

End-to-End Routing in SDN Controllers Using Max-Flow Min-Cut Route Selection Algorithm

Nada Alzaben¹ and Daniel W. Engels²

¹Dept. of Computer Science, College of Computer Science and Information, Princess Nourah Bint Abdulrahman University Riyadh, Saudi Arabia, Southern Methodist University Dallas, USA, nialzaben@pnu.edu.sa

²AT&T Center for Virtualization, Southern Methodist University, Dallas, USA, daniel.w.engels@gmail.com

Abstract—In this paper, we present a novel max-flow min-cut based algorithm to solve the flow routing problem in the Software Defined Network Controller. Routing using traditional shortest path first algorithms often results in bottlenecks that cause performance degradation including higher energy use, reduced throughput, and increased slowdown. Our algorithm uses the max-flow min-cut algorithm to identify potential bottlenecks in order to avoid them in the flow routing decisions. Our simulations show that our max-flow min-cut based algorithm improves the network performance by minimizing the mean wait time by 15.1%, minimizing the mean slowdown by 6.1 %, minimizing the maximum completion time by 9.6 %, and maximize the mean throughput by 18.3 % compared to the Shortest Path algorithm. Explicitly considering congestion in determining routes, such as with our Max-Flow Min-Cut algorithm, is necessary to maximize performance.

Keywords—Max-Flow Min-Cut, end-to-end routing, SDN controller, wait time, throughput, makespan.

I. INTRODUCTION

Routing is a fundamental problem in network performance. Routing is the process of finding a communication path between two devices in the network to transfer information flows from one device to another. Software Defined Networking (SDN) is a network layer paradigm where the network functionality is split into two planes of operation: the control plane and data plane [1] [6] [8]. [23]. The control plane is responsible for finding routes for the flows benefiting from the network global view then deliver these routes to the data plane which is primarily responsible for the actual forwarding of packets by network switches.

Routing flows have become critically important as the use of Virtual Private Networks (VPNs), secured communications with Transport Layer Security (TLS), and the consumption of videos have come to dominate the network traffic. These types of traffic creates flows that need to be routed from source to destination with all packets in a flow following the same path. The use of Software Defined Networks (SDNs) enable global end-to-end routing of flows to optimize a given objective function such as maximize mean throughput [20] [21].

We define the end-to-end flow routing problem as follows. Given a graph $G = (V, E)$, where each edge e in E has a capacity c where $c \in \mathbb{Z}^+$ which means only c flows at a time can be transmitted on that edge, a set of flows F where each

flow f_i in F has a given number of packets and a release time Rt , non-preemptively route all flows in F on G to optimize the objective function.

We present a Max-Flow Min-Cut based algorithm that explicitly considers bottlenecks when routing a flow in the network. Shortest Path algorithms (*SP*) are most widely used in end-to-end routing algorithms [7] [24], however, they do not consider explicitly bottlenecks when routing which results in poor performance in high load networks. The shortest path based algorithms consider the shortest distance path or the minimum hops path, in either static or dynamic manner. In static shortest path, the algorithm consider the topological link state regardless of network links availability, however, the dynamic shortest path consider the logical link state in regards of network links availability at the release time of the flow.

Bottlenecks in networks occur when a high traffic flow demands to be routed over a set of edges where the number of flows are exceeding the edge capacity which leads to congestion at the edges. The Dynamic Shortest Path *DSP* finds the shortest path available considering congestion implicitly only at release time. However, this choice for a path causes the bottleneck problem by using shortest path to send the traffic which most likely will meet in a specific path segment. Clearly, the routing algorithm that will resolve or at least minimize the occurrence of bottleneck problem has to consider congestion explicitly within the algorithm.

The Max-Flow Min-Cut algorithm (*MFMC*) is a polynomial time algorithm that identifies the set of minimum number of edges in a network; such that if the edges in this set are removed, the source node is disconnected from the destination node [10] [11] [18]. Our MC suite of algorithms utilize the *MFMC* to successfully partition the network into two subnetworks, one with the source node and the other with the destination node, by identifying the minimum set of edges that hold maximum possible flow across the cut, i.e. the cut identifies explicitly the bottleneck edges. One edge from this cut is chosen for the path, and each of the subnetworks is then recursively partitioned into smaller subnetworks with edges selected until we have constructed a path between the source node and destination node. By using the *MFMC* algorithm, our MC suite of algorithms explicitly identify the bottleneck edges in the network at each step of the path generation process.

Different variations of our basic MC algorithm are evaluated

against variants of the shortest path algorithm under different flow size distributions and under high load networks. The different variations of the shortest path algorithm that we evaluate are: the shortest path based on minimizing the number of hops, we refer to them as Dynamic Shortest Path with Minimum Hop *DSP-mh* and Static Shortest Path with Minimum Hop *SSP-mh*, the shortest path based on the most congested links, we refer to them as Dynamic Shortest Path with Minimum Distance *DSP-md* and Static Shortest Path with Minimum Distance *SSP-md*, and the invert version of that which is the Shortest Path based on the least congested links, *DSP-inv* and *SSP-inv*.

We simulate the problem of routing non-preemptively with no clairvoyant, however, knowing the flow size once released, in various variations based on network topology (Barabasi Albert/ Mesh), release time (identical/arbitrary), flow size distribution (Pareto/Gaussian). We found that the mean wait time is minimized by 15.1%, the mean slowdown is minimized 6.1 %, the maximum completion time(makespan) is minimized by 9.6%, and the mean throughput is maximized by 18.3%. This indicates the need to consider other algorithms other than *SP*.

The results of our Max-Flow Min-Cut based algorithm *MFMC* is showing improvements in the network performance. Since our *MFMC* algorithm explicitly consider the network congestion when routing, we conclude that by explicitly considering network congestion when routing the network performance is maximized.

The remainder of this paper is organized as follows. We review the most relevant related work in Section II. The Max-Flow Min-Cut routing algorithm is presented in Section III. In Section IV we introduce the simulation environment. In Section V we present the simulation results and analysis of our algorithm. We draw the relevant conclusions in Section VI.

II. RELATED WORK

The Shortest Path algorithm (*SP*) was proposed by Richard Bellman and Lester Ford as the Bellman-Ford algorithm in 1958 [5] and Edsger W. Dijkstra as the Dijkstra algorithm in 1956 [17]. Some routing algorithms emerge finding the shortest path as part of their approach such as the Widest Shortest Path (*WSP*) [16], where the shortest paths are found then the path with the widest bandwidth is selected, and the Shortest-Widest Path [27] which remove links with capacity less than required flow from the graph then finds the shortest path by running Dijkstra's algorithm.

The shortest path algorithm can be obtained either from the static network link state or its dynamic link state. According to Chang H. S. *et al.* [15], the static algorithm minimizes the overhead of requesting updated link states, however, the dynamic approaches are far superior than static approaches.

Having an arbitrary network and a set of flows to be routed over it requires using routing algorithms that can achieve the goal of utilizing the network bandwidth. The existing routing algorithms in the literature are either based on using static link cost or dynamic link cost. Static link state algorithms, such as Minimum Hop Routing Algorithm (*MHR*) [9] [12] and Static Shortest Path Algorithm (*SSP*) [14], are based on hop

counts or distance, however, they do not consider bottleneck congestion in their calculations. The advantage of using these algorithms is to minimize the overhead link states updates messages between the controller and the network switches, however, they cause huge congestion by sending flows with same source and destination over the same path. Akin and Korkmaz [3] listed some of the algorithms that are based on available link bandwidth, dynamic link cost algorithms, such as Dynamic Shortest Path (*DSP*). Although the dynamic link cost algorithms may cause overhead communications with the network switches in the SDN network, with known flows these algorithms utilize network bandwidth and balance load. However, with unknown flows, on-line flows settings, their performance is not guaranteed. Our work is an extension of the line of research in the end to end routing problem with known flows in SDN where we compare our proposed algorithm with the *DSP* and *MHR* algorithms.

Routing on the World Wide Web tend to be heavy tailed with an approximately $\alpha = 1$ [4]. Park K. *et al.* [19] proved that self similarity traffic tends to be heavy tailed distribution file size and that self similarity effects the network performance. One common heavy tailed distribution is the Pareto distribution which we adopt for our flow size distribution.

To transfer large data, the network layer fragments it to fit the packet frame which does not exceed the Maximum Transfer Unit (MTU). The larger data size the more packet frames created to transfer that data. Researchers have shown that the size of the packet effect the network performance. The size of a packet can degrade the throughput once it exceeds its dedicated packet size [25]. The reason behind that is the allocated space for that packet to transfer would be double the size which will have an unused allocated space which is a waste. For example, if the dedicated packet frame size was 10 and we have a flow size of 12 then we will send two packet frames over a space of 20 which makes 8 space wasted, which doesn't utilize network resources. According to Lin *et al.* [22] to be able to send large files over the network and maintain a high quality of service, one must select an intelligent routing and scheduling algorithms for fast transfer and better network utilization.

Shortest path algorithm have been popular with both traditional networks and SDNs. Since it is widely applied in many routing algorithms, we choose to compare our algorithm performance to it.

III. MAX-FLOW MIN-CUT ROUTING ALGORITHM

The Max Flow Min Cut (*MFMC*) theory was developed in the 1950s by Ford Fulkerson and Elias-Feinstein-Shannon [18] and since then there have been a lot of algebraic topology versions made of *MFMC* [13] [2]. The *MFMC* algorithm is run in polynomial time with a complexity of $O(Ef^*)$, where f^* is the maximum flow sent on E . Assuming every node sends 10 flows, it is expected to take 122 ms for a graph of size 50 on a Microchip PIC at 5MIPS and a 49.95 ms for a graph of size 1000 on Intel Core i7 with 100,000 MIPS.

The flow routing process happens in two folds: selecting a flow from the flow set then finding a path to route the flow.

The flow selection process adopted in our simulation is either random selection when the flows have identical release times or shortest flow released first otherwise. The ordering of the flows picked is the same across all evaluated algorithms for the sake of consistency. The second fold is finding the path which we use our novel algorithm based on Max-Flow Min-Cut.

In the max-flow min-cut based algorithm a recursive loop of cuts is done where each cut contains a set of minimum number of edges with their removal the graph gets disconnected. The general algorithm is shown in Algorithm 1.

To find a path between two nodes (s, d), in an arbitrary network using the Max-Flow Min-Cut based routing algorithm, we use the pseudo code in Algorithm 1 knowing that G is an undirected graph, f_i is a flow that needs to be routed, Cut is the set of min-cut edges obtained by following the traditional min cut algorithm, $path$ is the list of selected edges representing the flow path used to route the flow, s is the source node and d is the destination node for the flow f_i .

Algorithm 1 General Min-Cut Routing Algorithm

```

1: find a path in  $G$  from source  $s$  to destination  $d$ .
2: Input: un-directed graph  $G$  and a flow  $f_i = (s, d)$ 
3: Output: a path in  $G$  for  $f_i$  that connects  $s$  to  $d$ 
4: function FIND PATH IN GRAPH ( $G, f$ )
5:   if {Connectivity( $f_s, f_d$ )} = False then
6:     return  $path = Null$ 
7:   else
8:      $Cut$  = get minimum edge cut between ( $s, d$ )
9:     partition  $G$  into  $G'$  and  $G''$  by removing edges in
 $Cut$  from  $G$ 
10:    Pick edge  $e = (e_1, e_2)$  from  $Cut$ .
11:    append  $e$  to  $path$  list
12:    if ( $f_s, f_d$ ) ==  $e$  then
13:      return  $path$ 
14:    else
15:      set new  $f_s = e_2$  and new  $f_d = e_1$ 
16:      set  $f' = (f_s, new f_d)$  and  $f'' = (new f_s, f_d)$ 
17:      if  $G' > 1$  then
18:        FIND PATH IN GRAPH( $G', f', path$ )
19:      if  $G'' > 1$  then
20:        FIND PATH IN GRAPH( $G'', f'', path$ )

```

TABLE I: Edge selection criteria in each MC algorithm variant

Algorithm	Edge Selection Criteria
MC1	the edge is selected randomly and then once path is constructed any cycles in the path will be removed
MC2	any edge in the cut edge list causing a cycle will be removed then the edge is selected randomly
MC3	the edge selection is based on being part of possible available shortest path based on minimum hops.
MC4	the edge with minimum available capacity will be selected.
MC5	the edge with maximum available capacity will be selected.

During the recursive process of cutting the edges in each loop, we select one edge to be part of the routing path. We

have five different decision criteria variations to choose the edge from the cut set. These variations are stated in Table I.

Based on the selected edge criteria the *MFMC* algorithm will produce different path. The first two decision criteria, *MC1* and *MC2* are based on randomness of selecting an edge from each cut set obtained by the recursive process. However, the difference between *MC1* and *MC2* is that *MC2* removes any cycles in the path from source to destination before selecting the edge. In *MC3* the selection criteria is based on having that edge as part of an available shortest path from source to destination at the time of the cutting process. The *MC4* and *MC5* are based on selecting the edge with the minimum and maximum capacity available, respectively. Therefore, flows with *MC4* will tend to fill up edge capacity while *MC5* tend to balance edge capacity usage.

IV. SIMULATION ENVIRONMENT

Our network is modeled as an un-directed graph $G(V, E)$, where V is a set of nodes and E is a set of edges connecting the nodes. Each edge $e_i = (v_{i-1}, v_i) \in E$ has a capacity $C(e) > 0$. A source node is indicated by s and the destination node by d and the path that connects s to d is defined by the set $P = \{e_{(s,v_1)}, e_{(v_1,v_2)}, \dots, e_{(v_i,d)}\}$. The bandwidth of a path $b(P)$ is defined as the minimum edge capacity in the path.

We simulate two different network topology to evaluate the min-cut algorithm in different environments, Mesh network and Barabasi Albert network. A Mesh network graph, $G = \{V, E\}$, is a general known network that constrains every node v_i to have at least one connection with one of the network nodes. If every node v_i is connected to all other nodes $v \in V$ we call it fully connected mesh with an edge degree of $N - 1$, otherwise it is partial mesh. Our simulation adopts a partial mesh with an edge degree of $N/2$ because it gives a distinctive difference in our objective functions compared to the full connected mesh.

The Barabasi Albert graph (AB) adopts the concept of generating networks based on power law degree distribution, where it starts with minimum 2 vertices then start to grow by adding a new node with k degree, where $k < |n|$, and it has the choice to attach to the nodes with higher degrees in the network [26].

The simulation evaluated the *MFMC* algorithm over the two network topology, Mesh network and Barabasi Albert (BA) network. In particular, we used Networkx¹ library to generate the BA network and implemented the algorithms in Python. The size of the network ranges from 10 to 50 nodes. For each generated network we randomly set each edge capacity $C(e)$ as $1 \leq C(e) \leq 10$. Each s node and d node is randomly selected in uniform distribution.

In general, given a network graph G and a set of flows F we non-preemptively route the flows F over G . The simulator represent each node as a potential source, a potential destination, or router node. The set of flows are randomly generated. Each flow is described with source node, destination node, release time, and flow size.

¹<https://networkx.github.io/>

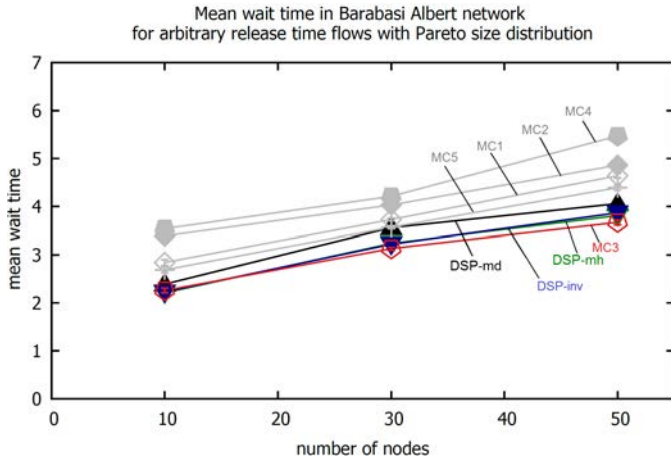


Fig. 1: Mean wait time in Barabasi Albert network with arbitrary release time flows and Pareto size distribution

We evaluate the algorithms, the *MFMC* and the shortest path algorithms, under the following scenarios:

- flows with arbitrary release time, $R_t \geq 0$, and Gaussian distribution flow size.
- flows with equal release time, $R_t = 0$, and Gaussian distribution flow size.
- flows with arbitrary release time, $R_t \geq 0$, and Pareto distribution flow size.
- flows with equal release time, $R_t = 0$, and Pareto distribution flow size.

The parameters for Gaussian distribution are set as $\mu = 5$ and $\sigma = 1$ while the parameter of Pareto distribution is set to $\alpha = 1$ in order to have a very long flows that represent large files in the network.

We denote the source node with s and the destination node with d . Time is measured as a unit time t . The simulation calculates the various objective functions as network optimization factors: the mean wait time representing average delay rate, the mean throughput, the mean slowdown, the maximum completion time, and the mean makespan. We denote the release time of a flow by R_t and the finish time by F_t with the Maximum Completion Time being $\max(F_t)$. The minimum time to travel from s to d is denoted by T_t and defined in Equation 1, where $l(f_i)$ and $l(p_i)$ represent flow size and path length respectively. We represent the wait time by WT , makespan by MS , throughput by TH , and the slowdown by SD . Equations 1–5 define these objective functions.

$$T_{t_i} = l(f_i) + l(p_i) - 1 \quad (1)$$

$$WT_i = F_{t_i} - R_{t_i} - T_{t_i} \quad (2)$$

$$MS_i = F_{t_i} - R_{t_i} \quad (3)$$

$$TH_i = l(f_i) / (F_{t_i} - R_{t_i}) \quad (4)$$

$$SD_i = (F_{t_i} - R_{t_i}) / T_{t_i} \quad (5)$$

We simulate a non preemptive scheduling problem, meaning the flow packets are sent in an uninterrupted sequence.

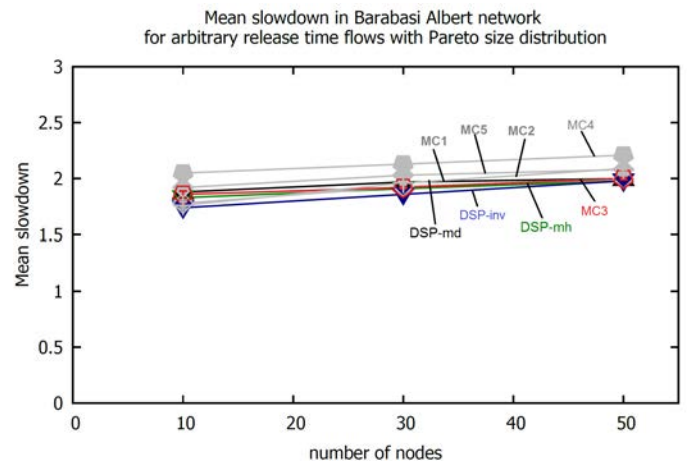


Fig. 2: Mean Slowdown in Barabasi Albert network with arbitrary release time flows and Pareto size distribution

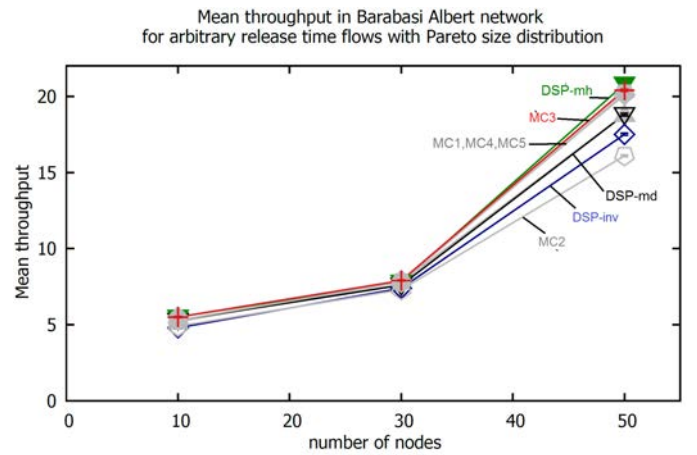


Fig. 3: Mean Throughput in Barabasi Albert network with arbitrary release time flows and Pareto size distribution

Moreover, we introduce the problem with the constraint of non clairvoyance. The non clairvoyance means we hold no knowledge over future flows in advance; therefore, the flow is known only once the flow is released.

V. RESULTS AND ANALYSIS

We evaluate our Max-Flow Min-Cut based routing algorithm, in different problem variation, with a variety shortest path algorithms: the Static/Dynamic Shortest Path algorithm with minimum distance (SSP-md/DSP-md), which chooses statically or dynamically the most congested path using Dijkstra's algorithm, the invert of (SSP-inv/DSP-inv), which chooses statically or dynamically the least congested path using Dijkstra's algorithm over inverted edge weight, and the topological Static/Dynamic Shortest Path with minimum hops (SSP-mh/DSP-mh), which chooses statically or dynamically the minimum number of hops. Evaluating our *MFMC* over

TABLE II: Comparing Max-Flow Min-Cut Algorithm to Three Path Finding Algorithms (DSP-md, DSP-mh, DSP-inv) in Maximizing Mean Throughput (positive means MC algorithm is better)

Flow Size Distribution	Release Time	Percent Difference with DSP-md					Percent Difference with DSP-mh					Percent Difference with DSP-inv				
		MC1	MC2	MC3	MC4	MC5	MC1	MC2	MC3	MC4	MC5	MC1	MC2	MC3	MC4	MC5
Network Type: Barabasi Albert																
Pareto	Identical	27.2%	18.2%	36.5%	25.9%	16.2%	28.2%	19.1%	37.5%	26.9%	17.1%	40.6%	30.7%	50.9%	39.3%	28.5%
	Arbitrary	6.7%	-9.9%	7.4%	1.1%	11.0%	-1.5%	-16.9%	-1.0%	-6.7%	2.5%	13.1%	-4.5%	13.9%	7.2%	17.7%
Gaussian	Identical	25.9%	10.8%	32.1%	23.4%	29.1%	11.4%	-2.0%	16.8%	9.1%	14.2%	18.2%	4.0%	24.0%	15.8%	21.2%
	Arbitrary	5.9%	-10.1%	9.2%	3.0%	12.1%	-3.1%	-17.7%	0.0%	-5.7%	2.7%	7.7%	-8.5%	11.1%	4.8%	14.1%
Network Type: Mesh																
Pareto	Identical	-13.6%	-18.5%	23.4%	-0.1%	3.2%	4.5%	-1.5%	49.2%	2.0%	20.9%	-10.8%	-15.9%	27.4%	3.2%	10.0%
	Arbitrary	6.1%	-9.2%	7.2%	1.5%	10.5%	2.6%	-12.2%	3.6%	-1.8%	6.9%	14.5%	-2.0%	15.6%	9.6%	19.3%
Gaussian	Identical	-5.1%	-12.0%	15.7%	6.3%	5.5%	-11.2%	-13.3%	-1.6%	-3.2%	-2.1%	3.9%	1.4%	30.1%	11.3%	12.7%
	Arbitrary	6.4%	-17.1%	15.8%	-0.7%	10.4%	0.7%	-21.5%	5.8%	-6.0%	1.2%	2.6%	-20.1%	9.8%	-4.3%	7.2%

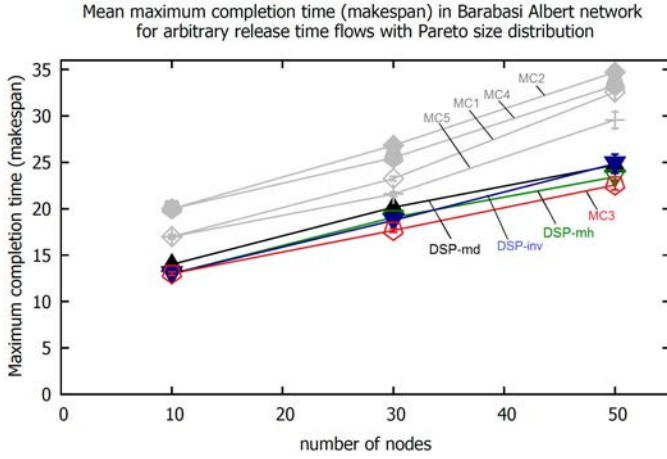


Fig. 4: Mean Maximum Completion Time in Barabasi Albert network with arbitrary release time flows and Pareto size distribution

these algorithms in these problem variations helps in evaluating the *MFMC* algorithm in more depth.

We evaluated four objective functions: minimizing mean wait time, minimizing mean slowdown, minimizing the maximum completion time (makespan) and maximizing mean throughput.

Since the dynamic routing algorithms performed far better than the static routing algorithms we will focus our result and analysis compared to the dynamic shortest path algorithms. For the objective function minimizing the mean wait time, regardless of network topology, we found that from the *MFMC* variation only *MC3* minimizes the mean wait time more than the *DSP-md*, *DSP-mh*, *DSP-inv* on average by 18.4%, 11.7% and 15.3%, respectively. Moreover, we have found that the mean wait time is minimized with flows with Pareto size distribution more than with flows of Gaussian size distribution by 20.6% and 16.1%, respectively. However, the other variation *MC1*, *MC2*, *MC4* and *MC5* have increased the mean wait time over all *DSP* on average by 16.4%.

We take as an illustration example the case where we have Barabasi Albert network and we route a set of flows in arbitrary

release times and Pareto flow size as in Fig 1. The Red line represent our *MC3* algorithm and the gray lines represent the other variation of *MFMC* and we compare them to the dynamic algorithms, *DSP-md*, *DSP-mh*, *DSP-inv*.

In the effort of minimizing the mean slowdown we found that *MC3* minimizes the mean slowdown more than the *DSP-md*, *DSP-mh*, *DSP-inv* on average by 10.2%, 1% and 7.2%, respectively. Moreover, we have found that the mean slowdown is minimized with flows with Pareto size distribution more than with flows of Gaussian size distribution by 12% and 8.4%, respectively. However, the other variation *MC1*, *MC2*, *MC4* and *MC5* have increased the mean slowdown over all *DSP* on average by 5% as illustrated in Fig 2.

To maximize the mean throughput, regardless of network topology, we also found that *MC3* maximized the mean throughput more than all *DSP* on average by 18.3%. Moreover, we have found that the mean throughput is maximized with flows with Pareto size distribution same as with flows of Gaussian size distribution by 18.6% and 17.2%, respectively. However, *MC2* have not increased the mean throughput over the *DSP*, it has minimized it by 5.4%, as illustrated in Table II. As for *MC1*, *MC4*, *MC5*, they maximized the mean throughput more than all *DSP* on average by 9.0%.

We take the same illustration example with Barabasi Albert network, routing a set of flows released in arbitrary times with Pareto flow size, as in Fig 3. The Red line represent the *MFMC* variant that most maximizes the mean throughput, our *MC3* algorithm, and the gray lines represent the other variation of *MFMC* and we compare them to the dynamic shortest path algorithms as well.

For the objective function of minimizing the maximum completion time(makespan), regardless of network topology, we have found that only *MC3* minimizes the makespan more than all *DSP* on average by 9.6%. Moreover, we have found that the mean wait time is minimized with flows with Pareto size distribution more than with flows of Gaussian size distribution by 18.2% and 7.5%, respectively. However, the other variation *MC1*, *MC2*, *MC4* and *MC5* have increased the makespan over the *DSP* on average by 5.7%.

An illustration of the result of minimizing the maximum completion time is presented in Fig 4. The Red line represent the *MFMC* variant that most minimizes the maximum com-

pletion time, the makespan, which is our *MC3* algorithm, and the gray lines represent the other variation of *MFMC* and we compare them to the dynamic shortest path algorithms as well.

In all of our simulations we found that the constrain on flow release time, either arbitrary or identical, didn't have any impact on the evaluation on our objective functions. Furthermore, the min-cut algorithm that chooses its path based on minimum hop counting performed better in minimizing the mean wait time, minimizing the maximum completion time and maximizing the mean throughput, however, it has insignificant change in the mean slowdown.

VI. CONCLUSIONS

To optimize the end to end routing in the Software Define Network controllers, a fast to compute and near optimal routing algorithm should be used to maximize performance of the network. Our simulation showed that the novel max-flow min-cut based algorithm with a decision based on minimum hops provides superior performance compared to the traditional shortest path algorithm variants. Routing using traditional shortest path first algorithms often results in bottlenecks that cause performance degradation including reduced throughput and increased slowdown. Our Max-Flow Min-Cut based algorithms explicitly identify the bottlenecks through the min-cut edges which holds the least available flows, and, by choosing the edge on the shortest dynamic path, our algorithm is able to find paths that yield significant improvement over the commonly used shortest path algorithms. The diversity of our MFMC suite of algorithms demonstrates that a shortest path approach, such as *MC3*, as opposed to a balanced or longer path approach, yields the best expected performance across a broad range network types and flow size distributions.

REFERENCES

- [1] Sugam Agarwal, Murali Kodialam, and TV Lakshman. Traffic engineering in software defined networks. In *2013 Proceedings IEEE INFOCOM*, pages 2211–2219. IEEE, 2013.
- [2] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: Theory, applications and algorithms*. Prentice-Hall, Englewood Cliffs, New Jersey, USA Arrow, KJ (1963). *Social Choice and Individual Values*, Wiley, New York. Gibbard, A.(1973). "Manipulation of Voting Schemes: A general result", *Econometrica*, 41:587–602, 1993.
- [3] E. Akin and T. Korkmaz. Comparison of Routing Algorithms With Static and Dynamic Link Cost in Software Defined Networking (SDN). *IEEE Access*, 7:148629–148644, 2019.
- [4] Martin F. Arlitt and Carey L. Williamson. Web server workload characterization: The search for invariants. In *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '96, page 126–137, New York, NY, USA, 1996. Association for Computing Machinery.
- [5] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.
- [6] C. J. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zúñiga. An architecture for software defined wireless networking. *IEEE Wireless Communications*, 21(3):52–61, June 2014.
- [7] Q. Chen, K. Zheng, F. Ouyang, X. Gan, Y. Xu, and X. Tian. A shortest path routing algorithm based on virtual coordinate in nels. In *2016 8th International Conference on Wireless Communications Signal Processing (WCSP)*, pages 1–5, Oct 2016.
- [8] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.*, 44(2):87–98, Apr 2014.
- [9] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.
- [10] L. R. Ford and D. R. Fulkerson. Network flow and systems of representatives. *Canadian Journal of Mathematics*, 10:78–84, 1958.
- [11] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
- [12] J.J Garcia-Luna-Aceves. A minimum-hop routing algorithm based on distributed information. *Computer Networks and ISDN Systems*, 16(5):367 – 382, 1989.
- [13] R. Ghrist and S. Krishnan. A topological max-flow-min-cut theorem. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 815–818, Dec 2013.
- [14] R. A. Guerin, A. Orda, and D. Williams. Qos routing mechanisms and ospf extensions. In *GLOBECOM 97. IEEE Global Telecommunications Conference. Conference Record*, volume 3, pages 1903–1908 vol.3, Nov 1997.
- [15] Hong Seong Chang, Byoung Wan Kim, Chang Gun Lee, Sang Lyul Min, Yanghee Choi, Hyun Suk Yang, Doug Nyun Kim, and Chong Sang Kim. Performance comparison of static routing and dynamic routing in low-earth orbit satellite networks. In *Proceedings of Vehicular Technology Conference - VTC*, volume 2, pages 1240–1243 vol.2, 1996.
- [16] Jang-Ping Sheu, Quan-Xiang Zeng, R. Jagadeesha, and Yeh-Cheng Chang. Efficient unicast routing algorithms in software-defined networking. In *2016 European Conference on Networks and Communications (EuCNC)*, pages 377–381, June 2016.
- [17] Donald B Johnson. A note on Dijkstra's shortest path algorithm. *Journal of the ACM (JACM)*, 20(3):385–388, 1973.
- [18] Lester R Ford Jr. Network flow theory. Technical report, Rand Corp Santa Monica Ca, 1956.
- [19] Kihong Park, Gitae Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proceedings of 1996 International Conference on Network Protocols (ICNP-96)*, pages 171–180, 1996.
- [20] H. Kim and N. Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, February 2013.
- [21] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.
- [22] Chuan Lin, Yuanguo Bi, Hai Zhao, Zeshen Wang, and Jinfa Wang. Scheduling algorithms for time-constrained big-file transfers in the internet of vehicles. *Journal of Communications and Information Networks*, 2(2):126–135, Jun 2017.
- [23] N. McKeown. How SDN will shape networking, Oct 2011.
- [24] A. Nanda and A. K. Rath. Cost effective modleach-a x2217; search algorithm for shortest path routing in wireless sensor networks. In *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, pages 147–151, Dec 2016.
- [25] A. U. Shah, D. H Bhatt, P. R. Agarwal, and P. R. Agarwal. Effect of packet-size over network performance. *International Journal of Electronics and Computer Science Engineering*, 1:762–766, 2012.
- [26] S. Tie-li, D. Jing-wei, and D. Kai-ying. Scale-free network model with evolving local-world. In *2008 Fourth International Conference on Natural Computation*, volume 1, pages 237–240, Oct 2008.
- [27] Man-Ching Yuen, Weijia Jia, and Chi-Chung Cheung. Simple mathematical modeling of efficient path selection for qos routing in load balancing. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 1, pages 217–220. IEEE, 2004.

Nada Alzaben. This author became a Member (M'18)

of IEEE in 2018. Born in Riyadh, Saudi Arabia. received her B.S. (2001) and M.S. (2012) in computer science and information from King Saud Univ. in Riyadh, Saudi Arabia. She is currently working on her Ph.D. in computer science at Southern Methodist Univ., Dallas, TX, USA. Her research interests include network communications, routing and scheduling, network performance analysis. She is a Lecturer at Princess Nourah Bint Abdulrahman Univ., Riyadh, Saudi Arabia. Previously she worked as a senior programmer at King Saud Univ. She has 3 research publications in the field of computer networks and AI.

Dr. Daniel W. Engels is currently Head of AI, Cybersecurity Science and Analytics, at HSBC. Previously, Dr. Engels was a professor at Southern Methodist University (SMU) where he was one of the creators of, and the first Director of, the Master of Science in Data Science (MSDS) program where he grew the program from 0 students to more than 300 students in its first two years of operation. In addition to his data science expertise, Dr. Engels is an expert in security and in RFID and IoT technologies, systems, and applications. He was one of the founders of the IEEE International Conference on RFID, and he was the Chair of the IEEE Technical Committee on RFID in 2011 and in 2012. Dr. Engels is the former Director of Research of the Auto-ID Labs at MIT, where he led the development of several RFID protocols including the original "Gen2" protocol, and he is an original member of the research team that founded the Auto-ID Center at MIT and created the EPC System. Dr. Engels received his Ph.D. from the Massachusetts Institute of Technology. He has over 100 peer reviewed publications and 8 issued U.S. patents in data science, RFID, RFID applications, Internet of Things, security, embedded computing, and computer-aided design. Dr. Engels received the 2014 AIM Ted Williams Award in recognition of his contributions to the AIDC industry. Dr. Engels is a member of AIDC 100 and is a Senior Member of IEEE.