Journal of Engineering Science and Technology
Vol. 12, No. 11 (2017) 3057 - 3067

MIXED PATH FOR ELEPHANT FLOW AND MICE FLOW FOR DELAY REDUCTION IN SDN

KOJ SAMBYO*, ANISH KUMAR SAHA, C.T. BHUNIA

Department of Computer Science and Engineering,
National Institute of Technology, Arunachal Pradesh-791112, India
*Corresponding Author: sambyo.koj@gmail.com

Abstract

Software-Defined Networking is gaining popularity in industries as well as in the academic since its inception. Open flow is widely accepted southbound interface and is managed by Open Networking Foundation (ONF). Open flow uses the concept of backup path for packet transmission in the switches so that if primary path fails, it may use backup path without having to pass through controller so that path stability is maintained. But this leads to high miss rate in the lookup table of all the switches of the path in the network and thus adds extra overhead of miss penalty to all the switches in path (encapsulation of the packet and sending the packet to the controller through secure channel for further processing and controller sends the packet back to the switch). This adds extra latency to end delay. In this paper we proposed mixed path where elephant flow uses double (backup) path and mice flow uses single path so as to reduce miss rate in lookup table of the switches and path stability is maintained.

Keywords: SDN, Miss rate, Delay, Latency, Open flow, Backup path, Switch, Elephant flow, Mice flow.

1. Introduction

The network that we use today are vertically integrated and vendor specific. Because of vendor specific and coupling of data plane and control plan of the network, it is difficult to configure the network devices according to predefined policies. Software-Defined Networking has opened new field for the growth of networking by decoupling data plane and control plan in the network devices. It gives more programmability, automation and self-service innovation. It consists of data plane, control plan and application plane. Data plane and control plane is connected by an interface known as southbound interface and control plane is connected to application plane via northbound interface.

Nomenclatures

A	Cache Size
A_1	Line size of cache
$E(\tilde{x})$	Expectation of $\tilde{x}$
H	Size of header
M	Miss rate
M_{ave}	Average miss in a cache
$M/G/1$	Queue model where arrivals are Markovian service times have General distribution and there is single server
P	Probability that path will fail
pd	Propagation delay
p_i	Probability that the node i is unstable
q	Stability of node
R	Rate of transmission
R_0	Constant Number
T	Total time in the system
w	Waiting time
$\tilde{x}$	Mean
$\tilde{x}^2$	Second moment

Greek Symbols

α	Delay while parsing the header of the packet
Γ	Gamma function
$1-\beta$	Set of constant
θ	Semi-vertex angle of the conical nose (Fig. 1), degree
λ	Average packet arrival rate
μ	Service rate
ρ	System utilization
φ	Path stability

Abbreviations

ONF	Open Networking Foundation
SDN	Software-Defined Networking
TCAM	Ternary Content Addressable Memory

In SDN, controller responds to connection initiated by end hosts such as changes in network topology, shifts in traffic load, or messages from other controllers, by computing packet-forwarding rules. Whenever the switches send request to the controller it gives flow rules to the switches that implement the required functionality.

The controller in control plane exercises direct control in the data plane elements via southbound interface. The most prominent southbound interface is open flow [1, 2]. Open flow is an open, standards-based communications protocol. Open flow provides access to the data plane of a network switch or router, facilitating more sophisticated traffic management. The open flow protocol is standardized and managed by the Open Networking Foundation (ONF).

The Open flow architecture consists of following basic concepts. (a) The data plane is built up by Open flow-compliant switches (b) The control plane consists of one or more Open flow controller (c) data plane communicate with control plane through secure channel [3-25].

Open flow switch is simple forwarding element without control for forwarding decision. To allow fast packet forwarding with Open flow, most hardware switches uses ternary content addressable memory (TCAM) that allows the fast lookup of wildcard matches. The header fields are used to match different protocols depending on the Open flow specification.

When packet enters in the switch it looks for the match in the lookup table. If match is found then packets are forwarded, drop, modify, etc. as per action specified by the controller. If match is not found in the lookup table then packet is forwarded to the controller through secure channel and controller set the action in the switches.

Path stability is one of the major concerns in unpredictability nature of the current network if sizes of the packets are large (elephant flow) since the path would be used for longer amount of time. Thus double path (primary and secondary path) can be used for elephant flow.

Double entry for single route increases miss rate of the lookup table. Small packets (mice flow) retains in the path for small amount of time, so path stability for mice flow is not that important. Thus, single path is sufficient for mice flow.

We proposed to use mixed path, that is, single path for mice flow and double path for elephant flow so that mice flow decrease miss rate of lookup table also the stability of the path is not effected for elephant flow.

2. Literature Review

In order to reach a packet from source to destination it has to travel from different switches, hop, gateways etc. or in other words transmission path. Several sources of delay accumulate along transmission paths such as transmission delay (time to send packet onto wire), queuing delay (time the packet is buffered before it can be sent), propagation delay (time it takes to transmit the packet via wire) and processing delay (time it takes to handle the packet on the network system).

Grouping of packet flows helps to get a better view of packet transmissions in the network. The flow that has a lot of data or lives a long period is known as Elephant flow else it is referred as mice flow. Different elephant flow scheduling has been done in order to reduce latency in the network.

Hedera is centralized flow scheduling for fat-tree (Multi stage switch topology) [4, 24, 26]. It detects large flows at edge switches, and selects a path according to the result of the estimated demand of large flows i.e. it collects flow information from constituent switches, computes non-conflicting paths for flows, and instructs switches to re-route traffic accordingly.

Fincher uses stable matching theory for scheduling elephant flow in the data centre for reduction of latency and congestion using software defined networking [5].

Devo flow reduces both the intrinsic and implementation overheads of flow-based networking, by reducing load on the network, the switch control-plane, and the central controller. It allows controller to give the flow control to the switches only for significant flows especially elephant flow. It reduces the switch-controller communication between control and data planes by introducing mechanisms that allow open flow switches to make local routing decisions, which forward flows that do not require vetting by the controller by the use of wild-carded open flow rules [6].

3. Lookup Table entry in Switch

In Open flow v1.2 and higher, if switches encounter new packet, it sends a packet to the controller through secure channel, controller process the header of the packet and sends two paths (primary path and backup path) to the switches so that if the link of primary path is down then the switch may use backup path without sending request to the controller [2, 7]. Switches make an entry of primary as well as backup path in the lookup table (Double path).It leads to the higher miss rate in the lookup table of the switches because for same source to destination packet there would be two entries in the lookup table. [8-16] discuss analytical model to predict miss rate. Miss rate varies with size of lookup table in the switches [12]. The size of lookup table has no influence until and unless all the entries in the lookup table are filled. Miss rate with respect to cache size is given by [17]:

$$M = \frac{R_0}{(A_1 M_{ave})^{1-\beta}} \Gamma(\beta - 1) A^{1-\beta} \tag{1}$$

Here miss rate is given by the cache size to the power 1-β multiplied by particular set of constant. A_1 is line size of the cache, M_{ave} is average miss, A is cache size and R_0 is constant number.

Number of different source to destination path that can accommodate in the lookup table will be half in double path compare to one that use only primary path. Eq. (1) is used for evaluating miss rate in single path**,** backup path and mixed path. Figure 1 reviles that miss rate is highest in backup path compare to single path and mixed path. The values of miss rate are given in Table 1.

Table 1. Values of miss rate.

Size	Single path	Double path	Mixed path
8	0.0300	0.0424	0.0379
16	0.0212	0.0300	0.0268
32	0.0150	0.0212	0.0190
64	0.0106	0.150	0.0134
128	0.0075	0.0106	0.0095
256	0.0053	0.0075	0.0067
500	0.0038	0.0054	0.0048

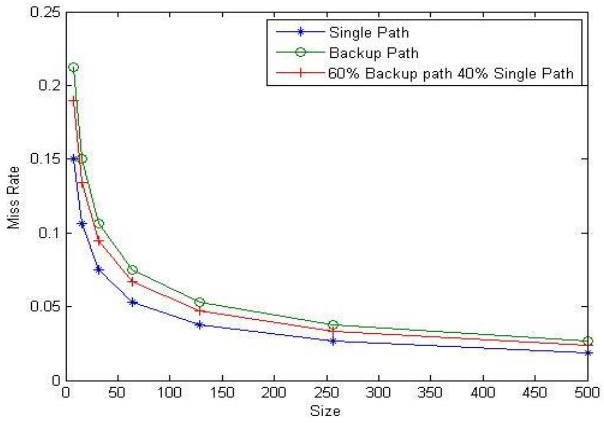

Fig. 1. Comparison of miss rate.

Increase miss rate in switches raises overall access time. Here miss penalty is delay that occurs while sending packet from switch to the controller and back to the switch.

$$AccessTime = HitTime + MissRate \times MissPenalty \tag{2}$$

While travelling from switch to controller and back to switch the packet encounter following delay:

1) **Transmission Delay:** it is the delay between the times that the first and last bits of the packet are transmitted. It depends on the link capacity or rate at which the packets are transmitted or rate at which bits are transmitted (R).

$$Transmission\ Delay = \frac{PacketSize}{R} \tag{3}$$

2) **Queuing Delay**: it is the time the packet is assigned to a queue for processing in the controller and the time it starts being transmitted. During this time, the packet waits while other packets in the transmission queue are processed.

In order to find the queuing delay let us consider packets arrive in Poisson process (commonly used model for random, mutually independent message arrivals) to the controller and service time is identically distributed with a general distribution. This gives $M/G/1$ queuing system process.

Let λ be the average packet arrival rate then

$$\lambda = \frac{Total\ Packet\ \ Arrived}{Total\ Time} \tag{4}$$

Assuming there is a server with general service time distribution with mean $\tilde{x}= E(\tilde{x})$ and second moment $\tilde{x}^2 = \mathrm{E}(\tilde{x}^2)$. The service rate is

$$\mu = \frac{1}{\tilde{x}} \tag{5}$$

The system utilization which arrival rate over service rate is given by

$$\rho = \frac{\lambda}{\mu} \tag{6}$$

Pollaczek-khinchin formula [18] states a relationship between the queue length and service time distribution for *M*/*G*/1 queue. The average waiting time of packet is given by (Pollaczek-khinchin formula):

$$w = \frac{\lambda.E(\tilde{x}^2)}{2(1-\rho)} \tag{7}$$

Total time in the system is given by:

$$T = \tilde{x} + w \tag{8}$$

$$T = \tilde{x} + \frac{\lambda.E(\tilde{x}^2)}{2(1-\rho)} \tag{9}$$

3) Propagation Delay: it is the delay between the time the last bit is transmitted at the head node of the link and the time the last bit is received at the tail node. This is proportional to the physical distance between transmitter and receiver. Since packet first travel from switch to controller and then back to switch, total propagation is given by:

$$= 2pd \tag{10}$$

4) Processing Delay: it is the delay while parsing the header of the packet and assigning the required path to the switch. Let α be the constant to read the header and H be the size of the header. Then processing delay becomes:

$$= \alpha H \tag{11}$$

Therefore, miss penalty is given by:

Miss penalty = Transmission delay + Queuing delay + Propagation delay + Processing delay (12)

$$MissPenalty = \frac{PacketSize}{R} + (\tilde{x} + \frac{\lambda.E(\tilde{x}^2)}{2(1-\rho)}) + (2pd) + (\alpha H) \tag{13}$$

Result of run of Eq. (13) is portrayed in Fig. 2. The values of access time are given in Table 2. It is found that the access time is highest for the path that have highest miss rate.

Table 2. Values of access time.

Size	Single path	Double path	Mixed path
8	0.0300	0.4243	0.3795
16	0.2121	0.3000	0.2683
32	0.1500	0.2121	0.1897
64	0.1061	0.1500	0.1342
128	0.0750	0.1061	0.0949
256	0.0530	0.0750	0.0671
500	0.0379	0.537	0.0480

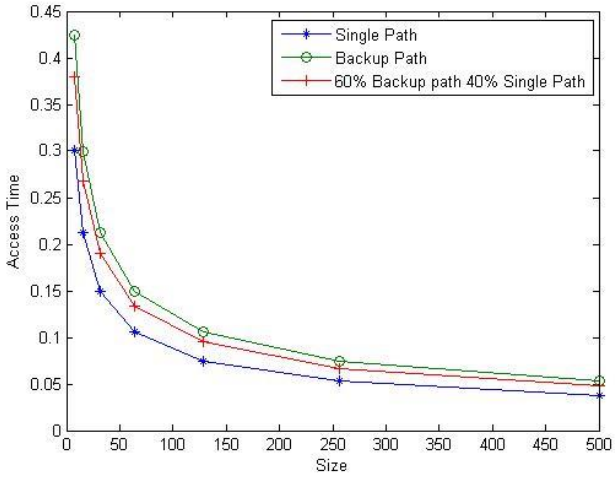

Fig. 2. Comparison of access time.

4. Path Stability

The average access time for the packet from source to destination depends on path stability. If the path fails or is broken then for single path whole process of path selection has to be made and in backup path if the primary path fails then backup path can be used thus saving time for path selection process. Path stability has been showed by Susmit Maity et al. [19].

Let p_i be the probability that the node i is unstable and let pause time be the time duration for which the node remain stationary then p_i is given by:

$$p_i = \frac{TotalMotionTimeof\ Node\_i}{TotalMotionTimeof\ Node\_i + TotalPauseTimeof\ Node\_i} \tag{14}$$

From above Eq. 14, stability of a node can be given by:

$$q_i = 1 - p_i \tag{15}$$

Let q_i and q_j denotes the stability of node *i* and *j*. Then stability of link is given by:

$$q_{ij} = q_i q_j = (1 - p_i)(1 - p_j) \tag{16}$$

Let φ_k denote the stability of path *k* and q_{kij} be the stability of link *ij* along the path *k* or the path stability is given by:

$$\varphi_k = q_{k12} * q_{k23} * \ldots * q_{k(1-n)n} = \prod_{\substack{i=1 \\ j=i+1}}^{n-1} q_{kij} \tag{17}$$

Now let P_1 be the probability that single path will fail

$$P_1 = 1 - \varphi_0 = 1 - \prod_{\substack{i=1 \\ j=i+1}}^{n-1} q_{0ij} = 1 - \prod_{\substack{i=1 \\ j=i+1}}^{n-1} q_{0i} * q_{0j} \tag{18}$$

Similarly let P_2 be the probability that both the primary as well as backup path fails

$$P_2 = (1-\varphi_1) * (1-\varphi_2) \tag{19}$$

$$= (1-\prod_{\substack{i=1 \\ j=i+1}}^{n-1} q_{1ij})(1-\prod_{\substack{i=1 \\ j=i+1}}^{n-1} q_{2ij}) \tag{20}$$

Assuming the stability of each link is q then we have the probability as:

$$P_1 = (1-\prod_{i=1}^{n-1} q) = 1-q^{n-1} \tag{21}$$

and

$$\begin{aligned} P_2 &= \left(1-\prod_{i=1}^{n-1} q\right)\left(1-\prod_{i=1}^{n-1} q\right) \\ &= (1-q^{n-1}) * (1-q^{n-1}) \\ &= (1-q^{n-1})^2 \\ &= (P_1)^2 \end{aligned} \tag{22}$$

Equation 22 shows that probability of path breakage is higher in case of single path compare to double path as $(P_1)^2 \leq P_1$ since $(1- q^{n-1}) \leq 1$.

Eq. (13) shows that the access time of backup (double) path of a switch is higher in single path and Eq. (22) indicates that the probability of path instability is higher in single path then backup (double) path. That means for a mice flow, single path can give quicker access time as miss rate will be low and for elephant flow double (backup) path can give better packet delivery as the probability of path breakage in double (backup) path is low. Therefore we propose that for elephant flow double (backup) path to be used and for mice flow single path has to be used so that there is decrease in average latency. Identification of elephant flow and mice flow is given in [20-23]. Thus, average access time can be given by:

$$Avg\_Access\_Time = (Instability \times Hit\ Rate + \alpha) + (Miss\ Penalty \times Miss\ Rate) \tag{23}$$

In Eq. (23), α is the time taken by switches to detect that the path has been broken.

For mixed path the average access time is given by:

$$\begin{aligned} &Avg\_Access\_Time = 0.6[(Instability\ of\ Backup\ Path \times Hit\ Rate + \alpha) \\ &0.4[Instability\ of\ single\ path * Hit\ rate + + Miss\ penalty \times Miss\ rate] + \\ &Miss\ penalty \times Miss\ Rate] \end{aligned} \tag{24}$$

The value of α would be zero in the single path as if the path breaks switch has to reconnect to the controller. Taking the value of α as 0.002 and different path instability value the portrait of run of the Eqs. (23) and (24) is given in Fig. 3. The values for average access time are given in Table 3.

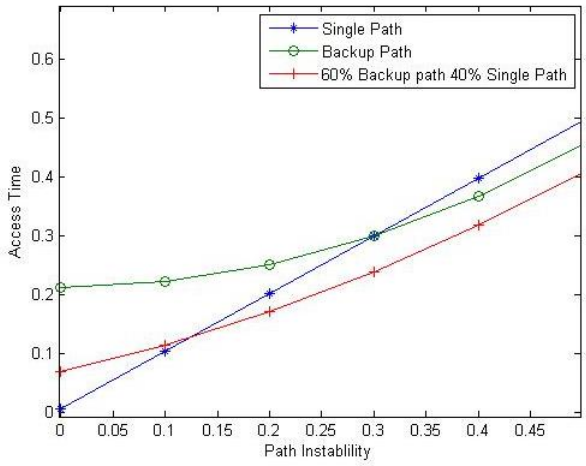

Fig. 3. Comparison of average access time.

Table 3. Values of average access time.

ath instability	Single path	Double path	Mixed path
0.05	0.06	0.21	0.09
0.1	0.105	0.215	0.12
0.15	0.15	0.22	0.14
0.2	0.201	0.25	0.17
0.25	0.25	0.28	0.201
0.3	0.3	0.3	0.24
0.35	0.35	0.34	0.28
0.4	0.4	0.37	0.32
0.45	0.45	0.41	0.36

It is found that the average access time of single path is lowest when the path is stable as the miss rate is lowest compare to backup path and mixed path but as instability increases access time of mixed path decreases because single path is unstable and double path has high miss rate and thus has to bear the burden of miss penalty that will increase the access time.

5. Conclusion

Reactive routing in SDN for single path, double or backup path and mixed path has been studied considering the lookup table entry and the path stability of the switches. It has been found that for mice flow in the network single path gives less average access time as miss rate in the lookup table entry is less as compare to double path but for elephant flow the double path gives better result as path stability play important roll. As double path has better stability compare to single path. Mixed path gives less average access time than single path and double path as practical network consist of combination of elephant flow and mice flow.

References

1. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; and Turner, J. (2008). Open flow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2), 69-74.
2. Open networking foundation (2015). Retrieved August 3, 2015, from: https://www.opennetworking.org/
3. Bairwa, S.; Memon, S.; Wakhet, O.; Sambyo, K.; Saha, A.K.; and Bhunia, C.T. (2015). Review of software defined networking for latency reduction in cloud computing. *Discovery*, 43(196), 32-38.
4. Al-Fares, M.; Radhakrishnan S.; Raghavan B.; Huang N.; and Vahdat, A. (2010). Hedera: Dynamic flow scheduling for data center networks. *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*. San Jose, California, 19-19.
5. Zhang, Y.; Cui, L; Chu, Q. (2015). Fincher: elephant flow scheduling based on stable matching in data center networks. *IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. Nanjing, China, 1-2.
6. Curtis, A.R.; Mogul, J.C.; Tourrilhes, J; Yalagandula, P.; Sharma, P.; and Banerjee, S. (2011). DevoFlow: scaling flow management for high-performance networks. *SIGCOMM'11*. Toronto, Canada, 254-265.
7. Braun, W.; and Menth, M (2014). Software-defined networking using open flow: protocols, applications and architectural design choices. *Future Internet*, 6(2), 302-336.
8. Chow, C.K. (1974). On optimization of storage hierarchies. *IBM Journal of* Research *& Development*, 18(3), 194 - 203.
9. Chow, C.K. (1976). Determination of cache's capacity and its matching storage hierarchy. *IEEE Transactions on Computers*, C-25(2), 157 - 164.
10. Harper, J.S.; Kerbyson, D.J.; and Nudd, G.R. (1999). Efficient analytical modelling of multi-level set-associative caches. *Proceedings of the International Conference HPCN Europe* '99. Amsterdam, Netherland, 473-482.
11. Przybylski, S.; Horowitz, M.; and Hennessy J. (1988). Performance tradeoffs in cache design. *Proceedings of the 15th Annual International Symposium on Computer Architecture*. Honolulu, USA, 290-298.
12. Przybylski, S.; Horowitz, M.; and Hennessy, J. (1989). Characteristics of performance-optimal multi-level cache hierarchies. *Proceedings of the 16th Annual International Symposium on Computer Architecture*. Jerusalem, Israel, 114-121.
13. Rao, G.S. (1978). Performance analysis of cache memories. *Journal of the ACM* (*JACM*) 25(3), 378-395.
14. Saltzer, J.H. (1974). A simple linear model of demand paging performance. *Communications of the ACM*, 17(4), 181-186.
15. Smith, A.J. (1982) Cache memories. *ACM Computing Surveys*, 14(3), 473-428.
16. Macdougall, M.H. (1984). Instruction-level program and processor modeling. *Computer*, 17(7), 14-24.

17. Hartstein, A.; Srinivasan, V.; Puzak, T.R.; and Emma, P.G. (2006). Cache miss behavior: is it√2?. *Proceedings of the 3rd Conference on Computing Frontiers*. Ischia, Italy, 313-320.
18. Lakatos, L. (2008). A note on the Pollaczek-Khinchin formula. *Annales Univ. Sci. Budapest., Sect. Comp*., 29, 83-91.
19. Maity, S.; Saha, S.; Sahnawaj, S.K.; Saha, B.K.; and Bhunia, C.T. (2008). Pre-emptive dynamic source routing: a reapired backup approach and stability based dsr with multiple routes. *Journal of Computing and Information Tehnology-CIT*, 16(2), 1-9.
20. Guang, C.; and Jain, G. (2008). Online identifying elephant flows through a scalable non uniform sampling algorithm. *11th IEEE International Conference on Communication Technology Proceedings*. Hangzhou, China, 525-528.
21. Platenkamp, R. (2007). Early identification of elephant flows in internet traffic. *Proceeding of 6th Twente Student Conference on IT*.
22. Azzana, Y.; Chabchoub, Y.; Fricker, Guillemin, C.F. and Robert, P. (2009). Adaptive algorithms for the identification of large flows in IP traffic. Work done during PhD preparation at INRIA Paris – Rocquencour, 1-18.
23. Lu, Y.; Wang, M.; Prabhakar, B; and Bonomi, F. (2007). Elephant trap: a low cost device for identifying large flows, *15th IEEE Symposium on High-Performance Interconnects*. Stanford, USA, 99-105.
24. Liu, S.; Xu, H.; and Cai, Z. (2014). Low latency datacenter networking: A short survey. *arXiv:1312.3455v2* [*cs.NI*], 1-6.
25. Sambyo, K.; and Bhunia, C.T. (2014). Application of multi-level ATM in reducing latency in clouds for performance improvement of integrated voice, video and data services. *11th International Conference on Information Technology: New Generations*. Las Vegas, USA, 607-607.
26. Briscoe, B.; Brunstrom, A.; Petlund, A.; Hayes, D.; Ros, D.; Tsang, I.; Gjessing, S; Fairhurst, G; Griwodz, C.; and Welzl, M. (2014). Reducing internet latency: a survey of techniques and their merits. *IEEE Communications Surveys & Tutorials*, 18(3), 2149-2196.