# LOAD-BALANCING MULTIPATH SWITCHING SYSTEM WITH FLOW SLICE

**R.Iniyavani[1] , S.John Justin Thangaraj[2]**

[1] *Final Year M.E-CSE/ Selvam College of Technology/ Namakkal/Tamilnadu (India)*

[2] *Associative professor/CSE/ Selvam College of Technology/ Namakkal/Tamilnadu (India)*

**ABSTRACT**

*Multi-Path Switching systems (MPS) are intensively used in the state-of-the-art core routers. One of the most intractable issues is how to load-balance traffic across its multiple paths while not disturbing the intra-flow packet orders. In this paper, based on the studies of tens of real Internet traces, we develop a novel scheme, namely Flow-Slice (FS), which cuts off each flow into flow-slices at every intra-flow interval larger than a slicing threshold set to 1ms~4ms and balances the load on the finer granularity. Through theoretical analyses and comprehensive trace-driven simulations, we show that FS achieves impressive load-balancing performance with little hardware cost while limiting the packet out-of-order chances to a negligible level (below 10-6).*

***Keywords: Load Balancing, Switching Theory, Traffic Measurement,***

## I INTRODUCTION

Multipath Switching systems (MPS) play a pivotal role in fabricating state-of-the-art high performance core routers In general, MPS is built by aggregating several lower speed switches and, therefore, exhibits multiple internal data paths .One major open issue in MPS is the load balancing problem defined as how to distribute incoming traffic A(t) across its k internal switching paths to meet at least three objectives simultaneously:

**1.1 Uniform Load Sharing:** Traffic dispatched to each path should be uniform. Specifically in MPS, traffic destined for each output should be spread evenly to avoid output contention, minimize average packet delay, and maximize throughput.

**1.2 Intraflow Packet Ordering:** Packets in the same flow should depart MPS as their arrival orders. (Unless otherwise stated, flow in this paper is defined by TCP/IP 5-tuple.) This ordering is essential since out- of-order packets will degrade the performance of higher level protocols

**1.3 Low Timing And Hardware Complexity:** The load-balancing and additional resequencing mechanisms at MPS should work fast enough to match the line rate, and should introduce limited hardware complexity. MPS

is most likely to hold hundreds of external ports operating at ultrahigh speed. To provide such scalability, the timing/hardware complexity of O(1) is necessary.

However, packets in the same flow may be forwarded in separate paths and experience different delays, thus violating the intraflow packet ordering requirement. A straightforward solution is to use an explicit resequencer at each output to restore packet orders. They delay each packet at output until the system delay upper bound is reached. Each packet is time shifted by the same offset before departing, thus preserving the arrival order. Nonetheless, the delay equalization method suffers from a huge penalty in magnifying the average delay. It is shown in our prototype simulations that even the latest adaptive resequencer increases the average delay nearly 10 times to about 10 ms. A route passing through five such routers will lead to at least 50 ms average delay, almost violating the QoS requirement for delay-sensitive applications. To avoid the packet out-of-order, another choice is to use flow-based load-balancing algorithms. They dispatch packets in the same flow to a fixed switching path by hashing its 5-tuple to path ID.

## II RELATED WORK

A basic timestamp-based resequencer was proposed by Turner to deal with the cell out-of-order issue, when cells within a virtual circuit are allowed to take different paths in ATM switching systems. In each scheduling, the resequencer implements a smart contention resolution mechanism to select the oldest cell and then compares its age with a predefined threshold equal to the system delay upper bound. If the selected cell exceeds this age threshold, it will be sent without disturbing cell orders since all previously arrived cells have left the system. Henrion improved the timestamp-based resequencer by introducing time-wheel-like hardware which does not need to compare cells' timestamps at output. It maintains an array of D pointer lists, where D is larger than delay upper bound at system measured by timeslots. Each pointer at the list stores location of a cell waiting for resequencing. At timeslot t, the pointer list at slot t modulo D is linked to the tail of the output cell list and removed from the array. After that, pointers of arrival cells at timeslot t are linked together and stored in this empty slot (t modulo D) of the array. This approach delays every cell by fixed D timeslots with O(1) complexity and strictly guarantees cell orders. Fixed-threshold timestamp-based resequencers work well for ATM switching systems where traffic is well defined and cell delay is moderate. However, in transition to an IP network, traffic becomes rather bursty, introducing a higher age threshold on these resequencers and equalizing cell delay to undesirable level. To cope with the traffic in the worst case, Turner proposed an adaptive resequencer that dynamically adjusts threshold ac-cording to a real-time delay bound. Trace-driven simulations show that the resequencer performs well better than the fixed-threshold one. Another kind of resequencer uses sequence number instead of timestamp. Cells with the same priority from one input to one output are numbered sequentially at their arrivals. At the output, an expected sequence number is maintained for all the packets in the same flow. Only the cells with an expected sequence can be sent after reaching output, thus preserving cell order. Predefined path scheduling protocol known by both input and output, such as Round- Robin (RR), can be employed to avoid attaching a sequence number to each cell. Resequencers using VIQ can be categorized in this class. Resequencing techniques are also studied in designing popular MPS. This approach

does not disorder cells but suffers from huge communication complexity in sharing information among all the input and outputs .This approach can be categorized into timestamp-based solution. Later in , by setting the configurations of LBvN's first and second stages to symmetrical patterns, each input and output is connected to each Virtual Output Queue (VOQ) between the two stages (called mailbox) in period. Each newly arrived flow is assigned to the NP with the lightest load. The major bottleneck of this method is to maintain a per-flow context table, which reaches up to a million entries at an OC 768c port. The unique features of HRW lie in its graceful dealing with server failures, and it supports load balancing across heterogeneous servers by introducing the scaling vector. In developed methods to detect NP's overload and optimally adjust a scaling vector to reshape flow distribution. However, flow remapping has negative impact on increasing packet out-of-order probability.

## III FLOW SLICE

The traces are collected at separate locations from the network core to edge/access points; hence, the traces represent comprehensive Internet traffic behaviors. Each data set records consecutive packet headers (TCP/IP) during an observation period of 1-10 min.

The original 5-tuple at each header is remapped, using the longest prefix preserving mapping, to new 5-tuple address spaces, but they still retain the original flow-level characteristics. The timestamp at each packet header has a resolution of at least 1 microsecond.

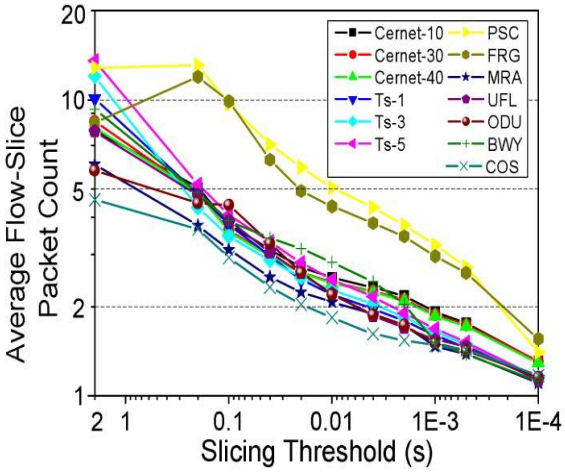


**Fig:1 Flow-Slice Packet Count**

**Flow Slice**

A flow slice is a sequence of packets in a flow, where every intraflow interval between two consecutive packets is smaller than or equal to a slicing threshold . Flow slices can be seen as miniflows created by cutting off every intraflow interval larger than . we depict the Cumulative Distribution Functions (C.D.F) of intraflow intervals in our traces.Most of the traces have more than 50 percent of their intervals larger than 1 ms, and more than 40 percent are larger than 4 ms. Two exceptions.

**Properties**

The flow-slice characteristics are investigated at different slicing thresholds between 100 s-200 ms. Compared with the original flow, which is equivalent to flow slice with a slicing threshold of 2 s (set to flow time-out value), three flow-slice specific properties are observed in all traces.

**Property 1 :** Small Size:

Both the average packet count (FC) and the average size (FS) of flow slice are much smaller than those of the original flow.

**Property 2 :** Light-Tailed Size

Distribution :Flow-slice packet count/size distributions are light tailed while it is well-known that original flow-size distribution is heavy tailed.

**Property 3** : Fewer Active Flow Slice:

The active flow-slice number is 1-2 magnitudes smaller than that of active flow.

## IV PROBLEM

The packet-based solutions are advocated where traffic is dispatched packet by packet to optimally balance the load. However, packets in the same flow may be forwarded in separate paths and experience different delays, thus violating the intraflow packet ordering requirement. A straightforward solution is to use an explicit resequencer at each output to restore packet orders. In timestamp-based resequencers are developed. They delay each packet at output until the system delay upper bound is reached. Each packet is time shifted by the same offset before departing, thus preserving the arrival order. Nonetheless, the delay equalization method suffers from a huge penalty in magnifying the average delay. To avoid the packet out-of-order, another choice is to use flow-based load balancing algorithms. They dispatch packets in the same flow to a fixed switching path by hashing its 5-tuple to path ID.

Problems**:**

1. High load balance
2. Delay delivered process
3. More waiting time

## V IMPLEMENTATION

The major implementation cost introduced by FS is that of hash table maintaining an active flow-slice context. The size of this hash table is a trade-off ,First a large hash table reduces hash collision probability. Here, hash collision is defined as the situation when more than one flow slices are hashed into the same entry, which degrades load-balancing uniformity; Second a small hash table is desirable as the table is accessed in each load-balancing operation and on-chip SRAM with relatively small size allows higher access speed. We calculate this trade-off in a Dual Hash Table (DHT) approach .Assume a hash collision probability below 0.5 percent as negligible, where only one flow

slice out of 200 is not load balanced independently. As each FS load-balancing operation only needs one query at the hash table, O(1) timing complexity is also achieved.

## VI CONCLUSION AND FUTURE WORK

A Novel load-balancing scheme, namely, Flow Slice, based on the fact that the intraflow packet interval is often, say in 40-50 percent, larger than the delay upper bound at MPS. Due to three positive properties of flow slice, our scheme achieves good load-balancing uniformity with little hardware overhead and O(1) timing complexity.The FS scheme can achieve optimal performance while keeping the intraflow packet out-of-order probability negligible (below $10^{-6}$), given an internal speedup up to two. If this delay is independent for intraflow packets and larger than the slicing threshold, the slicing probability will be about 0.5 and the average flow-slice packet count will be approximately two, coinciding with our trace observations. Actually, if only the slicing threshold is larger than the delay variation bound at all switching paths, packet order will be undisturbed. Under bursty input traffic, the delay at all switching paths may increase synchronously, leaving its delay variation bound nearly unchanged. The FS scheme is validated in switches without class-based queues. As QoS provisioning is also critical in switch designs, one of our future works will be studying FS performance under QoS conditions.

## REFERENCES

[1] Cisco CRS-1, http://www.cisco.com/go/crs/, 2011.

[2] Real Traces from NLANR, http://pma.nlanr.net/, 2010.

[3] Vitesse Intelligent Switch Fabrics, http://www.vitesse.com, 2011.

[4] A. Aslam and K. Christensen, “Parallel Packet Switching Using Multiplexors with Virtual Input Queues,” Proc. Ann. IEEE Conf. Local Computer Networks (LCN), pp. 270-277, 2002.

[5] J. Bennett, C. Partridge, and N. Shectman, “Packet Reordering Is Not Pathological Network Behavior,” IEEE/ACM Trans. Networking, vol. 7, no. 6, pp. 789-798, Dec. 1999.

[6] N. Brownlee and K. Claffy, “Understanding Internet Traffic Streams: Dragonflies and Tortoises,” IEEE Comm. Magazine, vol. 40, no. 10, pp. 110-117, Oct. 2002.

[7] Z. Cao, Z. Wang, and E. Zegura, “Performance of Hashing-Based Schemes for Internet Load Balancing,” Proc. IEEE INFOCOM, pp. 332-341, 2000.

[8] L. Carter and M. Wegman, “Universal Classes of Hashing Functions,” J. Computer and System Sciences, vol. 18, no. 2, pp. 143-154, 1979.

[9] C.S. Chang, D.S. Lee, and Y.S. Jou, “Load Balanced Birkhoff-von Neumann Switch, Part II: Multi-Stage Buffering,” Computer Comm., vol. 25, pp. 623-634, 2002.

[10] C.S. Chang, D.S. Lee, and Y.S. Jou, “Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering,” Computer Comm., vol. 25, pp. 611-622, 2002.