high incorrect prediction rate when the cache size is small. This is understandable since both methods stop searching the prediction cache under two conditions - when there is exactly 1 entry that matches the partial signature, or when there are no entries that match. When the cache size is small it is more likely that a small partial signature will match exactly 1 entry because there is little diversity in the cache. Larger caches support more diversity, reducing the chance of a false positive match and thus the number of incorrect predictions.

A prediction cache miss occurs when a new flow is established or the signature for an existing flow has been removed from the cache. Fig. 12 shows that the Direct Map method has a slightly lower prediction cache miss rate with small caches (~20%) than the Sub-Field Hash method. This is because the Direct Map method has a higher incorrect prediction rate with lower cache sizes. Incorrect predictions are not counted as cache misses - whether there is an incorrect prediction or a cache miss, the same switch latency penalty is paid, so the more speculative approach tends to benefit in the overall latency calculations. The downside to the more speculative Direct Map approach is that it potentially wastes backplane resources and power, which in practice is not free.

The Sub-Field Hash method has lower incorrect predictions, since it takes into account a greater number of bits when creating a signature. This is particularly relevant when the signature size is small and the cache is large as seen in Fig. 13. As the number of bits used to represent a signature grows the two methods perform similar.

Fig. 14 shows that the Direct Map method has lower prediction cache misses on the Client trace than the Sub-Field Hash method, but at the expense of greater incorrect predictions. Recall that the Client trace has the highest mix of non TCP/UDP traffic – the Client trace dataset has 18% ARP packets and 2% other layer 2 frames, while the Server and Router trace files have 99% and 98% IP traffic, respectively. Since the Direct Map method simply extracts bits from predetermined offsets, and those offsets are optimized for TCP/UDP traffic, it is no surprise that the Direct Map method has the higher number of false positive matches between the two. The two figures show the effectiveness of hashing over selecting predefined bits for all signature and cache sizes used with the Client trace.

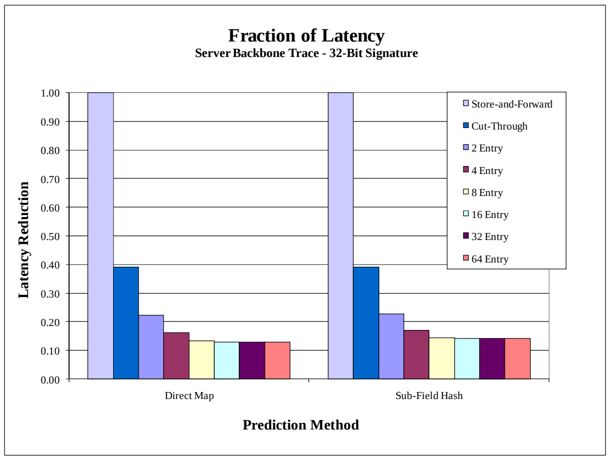

Fig. 15. A comparison of the latency reduction achieved by each prediction method on the Server traces using 32-bit signatures

When considering how latency can be reduced, one would expect that improved accuracy from the largest signature size and cache size would be the most effective. However, for latency reduction, the objective of packet prediction is to begin forwarding the packet as soon as possible with the highest probability that the speculation is correct. Fig. 15 shows the reduction in switch latency on the Server trace for different packet prediction schemes as compared to a conventional store-and-forward and cut-through switch.

The figure shows there is very little difference between methods when it comes to latency reduction. This is because

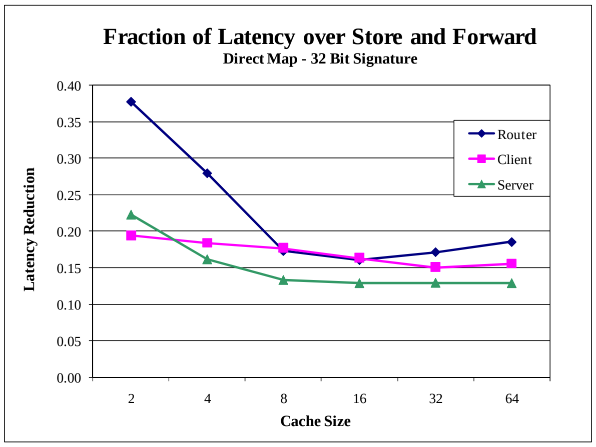

Fig. 16. Direct Map method latency reduction for the Client, Server and Router trace datasets showing increasing delays at larger cache sizes

both methods have similar correct prediction rates. Where the methods differ is in the mix of incorrect predictions and cache misses as seen in Fig. 13 and Fig. 14. These differences will have a bigger impact on power reduction since incorrect predictions require aborting backplane transfers which are strictly a waste of power. The Direct Map method has the lowest latency with a 64-entry cache - the switch latency for this configuration is 0.13 times the latency of a store-and-forward switch and 0.33 times the latency of a cut-through switch. This corresponds to nearly a factor of 8 and a factor of 3 reduction in latency, respectively.

Fig. 16 shows the Direct Map method latency reduction for the Client, Server and Router trace datasets. It is interesting to note that as the cache size increases the performance of the Direct Map method degrades for the Router trace. This phenomenon occurs because a larger number of stale entries exist in the prediction cache, which only serve to further delay the exact match of partial signatures.

The baseline power for an OpenFlow switch that uses a TCAM for flow matching is highly dependent upon the size of the TCAM. An OpenFlow switch can be optimized for different locations in the network topology by selecting different TCAM sizes. Locations where a large number of flows are aggregated are better served by a large TCAM, and locations in the topology that are closer to individual stations, such as the MPI cluster or client workgroup, can get by with a smaller TCAM.

As the size of the TCAM grows, the average energy consumed by a flow match increases. Packet prediction reduces this energy by avoiding the lookup process. Fig. 17 shows the per-packet energy savings over a switch without prediction. Even with the smallest TCAM size evaluated, the average energy consumed by the lookup phase for MPI traffic is only 7% of the power consumed by a switch without prediction. At the largest TCAM size considered, the average energy for MPI traffic is only 1%.

The Client trace has the highest energy consumption of all the sample datasets. This is primarily because it has the highest incorrect prediction rate and incurs the most cost for incorrect speculation. The lookup phase for the Client trace consumes 68% of the power required for a switch without prediction at the smaller TCAM size and 16% of the power

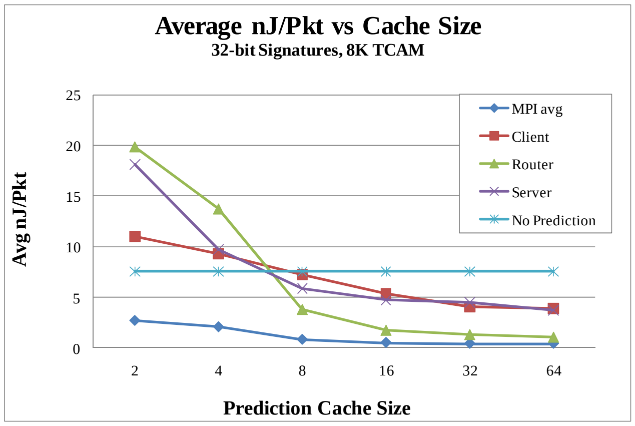

Fig. 18. Power consumption of Sub-Field Hash method on all trace files showing the breakeven point against a switch with no prediction

using the larger TCAM. Increasing the prediction cache depth tends to increase the prediction accuracy, so it should be an effective approach for reducing the load on the TCAM. The larger prediction cache will consume more power, but its increased effectiveness will offset the power required to deploy a larger TCAM.

There are cases where the prediction circuitry results in a switch that consumes more power than a switch without the circuitry. Fig. 18 shows the smallest TCAM size evaluated (8K) with the largest signature size (32-bits). Anytime the prediction cache is less than 8 entries deep, the system suffers too many cache misses and incorrect predictions to keep the cost of speculation below that of a system without prediction. This is true for all traces except the MPI traces which were collected from a relatively small cluster and exhibit extreme temporal locality. Clearly ASIC designers should not consider cache sizes less than 8 entries and larger depths will better support a greater number of simultaneous flows as seen in the Server and Router traces.

Since the prediction enhancements ultimately aim to simultaneously reduce both power and latency, the goal is to find a configuration that optimizes both. Fig. 19 shows the power reduction verses latency reduction for the Router trace using the Sub-Field Hash method. Values located in the lower right hand portion of the figure are best. Clearly 8-bit signatures are not able to predict flows well enough to be useful. As previously shown, 16 entry caches are a turning

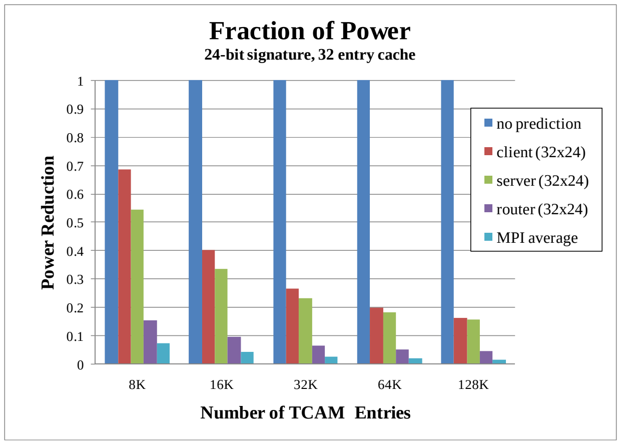

Fig. 17. A comparison of the power reduction achieved by the Sub-Field Hash method across all trace data sets

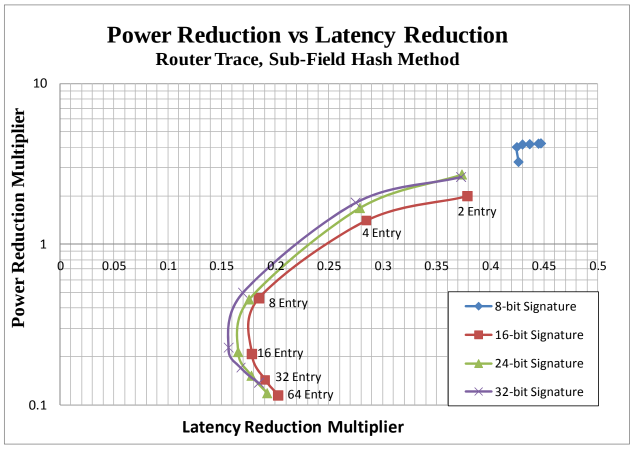

Fig. 19. The packet gap analysis shows the distribution of spacing between consecutive packets of the same flow up to a spacing of 10 packets.

point for the methods when reducing latency. Larger signatures and larger cache sizes appear to be effective at reducing power with only slight increases in latency.

In summary, the results show that even for the most diverse Router trace, correct flow prediction rates approaching 97% with a simultaneous reduction in latency by a factor of nearly 5 and a reduction in power of the lookup phase by a factor of 7 are achievable. While all parts of the network topology can benefit from a switch with prediction, the high performance computing cluster gains the most. The two methods perform similarly for latency reduction, but the Sub-Field Hash method has a slight performance advantage when also considering power reduction because of its lower incorrect prediction rate. Both methods are most effective with cache sizes around 16 entries. Larger signatures are more effective at reducing power and signatures less that 16 bits are not worth considering.

VII. Related Work

This work is an extension of a previous contribution that focused exclusively on latency reduction [36]. This work includes additional trace data sets for MPI communication, the Sub-Field Hash prediction method, a power model and the ability to also reduce the power of the lookup process.

A. *Reducing TCAM Power*

Reducing the power consumption of network devices that use TCAMs has been the focus of a number of different studies [31, 32, 37]. A common technique in most previous approaches is to segment the TCAM into blocks and only search individual blocks as needed. Additional front end logic is provided to locate the individual block that likely contains the search entry. Power is saved by only driving current through the lines of that individual block. The use of TCAM segmentation and intelligent organization to reduce power consumption during search is orthogonal to the prediction enhancement and in fact is compatible with the approach that further aims to reduce power by simply bypassing these operations.

Mogul et al reduce TCAM lookups in an OpenFlow switch by first searching for fully qualified flow-keys in a large hash table [38]. This multi-layer scheme saves power and also reduces pressure on the TCAM size. Kasnavi uses a multizone pipeline cache to reduce power and exploit the temporal locality of IP routed traffic [39]. These schemes are similar to the prediction enhancements because they use caching to reduce power, but neither attempt to simultaneously reduce latency.

B. *Latency Reduction using Prediction*

Using prediction techniques to reduce latency and improve communication performance has not been extensively studied. There is no known prior work that has used compressed packet content to facilitate prediction in parallel with the flow classification process on a statistically multiplexed packet switch.

The goal of reducing the number of pipeline stages in a routing switch by using predictive switching was proposed in [40]. The proposed technique looks at the forwarding history of an ingress port to predict the egress port, irrespective of the contents of the packet. The 2-D torus network for which this prediction scheme was developed is a connection oriented switch for specialized high performance computing applications. The authors achieved 77% prediction accuracy using the NAS parallel benchmarks.

Speculative techniques are proposed to reduce the latency of setting up paths across a connection oriented crossbar fabric in [41, 42]. In these works, fabric bandwidth is arbitrated under the speculation that fabric virtual channel allocation will typically succeed. All of this occurs after the decode and routing stages (i.e. packet Rx and lookup stages). However, as the richness of the forwarding policy increases, the complexity and latency of decode and routing stages will begin to dominate.

The comparison of predictive switching ideas with concepts from advanced computer architecture is best described in [43]. Surendra et al show how the temporal locality of network traffic can be exploited to improve instruction reuse and reduce latency in network processors. Their proposal is to have a separate instruction reuse buffer for each active flow and to swap the processor context when a packet is received from a different flow. The approach is conceptually similar to our prediction enhancement in that the prediction cache holds a set of forwarding instructions for packets from a particular flow. The same temporal locality is used to select a context and speculatively operate on a packet. However, they assume the packet classification completes with 100% accuracy and there is no speculation. They are focused on speeding up the instructions that operate on the packet after the classification is done, where our approach uses a cache in parallel to speculate the results of the classification.

VIII. Conclusion

Enhancing an OpenFlow switch with per-port packet prediction circuitry is an effective means for simultaneously reducing power and switch latency without sacrificing flexibility and rich packet processing. Correct prediction rates approaching 97% are achievable with a moderate amount of per-port circuitry. Two different prediction methods that trade-off per-port complexity for accuracy where shown to be effective. The more accurate Sub-Field Hash method is more effective at reducing power consumption because of a lower incorrect prediction rate while equivalent latency reduction can be achieved even with the simplistic Direct Map method. The results of simulations using real network data have shown that packet prediction can reduce the latency of a traditional store-and-forward switch by nearly a factor of 8 and reduce the already low latency of a cut-through switch by a factor of 3. Depending upon the locality of the network traces, the average energy required in the lookup phase of an OpenFlow-based Ethernet switch can simultaneously be reduced as well. While all parts of the network topology can benefit from a switch with the proposed circuitry, the high performance computing cluster gains the most.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69-74, 2008.

[2] B. Salisbury, "TCAMs and OpenFlow - What Every SDN Practitioner Must Know," SDN Central, 2012, http://www.sdncentral.com/products-technologies/sdn-openflow-tcam-need-to-know/2012/07/

[3] B. Heller, "OpenFlow Switch Specification," OpenFlow Consortium, 2008, http://www.openflowswitch.org/documents/openflow-spec-v0.8.9.pdf

[4] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1998, pp. 1240-1247 vol.3.

[5] H. H. Y. Tzeng and T. Przygienda, "On fast address-lookup algorithms," *Selected Areas in Communications, IEEE Journal on*, vol. 17, pp. 1067-1082, 1999.

[6] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," *Selected Areas in Communications, IEEE Journal on*, vol. 21, pp. 560-571, 2003.

[7] J. Liao, "SDN System Performance," 2012, http://pica8.org/blogs/?p=201

[8] R. Ozdag, "Intel Ethernet Switch FM6000 Series - Software Defined Neworking," Intel Corporation, 2012, p. 8.

[9] Arista, "7150 Series 1/10 GbE SFP Ultra Low Latency Switch," Arista Networks, Inc., 2012.

[10] Cisco, "Cisco Nexus 3548 Switch Architecture," San Jose, CA: Cisco Systems, Inc., 2012, p. 12.

[11] D. Serpanos and T. Wolf, *Architecture of Network Systems*. Boston: Morgan Kaufmann.

[12] P. Gupta and N. McKeown, "Algorithms for packet classification," *Network, IEEE*, vol. 15, pp. 24-32, March 2001 2001.

[13] D. Taylor, E., "Survey and taxonomy of packet classification techniques," *ACM Comput. Surv.*, vol. 37, pp. 238-275, 2005.

[14] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, "Next generation on-chip networks: what kind of congestion control do we need?," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* Monterey, California: ACM.

[15] C. Minkenberg, M. Gusat, and G. Rodriguez, "Adaptive Routing in Data Center Bridges," in *High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on*, 2009, pp. 33-41.

[16] R. Jain and S. Routhier, "Packet Trains--Measurements and a New Model for Computer Network Traffic," *Selected Areas in Communications, IEEE Journal on*, vol. 4, pp. 986-995, 1986.

[17] C. Partridge, "Locality and route caches," 1996, http://www.caida.org/outreach/isma/9602/positions/partridge.html

[18] D. C. Feldmeier, "Improving gateway performance with a routing-table cache," in *INFOCOM '88. Networks: Evolution or Revolution, Proceedings. Seventh Annual Joint Conference of the IEEE Computer and Communcations Societies, IEEE*, 1988, pp. 298-307.

[19] P. Newman, G. Minshall, T. Lyon, and L. Huston, "IP switching and gigabit routers," *Communications Magazine, IEEE*, vol. 35, pp. 64-69, 1997.

[20] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding Data Center Traffic Characteristics," in *SIGCOMM 2009 Workshop on Enterprise Networking* Barcelona, Spain: Association for Computing Machinery, 2009.

[21] A. Partow, "General Purpose Hash Function Algorithms," 2013, http://www.partow.net/programming/hashfunctions/index.html

[22] G. Ananthanarayanan and R. H. Katz, "Greening the switch," in *Proceedings of the 2008 conference on Power aware computing and systems* San Diego, California: USENIX Association, 2008.

[23] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," *NETWORKING 2009*, pp. 795-808, 2009.

[24] H.-S. Wang, L.-S. Peh, and S. Malik, "A power model for routers: Modeling Alpha 21364 and InfiniBand routers," *Micro, IEEE*, vol. 23, pp. 26-35, 2003.

[25] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," in *Design Automation Conference, 2002. Proceedings. 39th*, 2002, pp. 524-529.

[26] K. Le, "SerDes power minimization allows SoC solutions," in *EE Times*: TechInsights, 2002.

[27] A. D. Bovopoulos, "Dune Fabric Power," ptcongdon@ucdavis.edu, Ed. Davis, CA, 2009.

[28] N. Jouppi, "CACTI," HPLabs, 2008, http://www.hpl.hp.com/research/cacti/

[29] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," in *Performance Analysis of Systems and Software, 2006 IEEE International Symposium on*, 2006, pp. 120-129.

[30] Z. Kai, H. Chengchen, L. Hongbin, and L. Bin, "An ultra high throughput and power efficient TCAM-based IP lookup engine," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, pp. 1984-1994 vol.3.

[31] V. C. Ravikumar, "EaseCAM: An Energy and Storage Efficient TCAM-Based Router Architecture for IP Lookup," *IEEE Trans. Comput.*, vol. 54, pp. 521-533, 2005.

[32] F. Zane, N. Girija, and A. Basu, "Coolcams: power-efficient TCAMs for forwarding engines," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2003, pp. 42-52 vol.1.

[33] J. L. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Fourth Edition ed. San Francisco, CA USA: Morgan Kaufmann Publishers Inc., 2006.

[34] R. Pang, M. Allman, M. Bennett, J. Lee, P. Vern, and B. Tierney, "A first look at modern enterprise traffic," in *Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference* Berkeley, CA: USENIX Association, 2005.

[35] D. Bailey, T. Harris, W. Saphir, and R. van der Wijngaart, "The NAS Parallel Benchmarks 2.0," Report NAS95-020 NASA Ames Research Center, Moffett Field, CA1995.

[36] P. Congdon, M. Farrens, and P. Mohapatra, "Packet Prediction for Speculative Cut-Through Switching," in *ANCS-2008* San Jose, CA, 2008, p. 10.

[37] R. Panigrahy and S. Sharma, "Reducing TCAM Power Consumption and Increasing Throughput," in *Proceedings of the 10th Symposium on High Performance Interconnects HOT Interconnects (HotI'02)*: IEEE Computer Society, 2002.

[38] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "DevoFlow: cost-effective flow management for high performance enterprise networks," in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks* Monterey, California: ACM.

[39] S. Kasnavi, P. Berube, V. Gaudet, and J. Amaral, "A Multizone Pipelined Cache for IP Routing," *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, vol. 3462, pp. 187-198, 2005.

[40] T. Yoshinaga, S. Kamakura, and M. Koibuchi, "Predictive Switching in 2-D Torus Routers," in *Proceedings of the International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems*: IEEE Computer Society, 2006.

[41] Z. Ding, R. Hoare, A. Jones, D. Li, S. Shao, S. Tung, J. Zheng, and R. Melhem, "Switch Design to Enable Predictive Multiplexed Switching in Multiprocessor Networks," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, pp. 100a-100a.

[42] L.-S. Peh and D. W. J., "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*: IEEE Computer Society, 2001.

[43] G. Surendra, S. Banerjee, and S. K. Nandy, "On the effectiveness of flow aggregation in improving instruction reuse in network processing applications," *Int. J. Parallel Program.*, vol. 31, pp. 469-487, 2003.