© 2015 IJSRSET | Volume 1 | Issue 2 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Packet Prediction Circuitry to Reduce Latency and Power Using OpenFlow Switches

Adhirai B, Akshaya K, P Prema
Department of Computer Science, Dhanalakshmi College of Engineering, Chennai, Tamilnadu, India

ABSTRACT

The Ethernet switch is a major building block for today's enterprise networks and data centers. As network technologies congregate ahead a single Ethernet fabric,there is enduring pressure to increae the performance and efficiency of the switch while maintaining elasticity and a well-to-do position of packet processing features. The OpenFlow architecture aims to provide elasticity and programmable packet processing to meet these converging needs. Of the several ways to generate an OpenFlow switch, a popular preference is to create deep use of ternary content addressable memories (TCAMs). Regrettably, TCAMs can consume a significant amount of power and, when used to equal flows in an OpenFlow switch, put a hurdle 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 sequential position network communications to predict the flow arrangement of arriving packets. When predictions are correct, latency can be reduced, and considerable power savings can be achieved from bypassing the full lookup process. IP and Transport networks are controlled and operated independently today, leading to significant Capex and Opex inefficiencies for the providers. We discuss a unified approach with OpenFlow, and present a recent demonstration of a unified control plane for OpenFlow enabled IP/Ethernet networks. Imitation studies using actual network traces point out that correct prediction rates of 97% are achievable using only a small amount of prediction circuitry per port. OpenFlow is based on an Ethernet switch, with an internal flow-table, and a standardized interface to add and remove flow entries. Our goal is to encourage networking vendors to add OpenFlow to their switch products for deployment in college campus backbones and wiring closets. We believe that OpenFlow is a pragmatic compromise: on one hand, it allows researchers to run experiments on heterogeneous switches in a uniform way at line-rate and with high port-density; while on the other hand, vendors do not need to expose the internal workings of their switches.

Keywords: Ethernet networks, packet switching, software, Transmission Control Protocol, Transport Layer Security, Media Access Controller, TCAM, Packet Prediction Circuitry, TCAMmustbe

I. INTRODUCTION

The OpenFlow enables remote controllers to determine the path of network packets through the network of switches. At least two controllers are recommended — a primary, and a secondary as backup. This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols. Also, OpenFlow allows switches from different suppliers — often each with their own proprietary interfaces and scripting languages — to be managed remotely using a single, open protocol. Its inventors consider OpenFlow an enabler of Software defined networking (SDN).OpenFlow allows remote administration of a switch's packet forwarding tables, by adding, modifying and removing packet matching rules and actions. This way, routing decisions can be made periodically or ad hoc by the controller and translated into rules and actions with a configurable lifespan, which are then deployed to a switch's flow table, leaving the actual forwarding of matched packets to the switch at wire speed for the duration of those rules. Packets which are unmatched by the switch can be forwarded to the controller. The controller can then decide to modify existing flow table rules on one or more switches or to deploy new rules, to prevent a structural flow of traffic

between switch and controller. It could even decide to forward the traffic itself, provided that it has told the switch to forward entire packets instead of just their header. The OpenFlow protocol is layered on top of the Transmission Control Protocol (TCP), and prescribes the use of Transport Layer Security (TLS). Controllers should listen on TCP port 6653 for switches that want to set up a connection. Earlier versions of the OpenFlow protocol unofficially used port 6633.Networks have become a critical part of business and institutions. A failure in a network could become a failure in business processes and the consequent money lost. Therefore network administrators have to ensure a perfect network running close to 100% of the time. But many times researchers need real environments in which they can test experimental network protocols and usually encounter opposition from network administrators who forbid them to test their experiments in production networks. Here is where it appears the term programmable networks and where OpenFlow technology can help to solve this problematic. OpenFlow technology allows network administrators to segment telecommunication networks programming the devices involved in the system. OpenFlow devices identify different traffic flows following rules pre-configured by network managers. This technology virtualizes network into flows in a way that there are no interferences between traces. Furthermore, once the virtualization is done the network administrator can delegate the management of network segment/s to the researchers as if it was a new network. Summarizing programming networks with OpenFlow technology take following advantages:

- Network virtualization: experimental tests can be deployed into production networks without disturbing existent isolated traffic
- Network managers can delegate virtual segments to be managed by researchers
- Low cost devices: OpenFlow switches can be deployed into UNIX/Linux platforms
- OpenFlow protocol can be exploited in modern Ethernet switches/routers from different vendors as an extra functionality using TCAM

II. METHODS AND MATERIAL

Switch Architecture

A. Switch Architecture

There are many ways to create an Ethernet switch .The OpenFlow Ethernet switch architecture used in this paper Adopts line cards with physical media ports connected to switched backplane fabric. The architecture is similar to one of many described in , but with an prominence 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 furnished 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 TCAMused to support per-flow forwarding features. Fig. 1 shows the high-level switch architecture.

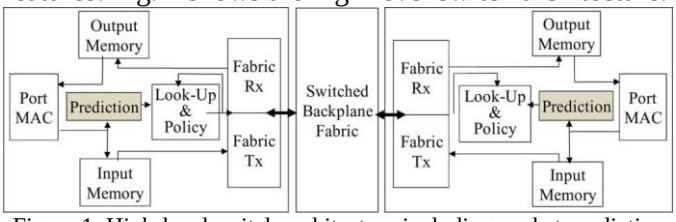

Figure 1: High-level switch architecture including packet prediction circuitry atthe input port

Packets are received at line rate from the physical media ports and put into the input memory. While the packet is being acknowledged 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 aglow-key. A flow-key is the fundamental structure used to look up 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 .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 -tuple that is defined by a setoff fields from the packet. All packets that are part of a flow are subject to the same policy and treatment by the switch.

B. Packet Prediction Circuitry

The switch architecture of Fig. 1 includes packet prediction circuitry on each way in port. The goal of the circuitry is to forecast 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 Sneaks on the input memory bus as the packet is streamed into memory. While the flow-key is take along together, significant bits from the packet are extracted and used to generate a packet signature according to a prediction method .The packet signature is searched in a local per-port prediction Cache that consist of a signature CAM, flow-key RAM, and forwarding RAM. If exactly one match is found, the packet is expected to be part of the same flow

as a previously received packet with the same signature and forwarding can begin instantaneously. This constitutes hypothetical packet forwarding that 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 sensible 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 related 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 directives for packets of the matching flow. Once a complete flow-key has been received and assembled by the packet parser, there are three possible circumstances that may occur with respect to the prediction logic and theoretical 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.

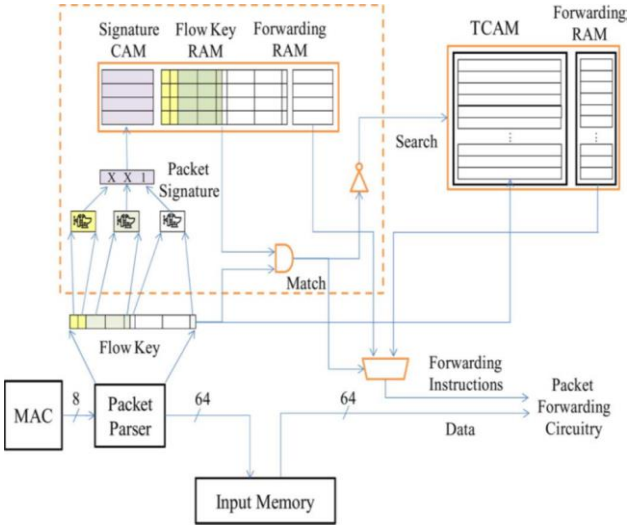

Figure 2: The per-port packet prediction circuitry snoops packet data

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. Possible dormancy 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 TCA Mmustbe 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. There are differences of opinion as to whether the temporal locality of Internet traffic is sufficient to enable optimized forwarding using caching. 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 centre traffic, for example, has been observed to exhibit an ON–OFF pattern with strong temporal locality among the packet trains.

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

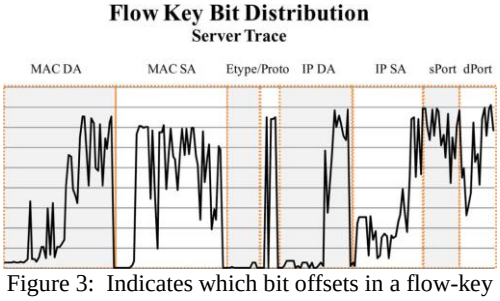

Figure 3: Indicates which bit offsets in a flow-key

Fig. 3 indicates which bit offsets in a flow-key vary the most between subsequent flow-keys in the Server Trace. 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 trade off 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 predetermined 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 mis prediction rate than methods that perform more intelligent parsing.
2) Sub-Field Hash: The Sub-FieldHashmethod intelligently parses incoming packets to extract precise subfields 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 subfields of the 29-B 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.

D. Algorithm

ENCRYPTION ALGORITHM

A mathematical procedure for performing encryption on data Through the use of an algorithm, information is made into meaningless cipher text and requires the use of a key to transform the data back into its original form. If you are not use that eclipse project means use this algorithm, it's before I gave one project that is give alternation. Otherwise you won't use this one.

SERIAL ALGORITHM

A sequential algorithm or serial algorithm is an algorithm that is executed sequentially – once through, from start to finish, without other processing executing –

as opposed to concurrently or in parallel. The term is primarily used to contrast with concurrent algorithm or parallel algorithm; most standard computer algorithms are sequential algorithms, and not specifically identified as such, as sequential is a background assumption. Concurrency and parallelism are in general distinct concepts, but they often overlap – many distributed algorithms are both concurrent and parallel – and thus "sequential" is used to contrast with both, without distinguishing which one. If these need to be distinguished, the opposing pairs sequential/concurrent and serial/parallel may be used.

E. 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.

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-Gb/s 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 B and is therefore received in 512 bit times as determined by the speed of the ingress link. 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 Ksf be the number of bits in a minimum-size Ethernet packet (which is the constant 512). Let Rp be the received port line rate in bits/s, and let be the length of the packet in bits. Let Rs be the fabric interface transfer rate in bits/s, and assume that Rf>Rp 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

Store and Forward latency

$$=(L/Rp)+(Ksf/Rp)+(L/Rf). \quad (11)$$

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.

$$D=\{D1(H1)+D2(H2)+\ldots\ldots..Dn(Hn)$$

be the set of functions in the classification process that return the starting bit displacement for the flow-key fields in .Then, Dn(Hn) is the starting bit offset for the last field necessary to create the tuple needed for the lookup. If Kct is defined as the number of bits that must be received to construct the flow-key for the lookup stage to begin, then Kctis determined as

$$Kct=Dn(Hn)+|Hn|$$

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 we have the following formula for cut-through switch latency:

Cut-Through Latency=(Kct/Rp)+(Ksf/Rp)+1/Rf

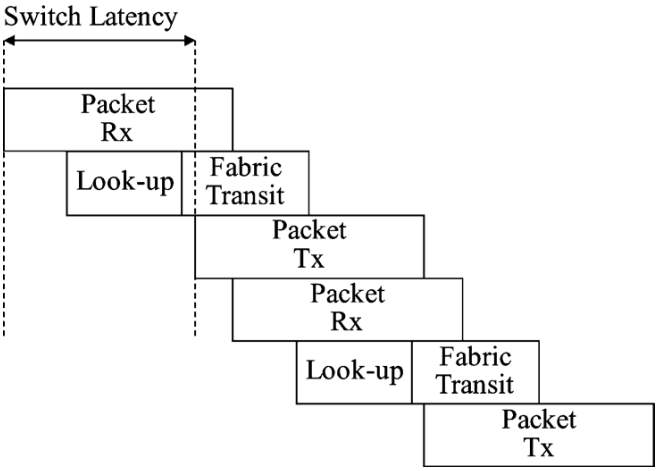

Figure 4: Switch latency for a cut-through switch pipeline.

F. Evaluation

a. Experimental Setup

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

Table 1: 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

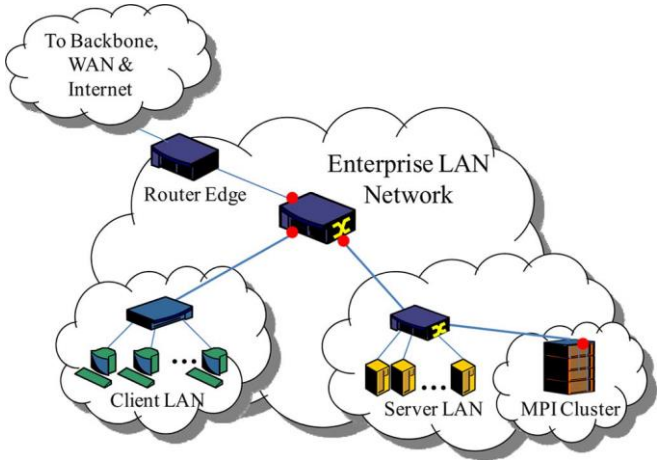

Figure 5. Network topology containing representative trace capture points.

multiplexed flows. Prediction methods are expected to be most effective in the data centre, 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 anonym zed packet header traces from October 2004 through January 2005, which are available for download from http://www.icir.org/enterprisetracing/ 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 10-GbE 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 in Section II.

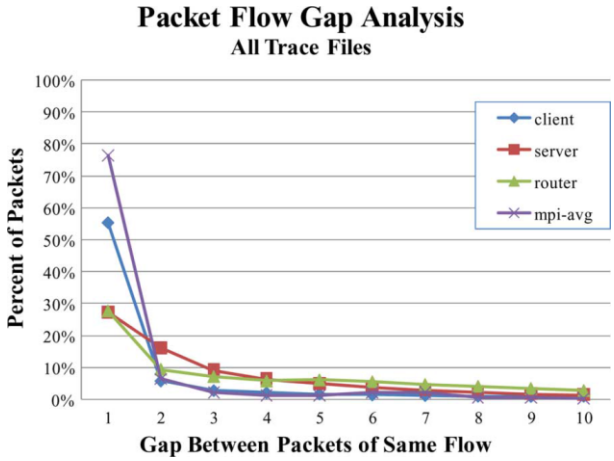

Figure 6: Packet gap analysis shows the distribution of spacing between consecutive packets of the same flow up to a spacing of 10 packets.

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 Aeronautics 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 intercluster communication used was MPI over Ethernet. Complete details of the NPB suite may be found at [35].

c. Temporal Locality of Network Traces

Fig. 8 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 previous packets 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.

III. RESULTS AND DISCUSSION

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.

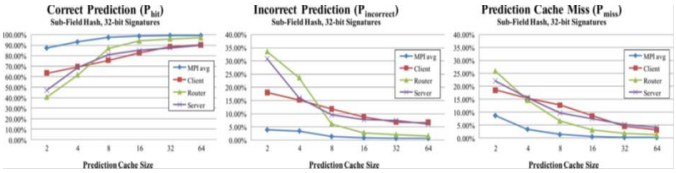

Figure 7: Prediction rates for each trace data set using the Sub-Field Hash method with 32-bit signatures and varying prediction cache size

Fig. 8 shows , and for the Sub-Field Hash method when run with all traces files. The figure confirms that the prediction circuitry is more effective when placed closer to servers in the data centre. The MPI traces have very high temporal locality, resulting in rates nearing 99%.

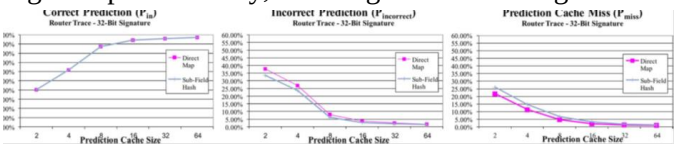

Figure 8: Prediction rates for each prediction method run against the Router trace with 32-bit signatures and varying prediction cache size

Fig. 8 compares the different prediction methods with the most diverse Router traces using 32-bit signatures. The figure 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 one 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 one 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. 9 shows that the Direct Map method has a slightly lower prediction cache miss rate with small caches % 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.

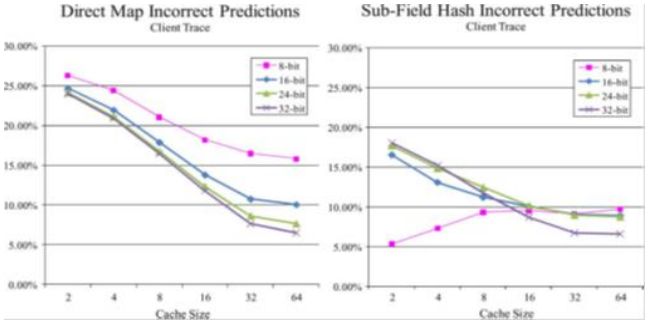

Figure 9. Difference between Direct Map and Sub-Field Hash methods when making incorrect predictions on the Client trace.

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. 9. As the number of bits used to represent a signature grows, the two methods perform similarly.

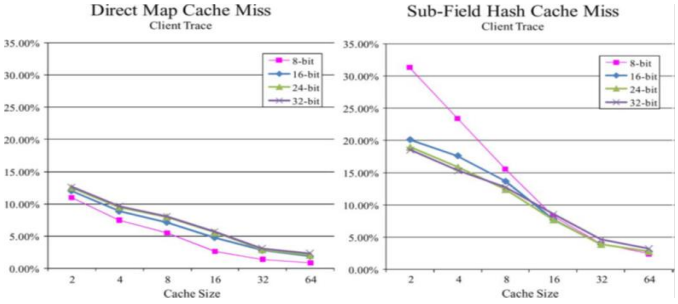

Figure 10. Comparison of prediction cache misses on the Client trace. Cache misses are preferred to incorrect predictions that need speculation cleanupdhs

Fig. 10 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. Fig. 14 shows the effectiveness of hashing over selecting predefined bits for all signature and cache sizes used with the Client trace. 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. 11 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.

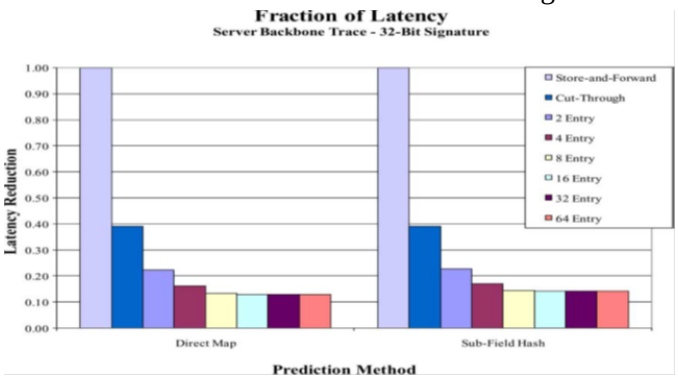

Figure 11:. Comparison of the latency reduction achieved by each prediction method on the Server traces using 32-bit signatures.

Fig. 11 shows there is very little difference between methods when it comes to latency reduction. This is because both methods have similar correct prediction rates. Where the methods differ is in the mix of incorrect predictions and cache misses as seen in Figs. 13 and 14. These differences will have a bigger impact on power reduction since incorrect predictions require aborting backplane transfers that 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.

IV. 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. The OpenFlow Switching community is

working to add Quality Of Service (QoS) criteria to the technology. It could be interesting to create an environment in which there are different flows with different QoS priorities. 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. Other important reasons why OpenFlow must be taken into account are that it can be installed on a PC or a commercial router as an added function in TCAM and that is a technology backed by major networking manufacturers such as CISCO, Juniper, HP and NEC.Finally we can conclude that Open Flow although can be still considered novel or immature, it is an interesting technology to network managers that want to isolate flows separating production and experimental traffic, or to someone who wants to design a configurable network

V. 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," Comput. Commun. Rev., vol. 38, pp. 69–74, 2008.

[2] B. Salisbury, "TCAMs and OpenFlow—What every SDN practitioner must know," SDN Central, 2012 [Online]. Available:http://www.sdncentral.com/products-technologies /sdn-openflow-tcam-need-to-know/2012/07/

[3] B. Heller, "OpenFlow switch specification," OpenFlow Consortium, 2008 [Online]. Available: 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 Proc. 17th Annu. IEEE INFOCOM, 1998, vol. 3, pp. 1240–1247.

[5] H. H. Y. Tzeng and T. Przygienda, "On fast address-lookup algorithms," IEEE J. Sel. Areas Commun., vol. 17, no. 6, pp. 1067–1082, Jun. 1999.

[6] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," IEEE J. Sel. Areas Commun., vol. 21, no. 4, pp. 560–571, May 2003.

[7] J. Liao, "SDN system performance," 2012 [Online]. Available: http:// pica8.org/blogs/?p=201

[8] R. Ozdag, "Intel ethernet switch FM6000 series—Software defined neworking," Intel Corporation, 2012, p. 8.

[9] Arista Networks, Inc., Santa Clara, CA, USA, "7150 series 1/10 GbE SFP ultra low latency switch," 2012.

[10] Cisco Systems, Inc., San Jose, CA, USA, "Cisco Nexus 3548 switch architecture," 2012.

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

[12] P. Gupta and N. McKeown, "Algorithms for packet classification," IEEE Netw., vol. 15, no. 2, pp. 24–32, Mar. 2001, 2001.

[13] D. E. Taylor, "Survey and taxonomy of packet classification techniques,"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 Proc. 9thACM SIGCOMMWorkshopHot Topics Netw.,Monterey, CA, USA, Art. no. 12.

[15] C. Minkenberg,M. Gusat, and G. Rodriguez, "Adaptive routing in data center bridges," in Proc. 17th IEEE HOTI, 2009, pp. 33–41.

[16] R. Jain and S. Routhier, "Packet trains-measurements and a new model for computer network traffic," IEEE J. Sel. Areas Commun., vol. SAC-4, no. 6, pp. 986–995, Sep. 1986.

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

[18] D. C. Feldmeier, "Improving gateway performance with a routing-table cache," in Proc. 7th Annu. IEEE INFOCOM, 1988, pp. 298–307.

[19] P. Newman, G. Minshall, T. Lyon, and L. Huston, "IP switching and gigabit routers," IEEE Commun. Mag., vol. 35, no. 1, pp. 64–69, Jan. 1997.

[20] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in Proc. ACM SIGCOMM Workshop Enterprise Netw., Barcelona, Spain, 2009, pp. 65–72.

[21] A. Partow, "General purpose hash function algorithms," 2013 [Online].Available: http://www.partow.net/programming/ hashfunctions/index.html

[22] G. Ananthanarayanan and R. H. Katz, "Greening the switch," in Proc. USENIX Conf. Power Aware Comput. Syst., San Diego, CA, USA, 2008, p. 7.

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

[24] H.-S.Wang, L.-S. Peh, and S. Malik, "Apower model for routers: Modeling alpha 21364 and InfiniBand routers," IEEE Micro, vol. 23, no. 1, pp. 26–35, Jan.–Feb. 2003.

[25] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," in Proc. 39th Design Autom. Conf., 2002, pp. 524–529.