On Multipath Routing Algorithm for Software Defined Networks

Siddharth S. Kulkarni and Venkataramana Badarla†
Indian Institute of Technology Jodhpur, Rajasthan 342011

***Abstract*—Software Defined Networking (SDN) is an emerging phenomenon in computer networking. SDN splits control element and forwarding element of a network. It has characteristics such as centralized management, programmability, more security, granularity and reliability. Therefore SDN is suitable for data center, cloud computing and carrier networks. As reliability and fault tolerance are main concerns, these networks inherently add multiple paths between any two nodes in the network, thus creating opportunities for potential gain in network capacity (throughput). Back-pressure algorithm which uses all possible paths between the source and destination nodes is proven as a throughput optimal routing and scheduling algorithm. However, it exhibits excessively high delays and jitter. In order to realize the key benefit of gain in network capacity, in this work we propose to use back-pressure routing in software-defined networks. However, we address the problem of high delays and jitter associated with back-pressure algorithm by using centralized control and monitoring feature of SDN. The modified back-pressure algorithm is evaluated through extensive simulations and observed a significant reduction in delays while retaining throughput optimality of the basic back-pressure algorithm.**

I. Introduction

Software Defined Network (SDN) is architectural rearrangement of traditional networking. In SDN key focus is on decoupling of control from hardware. Traditionally Internet is built on basic principle of distributed system. SDN chose different path with separation of control plane from forwarding devices as shown in Figure 1. After separation of control plane and data plane at each forwarding device, control plane is consolidated into one logical centralized entity. This centralized entity can control multiple data plane elements with single control software.

SDN provides granular control, detailed and real time traffic statistics, abstract view of network which provides common APIs and easy management. SDN achieves these with centralized control plane, open interface to forwarding devices and uniform data plane. Routing in SDN is different from traditional routing where distributed systems collaboratively create routing table. In SDN based system, routing is a service or software that uses network APIs provided by controller.

A. Back-pressure Routing and Scheduling Algorithm

Back-pressure routing [2] is a dynamic and multipath routing algorithm works based on congestion gradients at each node. For each flow at a node, it computes an utility with respective to all direct neighbors of the node, where the utility is a function of queue differential backlog of the flow and link rates to the neighbors. Then the link that yields highest utility is selected as outgoing link for transmission. Each node periodically computes the utility based on current queue gradients and accordingly decides the next hop for each flow. Each node maintains a separate queue for each destination. Major benefits of back-pressure routing are i) maximal network throughput, ii) simplicity and robustness and iii) no prior information is needed about traffic characteristics. However, as back-pressure routing considers simply differential backlog in making the routing decisions, the packets get routed in all possible directions at the intermediate nodes. Due to this phenomenon many packets take different paths. Therefore high packet reordering an unbounded delays take place.

†Corresponding author, Email: ramana@iitj.ac.in

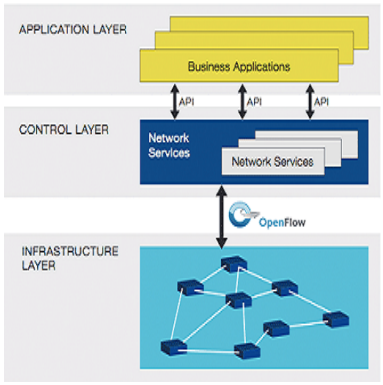


Fig. 1. Logical view of Software Defined Network [1]

B. Motivation and Contribution

Data centers are having East-West traffic which is much higher than North-South traffic and the ratio is simultaneously increasing. In traditional networking, control and statistics are two major problems which makes it very challenging to use of multipath routing in these networks. Due to introduction of SDN in data center networking we are able to control network from a single point and obtain real time statistics. SDN controller provides information about granular control, network topology and link statistics which is not available or difficult to obtain in traditional networking.

Motivation behind this work is enabling existing multipath routing techniques with SDN-based data center infrastructure, thus gaining network capacity. Major drawback with back-pressure routing algorithm is unbounded delays due to high packet reordering. In this work we address the issue of high delays in back-pressure routing with the help of centralized

management feature in software defined networking. Thus this work contributes in improving the performance of back-pressure routing in software defined networks. By obtaining optimal throughput while at reduced packet delays and jitter, this work is highly beneficial for the data center networks as the data migration takes place in a large scale in these networks.

II. Related Work

A. Routing in SDN

Software driven WAN (SWAN) [3] is a traffic engineering scheme based on software defined network architecture. It achieves high utilization in inter data center networks. Traditional traffic engineering schemes like MPLS TE lead to two major problems i.e., poor efficiency and poor sharing which are due to lack of coordination between peers. SWAN uses SDN for high utilization in inter DC networks. SWAN has a central controller which controls all activity inside network. It uses service demands from service brokers and network information from network agents for computing resource allocation. Controller notifies change in allocations to service brokers. Service broker estimates and notify service requirements to controller. Network agents provide information about traffic and topology to controller.

Software defined internet exchange [4] is a new mechanism for inter domain routing in the Internet. Traditional inter-domain routing primarily uses Boarder Gateway Protocol (BGP). BGP is difficult to manage and it is a complex protocol. There is an internet exchange point resides between routing domains. Software defined internet exchange uses SDN controller as a internet exchange point. This structure simplifies manageability of inter-domain routing. Major use cases are application based domain peering, granular policy enforcement in inter-domain routing, dynamic traffic engineering and time of day routing.

B. Back-pressure Routing Variants

Shadow queue back-pressure routing with minimum resource usage [5] uses shadow queues instead of real queues to reduce complexity and delays. To prefer the paths leading towards the destination, it discounts queue gradient by a constant factor of M. However, at low arrival rates the delays will be much higher to basic back-pressure routing and further selecting a value for discount factor M is difficult for any network.

Shortest path combined with back-pressure routing [6] maintains as many as n queues (where n is network diameter) at each intermediate node for each flow and ensures that the packets of a flow will reach the respective destination after passing through at-most n hops. Among all back-pressure variants, packet delay is minimum for this method. However, it has excessively high queue overhead.

Fountain codes based back-pressure [7] uses networking coding mechanism to regenerate packets which are delayed or lost at receiver side. Therefore inherent redundancy minimizes reordering delay.

Voting based back-pressure [8] uses a memory element *n* each interface which counts incoming packets from that interface. Each node uses this information while taking forwarding decisions. Limitations are modification to datapath protocol is required and static parameters does not work properly.

Layered back-pressure routing is a clustering based back-pressure routing protocol. Inter cluster topological distance is used with queue gradient while taking routing decisions. It is highly network dependent, initialization step takes more time and not suitable for non stationary nodes.

back-pressure collection protocol [9] uses last in first out (LIFO) buffers for delay minimization. Basic intuition behind LIFO buffer is imagining water cascading down the queue back-pressure gradient that is built up in steady state. Limitation of this method is some early packets get trapped inside LIFO buffer indefinitely.

III. Proposed Multipath Routing Algorithm for Software Defined Networks

To realize the potential gain in network capacity in software defined networks, this paper proposes a variant of back-pressure routing which takes advantage of centralized management feature from SDN and augments shortest path distances with basic back-pressure routing. This section talks about limitations of back-pressure routing as well as design details of the proposed algorithm.

In basic back-pressure routing, delay is a result of excessive looping and stalling of packets. Packet stalling happens when packets are routed over the paths (due to higher gradient) that never lead to destination. In other case, when there exists routing loops in the network, the packets may be routed inside the routing loop (due to higher gradient) several times, before routing over paths leading to the destination.

A. Proposed Algorithm

In basic back-pressure routing, the high delays are due to excessive looping and stalling of packets inside network. To avoid such looping and stalling of packets, we augment a distance based metric with queue gradient. However, direct comparison between these two entities will not be effective as they have different scales of magnitudes. Hence, we normalize distance metric and queue gradient and call these as *direction coefficient* and *normalized queue gradient*, respectively.

Normalized queue gradient: Normalization of queue gradient of a flow f at a node n is done with respective to the highest queue gradient for that flow among all direct neighbors of node n. As shown in Figure 2, node A is the source and F is the destination. Node B has neighbors as nodes A, C, D. In traditional back-pressure routing, node B calculate queue gradient with each neighbor node as $Q^F_{BA} = q_B - q_A$, $Q^F_{BC} = q_B - q_C$, $Q^F_{BD} = q_B - q_D$. However, these queue gradients will be normalized to combine the shortest path distance metric. Each node calculates maximum queue backlog in its direct neighborhood as shown in Equation 1.

$$maxq_B = max(q_A, q_B, q_C, q_D) \tag{1}$$

Then normalized queue gradient for node B is obtained as show in Equation 2.

$$NQ^F_{Bi} = \forall i \left(\frac{Q^D_{Bi}}{maxq_B} \right) \textit{ where i is neighbor of node B} \quad (2)$$

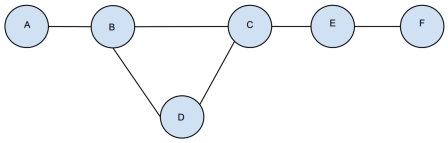


Fig. 2. A six node topology for direction coefficient calculation

Direction coefficient: Direction coefficient for a flow at a node with respect to a neighbor is computed as the ratio of shortest path distance from the node to the destination and shortest path distance from the neighbor to the destination. The ratio will be greater than one if neighbor is on shortest path (i.e., a preferred next hop) and it will be smaller than one if the neighbor is in opposite direction (i.e., unpreferred neighbor). In SDN, the shortest path distances can be obtained with a less effort.

For example consider the topology shown in Figure 2. Assume that shortest path distance from node A to destination D as SP^D_A; each neighbor i of node A has shortest path length N_iSP^D. Then direction coefficient for each neighbor is computed as shown in Equation 3.

$$C^D_{Ai} = \left(\frac{SP^D_A}{N_iSP^D} \right) \textit{ where i is neighbor of node A} \quad (3)$$

Equation 4 combines the two entities, normalized queue gradient and direction coefficient.

$$W^D_{Ai} = \left(NQ^D_{Ai}\right) - \left(1 - C^D_{Ai}\right) \quad (4)$$

Direction coefficient plays a greater role in Equation 4. It will modify weight parameter W^D_{Ai} according to link direction to the destination. It gets subtracted from normalized queue gradient if link is going in opposite direction of destination. If direction is aligned with destination direction then direction coefficient is added to normalized queue gradient. This way it enables the node to choose right link for packet transmission. Calculation of direction coefficient and normalized gradient is explained in algorithm 1. Algorithm 2 calculates maximum utility and decides forwarding node.

Algorithm 1: Proposed algorithm for calculation of direction coefficient and normalized queue gradient

Data: f is commodity, N is a set of all nodes in network, NEI_i is a set of all neighbors of node i

begin
  for $i \in N$ **do**
    for $j \in NEI_i$ **do**
      Calculate queue difference
      $Q^f_{ij} = \left(q^f_i - q^f_j\right)$;
    $maxq_i = max\,[$*queue size of all neighbors*$]$;
    $NQ^f_{ij} = \forall j \left(\frac{Q^f_{ij}}{maxq_i} \right)$;
    for $j \in NEI_i$ **do**
      Calculate direction coefficient for link i, j;
      $C^f_{ij} = \left(\frac{SP^f_i}{N_jSP^f} \right)$;
    for $j \in NEI_i$ **do**
      Calculate weight for each link i, j;
      $W^f_{ij} = \left(NQ^f_{ij}\right) - \left(1 - C^f_{ij}\right)$;
end

Algorithm 2: Proposed algorithm maximum utility calculation

Data: f is commodity, N is a set of all nodes in network, NEI_i is a set of all neighbors of node i, R_{ij} is rate of link (i, j)

begin
  for $i \in N$ **do**
    for $j \in NEI_i$ **do**
      Utility for commodity f;
      $U^f_{ij} = W^f_{ij} * R_{ij}$;
  for $i \in N$ **do**
    for $j \in NEI_i$ **do**
      Maximum utility for link i, j;
      $max \forall f \left[U^f_{ij}\right]$;
end

IV. Performance Analysis

As already mentioned, this work considers data center networking as a target application. Hence, we evaluate our proposal on popular topologies followed in data center networking. SDN can coexist with any topology that is listed below. We developed a time driven simulation environment in C language. Simulation has node structure, packet structure and link structure. Node structure contains transmission queues, reorder buffers, neighbor information and counters. Packet structure contains sequence number, time-stamp and various counters for hop count and delay calculation at receiver. Link structure has source node, destination node and bandwidth. Network topology and packet arrival rate of flow are given as input to Simulator. All delay parameters are calculated with reference to a global simulation counter. Several data flows are added to simulate realistic network. Each flow is identified by source and destination addresses and packet arrival rate. Using *max flow algorithm* (Ford-Fulkerson), the maximum capacity of a given network is calculated and accordingly arrival rates are selected.

Performance metrics are as listed below.

- Delay: It has two major parts; first end-to-end delay and second part is reordering delay. Delay is calculated with reference counter in simulation. Reordering delay is time required to make in order delivery to upper layers. Both delays are representing efficiency of routing techniques.
- Hop count for a packet is defined as number of nodes traversed by the packet. This parameter shows how

much looping is taking place inside network.

- Delay jitter is defined as a difference of delay experienced by two consecutive packets sent from source.
- Average queue occupancy is an indirect major of looping inside network.
- Average reorder buffer size represents the amount of packet reordering introduced by the network for a given arrival rate.

Simulation studies are conducted for widely used topologies in data center networking and performance metrics are computed for varying offered loads (λ) that range from 25% to 95% of maximum network capacity. The performance of proposed algorithm is compared against that of the following two routing techniques a) basic back-pressure routing and b) voting based back-pressure routing. Each simulation study is conducted for 10 random seeds and results are obtained with a confidence interval of 95%. The number flows simultaneously running in three tier, fat-tree, jellyfish topologies are 5, 6 and 6, respectively. Each flow runs between a random source and destination pair and sends packets at a given arrival rate λ throughout the simulation time.

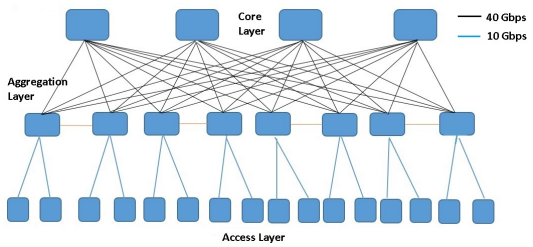


Fig. 3. Three tier DCN architecture

TABLE I. AVERAGE QUEUE OCCUPANCY (IN NUMBER OF PACKETS) OF THREE TIER DCN

Arrival Rate (λ)	BP	Voting BP	Proposed BP
25%	5.46	2.82	2.01
50%	6.47	6.20	2.48
75%	7.45	7.13	2.97
95%	10.56	10.32	4.8

TABLE II. AVERAGE REORDER BUFFER SIZE (IN NUMBER OF PACKETS) OF THREE TIER DCN

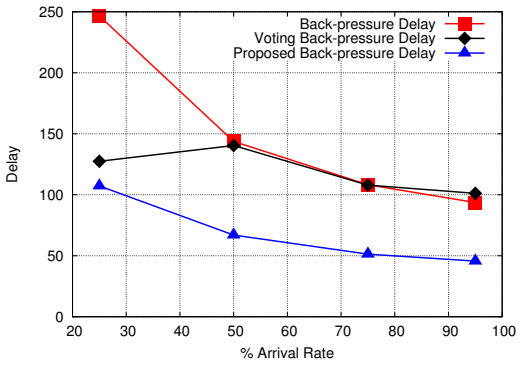
Arrival Rate (λ)	BP	Voting BP	Proposed BP
25%	575	287	219
50%	816	795	296
75%	932	980	365
95%	2811	2534	846

A. *Three Tier Data Center Network (DCN)*

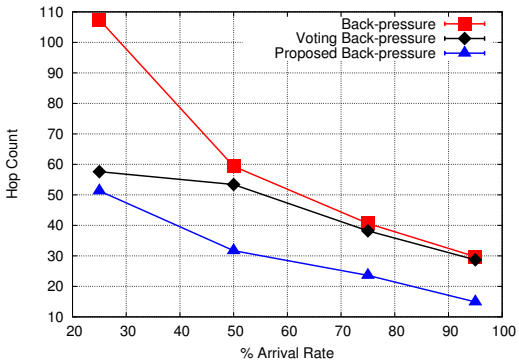
Three tier DCN [10] architecture is most widely used in data centers. As name suggests it has three layers namely (a) access layer (b) aggregation layer (c) core layer as shown in Figure 3. Access layer is connected to servers in data centers. The aggregate layer switches interconnects multiple access layer switches together. All of the aggregate layer switches are connected to each other by core layer switches. Core is junction point of data center where servers communicate with each other as well as connect to the Internet. Three tier architecture is tried and tested model in client server model. Major problems with three tier architecture are lack of fault tolerance, limited scalability and lower cross-section bandwidth.

By taking the capacity of the links shown in the topology as 10 Gbps, simulations are carried out for different offered loads (i.e., arrival rates) that range from 25% to 95% of maximum network capacity. Table I shows average queue occupancy for the three routing algorithms. We can observe that the average queue occupancy for the proposed algorithm is significantly low. Similar patterns are observed in reordering buffer sizes (in number of packets) which are presented in Table II. We can observe about 40% reduction in reordering buffer size for the proposed algorithm over the other two routing techniques.

As we can observe in Figure 4, the proposed algorithm also shows superior performance over the other two routing techniques in terms of significant reduction in packet delays and hop length. Figure 5(a) shows cumulative distribution of delay of competing methods, where we can observe that using the proposed algorithm about 90% packets reach the destination within 75 time units, while the other routing algorithms achieve the same in 200 time units. In Figure 5(b) we can notice that, like basic back-pressure routing, the proposed algorithm also obtains optimal throughput.

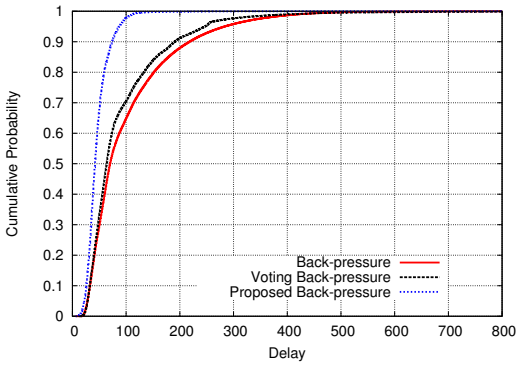


(a) Packet delay

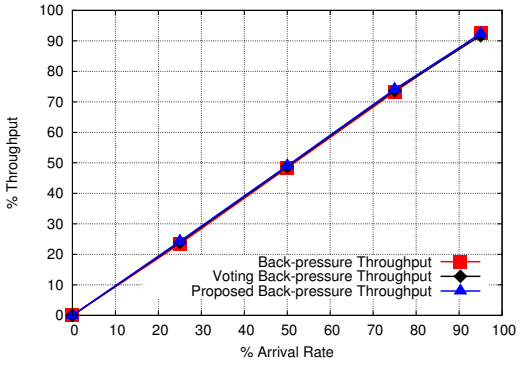


(b) Hop count

Fig. 4. Packet delay and hop count comparison for three tier DCN



(a) Cumulative distribution of delay



(b) Throughput optimality

Fig. 5. Delay and throughput comparison for three tier DCN

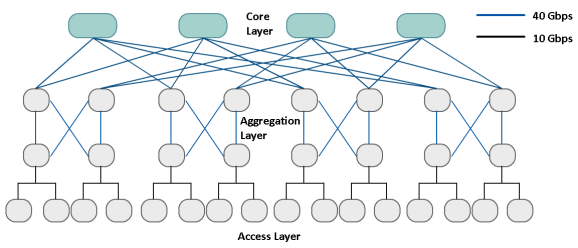
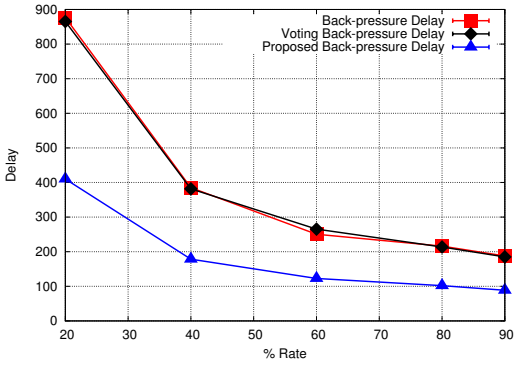


Fig. 6. Fat-tree DCN architecture

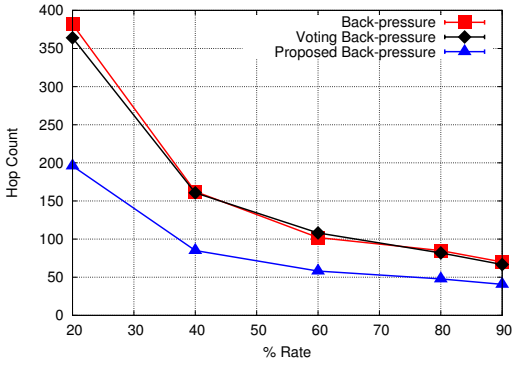
B. Fat-Tree DCN

Fat-tree [11] based data center network architecture is proposed in 2008. It offers more cross-section bandwidth with less infrastructure. It has three layers same as three tier DCN topology. Fat-tree is used in modern day data centers. It solves cross-section capacity problem and over subscription problem of three tier DCN. Fat-tree is highly scalable and cost effective.

Delay and hop count results are plotted in Figures 7(a) and (b). As we can observe that proposed algorithm outperforms the other two routing algorithms by showing about 50% reduction in packet delay and hop count. Similar trends and performance gains were noticed for other metrics also. For the purpose of brevity, we avoid presenting these results.



(a) Packet delay



(b) Hop count

Fig. 7. Delay and hop count comparison for Fat-tree DCN

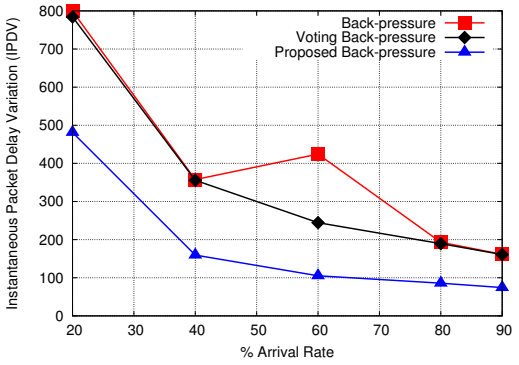


Fig. 8. Delay jitter for Fat-tree DCN

Figure 8 presents the results of delay jitter. We can observe that proposed algorithm shows about 40% reduction in jitter over the other two routing algorithms.

C. Jellyfish DCN

Jellyfish [12] is a latest work in area of data center networking architecture. It is designed to address data center requirements and growth. It is random graph based architecture therefore we can add more devices if we want to expand data center capacity. It is a flat network system where each node is

on the same layer. Randomized graph based topology makes jellyfish more agile. It is more scalable and dynamic. Jellyfish is having flat structure therefore manageability is a concern.

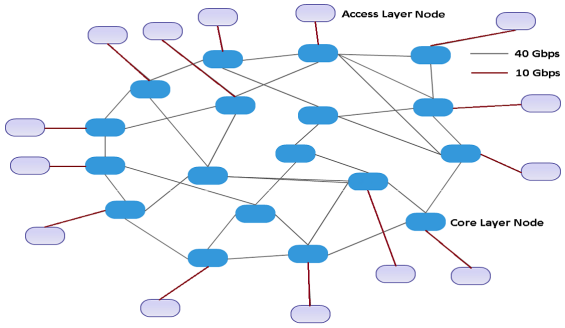
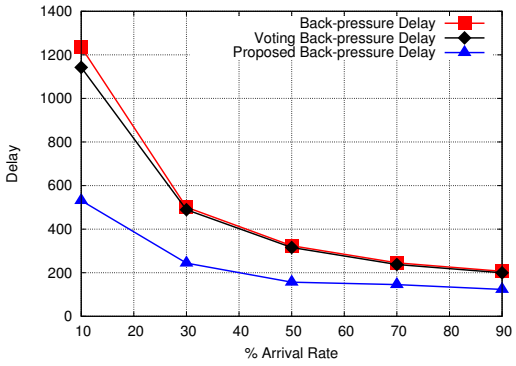
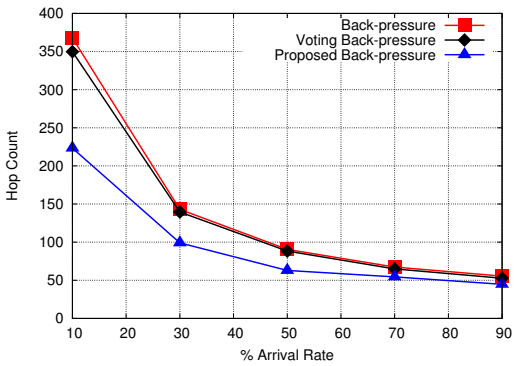


Fig. 9. Jellyfish DCN architecture



(a) Packet delay



(b) Hop count

Fig. 10. Delay and hop count comparison for jellyfish DCN

In jellyfish topology we observed similar results as observed previously. Reduction in average delay and hop count is decreased by 40% which can be seen in Figures 10(a) and (b). Similar trends and performance gains were noticed for other metrics also. For the purpose of brevity, we avoid presenting these results.

V. Conclusion

There is a great potential to gain network capacity from path diversity in Software defined networks. The gain can be realized through adopting maximum capacity routing technique such as back-pressure algorithm. This work shows that the issue of unbounded delays with back-pressure routing algorithm can be effectively mitigated by taking advantage from the centralized management feature of software defined networks and augmenting other metrics such as shortest path distance. The proposed algorithm shows significant gain in performance in terms of reduction in packet delays, average queue length, average hop length while retaining the throughput optimality property of basic back-pressure algorithm. In addition, it greatly improves the network utilization. This technique is highly useful for data centers in the transmission of delay tolerant traffic such as large data transfer or Virtual Machine (VM) transfer.

References

[1] O. N. Foundation, "Software-defined networking: The new norm for networks table," Open Networking Foundation, Tech. Rep., 2013.

[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936–1948, 1992.

[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013*, 2013, pp. 15–26.

[4] N. Feamster, J. Rexford, S. Shenker, R. Clark, R. Hutchins, D. Levin, and J. Bailey, "Sdx: A software-defined internet exchange," *Open Networking Summit*, 2013.

[5] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *INFOCOM 2009, IEEE*, 2009, pp. 2936–2940.

[6] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 3, pp. 841–854, 2011.

[7] V. Badarla, V. G. Subramanian, and D. J. Leith, "Low-delay dynamic routing using fountain codes," *IEEE Communications Letters*, vol. 13, no. 7, pp. 552–554, 2009.

[8] L. A. Maglaras and D. Katsaros, "Layered backpressure scheduling for delay reduction in ad hoc networks," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium*, 2011, pp. 1–9.

[9] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010, pp. 279–290.

[10] A. Headquarters, "Cisco data center infrastructure 2.5 design guide," 2007.

[11] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, 2008, pp. 63–74.

[12] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.