I. J. Computer Network and Information Security, **2023, 5, 50-62**
Published Online on October 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2023.05.05

Flow-aware Segment Routing in SDN-enabled Data Center Networks

Bommareddy Lokesh
National Institute of Technology Puducherry, Department of Computer Science and Engineering, Karaikal, 609609, India
E-mail: bommareddylokesh@gmail.com
ORCID iD: https://orcid.org/0000-0002-8753-6160

Narendran Rajagopalan*
National Institute of Technology Puducherry, Department of Computer Science and Engineering, Karaikal, 609609, India
E-mail: narendran@nitpy.ac.in
ORCID iD: https://orcid.org/0000-0002-1829-9587
*Corresponding Author

Received: 25 February 2022; Revised: 25 May 2022; Accepted: 17 October 2022; Published: 08 October 2023

Abstract: The underlying objective of segment routing is to avoid maintenance of the per-flow state at forwarding devices. Segment routing (SR) enables the network devices to minimize their forwarding table size by generalizing the forwarding rules and making them applicable to multiple flows. In existing works, optimizing the trade-off between segment length and the number of co-flows sharing the segment is considered the key to determining optimal segment endpoints. However, the flow characteristics like the lifetime of flows, and dynamically altering routing paths are critical and impact the performance of SR. Ideally, network flows considered for SR are expected to persist for a longer duration and adhere to static routing paths. But our analysis of flow characteristics at a typical data center reveals that the majority of flows are short-lived. Also, network flows are subject to alter their routing paths frequently for several reasons. Considering short-lived flows and flows that dynamically alter their routing paths may lead to choosing unstable segment endpoints. Hence, it is necessary to study the flow characteristics for determining more stable segment endpoints. In this paper, the authors implemented the SR technique considering the flow characteristics at an SDN-enabled data center and the results show a significant improvement with respect to the stability of segment endpoints.

Index Terms: Segment Routing, Segment Identifier, Segment List Depth, Forwarding Table.

1. Introduction

In large enterprises and data center networks, the increasing trend of Forwarding Information Base (FIB) is raising scalability concerns and needs immediate attention [1]. Existing network devices with limited TCAM (Ternary Content Addressable Memory) may fail to store such large forwarding tables and replacing them with higher capacities will incur huge costs. Hence forwarding table size (FTS) reduction techniques are gaining importance in recent times. Segment routing (SR) is a source routing technique that helps to minimize the FTS and yet satisfy strong forwarding correctness.

SR allows sharing of forwarding rules among multiple flows, provided they are associated with an identical sequence of network devices along their routing paths, commonly known as shared segments. In SR, an ordered list of segment IDs (SIDs) indicating the forwarding path is included in the packet header instead of the conventional source and destination addresses. Accordingly, the match criteria of forwarding rules at network devices are redefined to more flexible SIDs in place of the traditional source and destination address pair. Network devices will refer to the SIDs in their forwarding tables to find a matching entry and make routing decisions accordingly. Usually, while routing packets from source to destination, multiple flows might share a few network links along their routing paths. Two flows are termed as co-flows at a particular device if they share a forwarding table entry at that device. Several parameters like segment length, co-flow count, etc. play a key role in determining optimal segment endpoints. The primary objective of identifying co-flows is to bind them and forward packets along their shared segments using the same set of forwarding rules. Flow bundling allows for avoiding maintenance of exclusive entry in the forwarding table for each flow along with their shared segments. Segments can be uniquely identified using SIDs which are included in the packet header. Instead of the conventional source and destination addresses pair, the SID in the packet header is mapped with entries in the forwarding tables to find a matching entry.

For a given flow, several flows may converge in and diverge away at various phases along its routing path. So eventually different combinations of co-flows might result in different segment endpoints. In existing works, segment endpoints are determined based on the number of co-flows sharing the segment, the length of the segment, etc. The primary focus is on minimizing the FTSmax by determining co-flows in such a way that the majority of flows can be bundled at a maximum number of intermediate devices. The choice of co-flows clearly effects the segment endpoint distribution. Determining segment endpoints passively implies the combination of flows that can be bundled and forwarded. Binding co-flows and determining segment endpoints without considering the flow characteristics will result in determining transiently optimal segment endpoints instead of stable ones.

The forwarding devices store the forwarding table in their fast memory known as TCAM (Ternary Content Addressable Memory). Considering short-lived flows for SR might result in choosing unstable segment endpoints. Since the flows are short-lived, a majority of them would shortly terminate and the upcoming flows with unique source and destination addresses may seek an exclusive entry in the devices forwarding table resulting in a sharp rise in the forwarding table size. As a result, the co-flows are to be reexamined afresh considering the new flows for further reduction in FTS. Studies show that most data center flows are highly dynamic in nature and short-lived. Hence, it is important to study the flow characteristics like the lifetime of flows, and the granularity at which the data packets are being transferred, prior to considering them for SR.

Due to the unified view of the network, the SR technique can be effectively implemented in SDNs. Effective utilization of available bandwidth and detection of conflicting flow rules at the data plane in SDNs is presented in [2]. An SR controller helps to eliminate the need for storing network state information at intermediate devices and instead provides them with path information. The devices route the packets based on the path information often encoded as a stack of SIDs. This technique would help devices reduce their FTS significantly. SDNs not only provide data plane scalability but also enable many key features required for modern networks [3].

In the existing research works on FTS minimization, the choice of segment endpoints is primarily made based on either the segment length or the number of co-flows sharing the flow entries over the segment. However, to the best of our knowledge, none of the existing works focused on flow characteristics which is the key to determining stable segment endpoints.

1.1. Research Objective

The objective of our work is to determine stable segment endpoints and consistently maintain a smaller FTSmax at the network devices.

1.2. Unique Contributions of this Work

The novelty of our work lies in the choice of flows picked for segment routing. Flow characteristics play a crucial role in determining stable segment endpoints. To the best of our knowledge, none of the existing works addressed the importance of flow characteristics in determining stable segment endpoints. Our experiment proved that choosing long-lived flows helps in determining more stable segment endpoints.

2. Related Works

Existing works on SR in SDNs can be broadly classified into three categories.

2.1. SLD-centric

The number of SIDs that can be stored in the packet's header is limited to the amount of space available in the packet's header. Hence the number of segments that an end-to-end path can be partitioned is limited to the Segment List Depth (SLDmax). Several path encoding methods and techniques have been proposed to deal with such problems and compute the optimal routing path.

A technique for determining the smallest number of SIDs that enables packet forwarding over the selected path is proposed and implemented by Davoli et al. [4]. The proposed traffic engineering and segment routing modules enhance the controller's functionality and help in solving the classical flow assignment problem.

The ability to incorporate a limited number of SIDs in the IP header makes SR impractical in networks with a large number of flows. However, Huang et al. proposed a technique to tackle this issue to some extent through optimization techniques [5]. The authors proposed a novel architecture for optimal utilization of space in the packet's header. They have presented a technique based on the flexibility offered by OpenFlow through network programmability.

An efficient technique that collectively ensures optimal paths for flows and minimum SLD in segment routed networks is proposed and implemented by Lazzeri et al. [6]. Yet another technique to minimize the SLD using the flexibility offered by SDN controller and path computation modules was proposed and implemented by Dugeon et al. [7]. Giorgetti et al. proposed two algorithms for path encoding that ensures minimum SLD [8]. Guedrez et al. proposed SR-LEA to compute the minimum label stack using IGP shortest path technique [9].

Cianfrani et al. proposed a solution for the segment list encoding problem while minimizing the SLDmax [10]. Their procedure involves two steps, generating an auxiliary graph that resembles the routing paths between the source and destination devices and applying a solution of the multi-commodity flow problem on that auxiliary graph.

2.2. Traffic Engineering (TE)

Bhatia et al. proposed and implemented a technique for distributing the traffic across the network links through traffic engineering [11]. The primary objective of their work is to choose the segment endpoints such that the data traffic is well distributed across the network. The authors considered three scenarios for handling congestion in SR, namely *Traffic matrix aware segment routing, Traffic matrix oblivious segment routing,* and *Online segment routing*. such problems and compute the optimal routing path.

An efficient methodology for analyzing the performance of traffic engineering in segment routed networks through ILP and heuristics is proposed by Moreno et al. [12]. The inherent drawbacks of ignoring the ECMPs in SR are explored. Traffic engineering solutions with a minimum stack of labels in the IP header are proposed and implemented.

2.3. Optimizing the Trade-off between SLDmax and FTSmax

In extremely large networks, domain partitioning helps to restrict the scope of network devices within a partition, thereby helping in reducing the FTS. Anbiah and Sivalingam used the domain partitioning technique for minimizing the entries in the forwarding table [13]. Partitioning of the network into sub-domains is done using the sparse cut method [14]. Devices within a partition are unaware of SIDs outside their partition. However, the proposed technique cannot be applied on networks that do not support domain partitioning.

Another heuristic-based technique for minimizing the FTS in SDNs is proposed and implemented by Anbiah and Sivalingam [15]. This technique is suitable for networks that cannot be partitioned into subdomains. The underlying objective of their technique is to minimize the FTSmax by sharing forwarding table entries among co-flows. Their technique is limited to optimizing the trade-off between the segment length and the number of co-flows sharing the segment. However, the authors ignored flow characteristics like the lifetime of flows for determining segment endpoints, which led to the identification of transient segment endpoints instead of stable ones.

A comprehensive survey report on SR is presented by Ventre et al. [16]. The report presents the recent advancements in SR technology. Another survey report on the applications of segment routing technology in modern networking paradigms is presented by Abdullah et al. [17].

In this paper, we presented the CCHstable technique which is a variant of the heuristic-based CCH technique proposed and implemented by Anbiah and Sivalingam [15]. The CCH technique is suitable when several ECMPs exist in the network. Generally, it is difficult to determine co-flows when ECMPs exist between the source and destination nodes. The authors in [2] introduced a simple and straightforward approach to determine the co-flows. The underlying technique to deal with ECMPs in the CCH approach is to identify flows that converge at any of their downstream nodes along their routing paths so that all the co-flows can share a common forwarding table entry at their shared upstream devices.

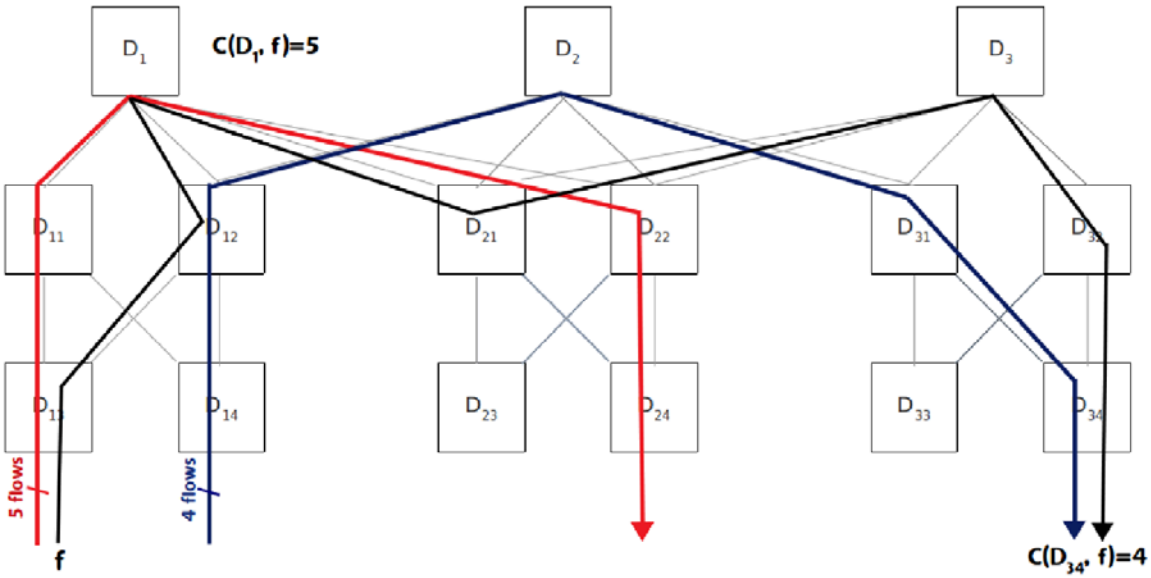

Fig.1. Demonstration of Coflow Count Heuristic (CCH) in a sample network

Consider a sample network with the flows specified as shown in Fig. 1. Let the red line and blue line represent a bundle of 5 flows and 4 flows respectively, and the black line indicates the routing path for flow ‘f’. The prerequisite for a set of flows to be considered as co-flows for a given flow ‘f’ at some device D_x is that, ‘f’ should have converged with the flows at least at one of their upstream devices. Flow ‘f’ converges with the red and blue bundles of flows at multiple devices. Hence, CCH can be computed for flow ‘f’ at the downstream device D_{34} with respect to flows in the blue bundle and at device D_1 with respect to flows in the red bundle. The CCH for flow ‘f’ at device D_{34} is 4 considering the blue bundle of flows because they share a common upstream device D_{12}. Similarly, the CCH for flow ‘f’ at device D_1 is

computed to be 5 because flow 'f' shares a common upstream device D_{13} with the flows represented by the red line. The device with the highest heuristic value is considered to be a better segment endpoint for a given flow. Hence, compared to D_{34}, D_1 is considered to be a better segment endpoint for flow 'f', since 'f' can share forwarding table entries with 5 other flows at device D_{13} which results in a better reduction in FTS at device D_{13}.

For a fair comparison of the result and to realize the exact outcome of the proposed technique with respect to FTSmax, we implemented the same CCH technique proposed by the authors in [15] (but exclusively on medium-sized long-lived flows) for determining co-flows. The results show that the proposed technique helped to identify more stable and optimal segment endpoints by selectively picking flows ideal for SR with respect to their characteristics.

3. Flow Characteristics at a Typical Data Center

In general, the lifetime of flows depends on the nature of workloads in the network [18,19,20]. Due to the workload-driven lifetime and flow sizes, metrics for flow classification cannot be fixed. Hence, we admit that flow classification with respect to flow duration or size should be purely relative. Kindly note that the metrics used for flow classification in this paper are not claimed as standard metrics.

In this paper network flows are classified into two categories based on the duration of their lifetime (assuming a 10-second duration as a threshold):

3.1. Short-lived Flows

Network flows that terminate within 10 seconds from the point at which they are initiated are categorized as short-lived flows.

3.2. Long-lived Flows

Flows that last for 10 seconds and longer are considered long-lived flows.

In addition, flows are classified into three categories based on the granularity at which they transfer data packets:

3.3. Tiny Flows

Tiny flows are those that carry a very small number of data packets, usually less than 10.

3.4. Medium-sized Flows

Medium-sized flows comprise at least 10 packets but not more than 100 packets each.

3.5. Elephant Flows

Elephant flows operate at a very high granularity in terms of packet count and are likely to account for at least 100 packets each.

Table 1. Notation summary table (flow size)

Notation	Packet count
tiny flows	<10
medium-sized flows	10 - 100
elephant flows	>100

Table 2. Notation summary table (elapsed time)

Notation	Duration (in sec)
short-lived	<10
long-lived	≥10

Table 3. Illustration of flow characteristics with respect to segment routing

	short-lived (≈75%)	**long-lived (≈25%)**
tiny flows (≤5%)	not suitable	good
medium-sized flows (≈85%)	less suitable	best
elephant flows (≈10%)	not suitable	better

We classified flows based on their lifetime and the granularity at which the data packets are being exchanged. The relevance of different flows to the proposed technique and an overview of the percentage of different flows at a typical data center is presented in Table 3. The value enclosed within the brackets next to the row headers indicates the percentage of flows at a typical data center that can be classified based on the flow size. In terms of packet count, flows can be broadly classified into three categories: i. tiny flows ii. medium-sized flows and iii. elephant-sized flows. The value next

to the column headers indicates the percentage of flows that are classified based on their lifetime at a typical data center. Data center traffic can be broadly classified into two categories based on their lifetime: i. short-lived and ii. long-lived flows.

Most of the elephant-sized flows are long-lived by nature and hence considering them for determining segment endpoints would result in more stable segment endpoints. Elephant-sized flows carry large volumes of data and hence not just the packet size, the volume of packets associated with elephant flows is also high. In SR, the upcoming segment SID needs to be updated in the packet's header at the beginning of each segment (pop the old SID from the list at the end of each segment so that the SID of the upcoming segment can be used by the subsequent devices in making their forwarding decisions). Although, the SR controller can provision the forwarding devices with the list of SIDs along the packet's routing path, considering elephant-sized flows for SR involves additional overhead for the forwarding devices. Also, elephant-sized flows are often delay-sensitive, hence they may not be ideal for SR. On the other hand, long-lived medium-sized flows transfer a relatively lesser number of packets and hence they can be forwarded with less overhead. Moreover, the benefit of SR is observed to be better when the flows follow static routing paths. Since Elephant flows often saturate the available bandwidth and are likely to alter their routing paths frequently due to congestion, elephant-sized flows are not well-suited for SR.

Long-lived tiny flows are less suitable than their counterpart in medium-sized flows because tiny flows remain inactive most of the time and occasionally transmit very few packets. Hence tiny flows usually don't need an exclusive entry in the device's forwarding table. A relatively smaller overhead involved in modifying the header fields of the packets at each segment endpoint corresponding to medium-sized flows makes them more suitable for the proposed technique compared to elephant flows.

One key observation to be noted from the result shown in Fig. 2 is that the FTSmax recorded while considering all medium-sized flows for SR resulted in a smaller FTSmax initially because the volume of flows considered for SR is relatively high compared to the volume of flows considered in the CCHstable technique. But the CCHstable outperformed the former technique and resulted in a smaller FTSmax over time because of the stable segment endpoints. A similar trend in FTSmax can be observed in the CCH technique (where the entire population of flows is considered for SR) and the resultant FTSmax where all long-lived flows are considered for SR. The resultant FTSmax in the CCHstable is smaller compared to the FTSmax considering all long-lived flows over time because of the dynamically changing routing paths of the elephant-sized flows. The conclusion that can be drawn from the above discussion is that the initial FTSmax is minimum when a larger sample of flows is considered for SR, and more stable segment endpoints can be determined when long-lived flows are considered for SR.

4. Proposed Model

The proposed CCHstable technique is based on the fact that, determining segment endpoints considering long-lived flows leads toward stable segment endpoints. In the CCHstable technique, the network flows are classified into various categories based on their size and lifetime. The relevance of flow characteristics to SR is presented in Table 3. Flows that aid in selecting stable segment endpoints should be identified and selected for SR.

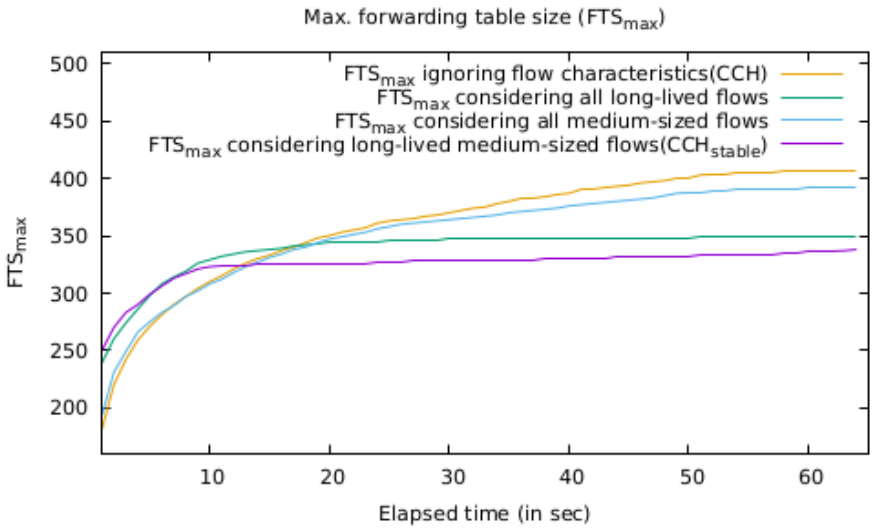

Fig.2. Variation in FTSmax considering different types of flows for SR

Fig. 2 presents the variation in FTSmax considering different types of flows for SR. The resultant FTSmax shows the importance of considering long-lived flows for determining stable segment endpoints in SR. One key observation to be noted is the FTSmax considering all long-lived flows for SR is less than the FTSmax recorded in CCHstable over time. The lower FTS recorded when all long-lived flows are considered for SR is because long-lived elephant-sized flows are

not considered for SR in the latter technique. However, the FTSmax considering all medium-sized flows is relatively less initially because so many short-lived medium-sized flows are also considered for SR. The short-lived flows led to transiently optimal segment endpoints instead of stable ones. The initial FTSmax is apparently smaller when a large number of flows are considered for SR compared to the FTSmax when flows are selectively picked for SR.

While a new flow is being generated, the source and destination pair are randomly selected, and hence flows with respect to flow characteristics are not always well diversified across the network. For instance, let us assume a device through which too many tiny or elephant-sized flows are being redirected. Due to the inability of the device to participate in SR, the FTSmax gets badly affected since we are looking at minimizing the FTSmax in the network as a whole.

The inference from the above discussion is that considering a large number of flows for SR may result in smaller FTSmax initially, but eventually leads to higher FTSmax compared to segments identified using stable and long-lived flows. Hence long-lived medium-sized flows are considered to be an ideal choice for the proposed CCHstable technique.

Since a relatively smaller number of flows are selected for identifying segments in the CCHstable technique, the resultant FTSmax may not be as good as the CCH technique. However, the rapid increase in the number of forwarding table entries in CCH proves that CCHstable outperforms the CCH technique over time.

4.1. The Key Concerns in the CCHstable Technique

- Due to the randomness in the topology generated with the Inet topology generator, few devices may not be able to participate in SR because of the non-uniformity in diverse flow characteristics among flows distributed in the entire network. FTSmax is seriously affected at a very few devices because the nature of flows that are being redirected through them are not suitable for our technique. Since FTSmax is calculated considering all the devices in the entire network, because of the unusual mix of flows at a few devices, the overall FTSmax could not be minimized to a level close to the original version of CCH.
- Having established a higher FTSmax initially, the proposed technique may take too long to reach the break-even point.

4.2. Quartile Level FTSmax

We analyzed the range-bound frequency distribution of FTS at all the devices to study the impact of CCHstable on FTSmax. The Quartile level FTSmax (QFTSmax) at different quartiles are presented in Table 5. The QFTSmax for range-bound frequency distribution of FTS is calculated using (1).

$$Q_k = R_{lb}(Q_k) + \left(\frac{\frac{kN}{4} - \left(cf_{Q_k} - f_{Q_k}\right)}{f_{Q_k}}\right) i \tag{1}$$

Table 4. Notation table

Notation	Description
k	Quartile number
i	Quartile class
$R_{lb}(Q_k)$	lower bound of the quartile class range
N	total number of devices
f_{Q_k}	the number of devices with FTS within the Quartile class range.
cf_{Q_k}	cumulative frequency of devices up to the Quartile class Q_k

The inference from the analysis of QFTSmax through different Quartiles is that the FTSmax was badly affected due to the unusual mix of flows forwarded through very few devices. These devices were not able to participate in SR because many flows forwarded through them were not selected for SR due to their unseemly nature of flow characteristics to the proposed technique.

Table 5. QFTSmax when FTSmax=254

Quartile	$QFTS_{max}$ value
Q1	183
Q2	188
Q3	197

The data presented in Table 5 shows the impact of unusual flow characteristics at very few devices on the resultant FTSmax. One key observation to be noted is that, even though the overall FTSmax is 254, the FTS at 90th percentile is only 204.

We studied the possibilities in enhancing the performance of the proposed technique by minimizing the time required to reach the break-even point. The conclusion that can be drawn from the analysis is that, the FTSmax can be further minimized significantly if more flows are selected for SR at devices with higher FTS.

4.3. 0/1 Knapsack Technique

The 0/1 Knapsack technique allows to sensibly pick more flows for determining co-flows at devices with higher FTS. For minimizing the FTSmax, we applied the 0/1 Knapsack technique for selecting flows at devices. The Knapsack technique allows considering more flows at devices with higher FTS to participate in SR more aggressively. Procedurally, we assumed certain weights and values for flows to apply the analogy of 0/1 Knapsack problem.

The weights and values assumed for different flows are presented in Table 6. The value next to the row headers indicates the weights assumed for flows. Tiny flows do not suit our proposed technique; hence the weight of elephant flows is considered to be high. Although elephant-sized flows are usually long-lived, due to the additional overhead involved in considering them for SR, their weight is assumed to be a bit higher compared to the medium-sized flows but much smaller compared to tiny flows.

Table 6. The value to weight ratios assumed for different flows

(values) / (weights)	**short-lived (1)**	**long-lived (10)**
tiny flows (5)	0.2	2
medium-sized flows (1)	1	10
elephant flows (2)	0.5	5

Since SR involves modification of mere fields in the packet header, considering elephant-sized flows for SR involves some overhead for the devices. Tiny flows are usually short-lived and hence they do not suit our proposed technique. Moreover, tiny flows do not require a dedicated entry in the devices forwarding table because they constitute only a few packets for which the routing path can be dynamically computed. Especially in SDNs, due to the centralized view of the entire network, packets associated with tiny flows can be exclusively forwarded by the controller whenever there is some data packet to be forwarded. Additionally, in tiny flows, the intra-flow packet arrival time is too high. So, retaining forwarding table entries corresponding to tiny flows for a longer duration leads to the wastage of forwarding table resources.

Similarly, the value assumed for different flows is enclosed within the brackets next to the column headers. Since short-lived flows do not suit our model, the value for short-lived flows is assumed to be 1. On the other hand, the value for long-lived flows is assumed to be 10. The value to weight ratios allows the selection of flows for SR tailor-made for individual devices. This approach allows all the devices to uniformly participate in SR resulting in a smaller FTSmax. However, the overhead involved in applying the 0/1 Knapsack analogy at few devices do exist in the Knapsack version of the CCHstable technique.

5. Experimental Setup

We simulated the SDN functionality for implementing the CCH and the CCHstable techniques. The autonomous system level Internet topology generator, Inet-3.0 [21] is used to set up a virtual network. The virtual network comprises 80 forwarding devices and 5k flows are triggered at distinct time intervals between pairs of forwarding devices. Information regarding path lengths of flows in terms of hop count is presented in Table 7. The average path length of all the flows in the said virtual network is equal to 3.41. Ideally, SLDmax is chosen to be analogous to the average path length of all the flows in the network. Hence, in our experiment, SLDmax is assumed to be the nearest integer to the average path length (i.e., 3). Increasing SLDmax beyond the average path length has no significant impact on both the co-flow identification and the segment endpoint distribution. The degree of nodes indicating the interconnecting links between the network devices is presented in Table 8. Table 9 showcases the information about the number of flows, number of devices, and the SLDmax. We analyzed the flow characteristics at Google and Facebook data centers to understand the nature of data center traffic [18,21]. Considering the flow characteristics at a typical data center, nearly 5% of flows in the entire network are characterized as tiny flows, about 85% of flows are specified to be medium-sized, and the remaining 10% of flows are characterized as elephant-sized flows for the experimental analysis. For implementing the proposed technique, we assumed certain weights and values for flows based on their size and lifetime respectively as shown in Table 6. The idea is to apply the 0/1 Knapsack technique for selecting co-flows at devices with higher FTS for further reduction in FTSmax. The values are assumed based on the relevance of flow characteristics to the proposed technique and have no other significance.

The topology of the test network used for validation of the proposed technique, and also the source and destination devices for flows are randomly chosen. Since they are randomly selected, the distribution of network traffic across the devices would be helpful to better realize the network traffic. Table 10 presents the confidence intervals and variance around the presented mean values for FTS in all three techniques. The huge variance in FTS among devices is understandable because the volume of data traffic is different from the flow distribution. Please note the difference between the traffic distribution and flow distribution (many tiny or medium-sized flows may account for a lesser volume of data traffic compared to the data transferred by a single elephant-sized flow). FTS_before_minimization represents the

descriptive statistics of the number of flows being redirected through devices before applying any of the FTS minimization techniques.

Table 7. Inet topology path lengths

path length	% of flows
1	< 1%
2	≈ 8%
3	≈ 52%
4	≈ 36%
5	≈ 4%
6	< 1%

Table 8. Inet topology node degree

node degree	% of nodes
1	≈ 10%
2	≈ 34%
3	≈ 25%
4	≈ 15%
5	≈ 8%
6	≈ 4%
≥ 7	≈ 3%

Table 9. Inet topology other parameters

parameter	value
flows	5000
devices	80
SLD	3

Table 10. Descriptive statistics of forwarding table size (FTS) of all 80 devices (to realize the flow distribution in the network)

	FTS_before_ minimization	FTS_CCH	FTS_CCH_stable	FTS_CCH_stable +Knapsack
Mean	258.95	150.9375	186.7875	178.0625
Standard Error	7.468772542469	2.68347874013	2.632122269	2.9906152657
Mode	244	150	187	189
Median	252.5	153.5	187	188
First Quartile	215	138.75	184	184
Third Quartile	300	172	197.25	191.25
Variance	4462.60506329	576.084651898	554.2454113	715.50237341
Standard Deviation	66.802732453	24.001763516	23.54241728	26.74887611
Kurtosis	0.39156845259	0.83795644082	4.5341742908	3.2937480109
Skewness	0.31895370782	-1.00684431	-1.3071225	-2.068612228
Range	348	110	157	117
Minimum	104	71	97	86
Maximum	452	181	254	203
Sum	20716	12075	14943	14245
Count	80	80	80	80
Confidence Interval(95%)	[217.546, 300.354]	[136.061328, 165.813672]	[172.196031, 201.378969]	[161.483683, 194.641317]
Confidence Interval(90%)	[224.2027, 293.6973]	[138.453023, 163.421977]	[174.541953, 199.033047]	[164.149118, 191.975882]

6. Results and Discussion

We implemented both the CCH and the CCHstable techniques to realize the variation in FTSmax. The proposed CCHstable technique is relatively fast and stable because we are dealing with only a sub-population of flows that are selectively picked for SR. The resultant FTSmax on implementing the CCH technique versus the FTSmax recorded while

implementing the CCHstable is presented in Figure 3. Determining co-flows and segment endpoints using the CCH technique managed to deliver a smaller FTSmax initially (for about 13 seconds) compared to the FTSmax recorded while implementing the CCHstable technique. Later, the FTSmax using CCH increased beyond the FTSmax recorded in the CCHstable technique because in the CCHstable technique co-flows are determined considering flows that are relatively long-lived which helped in determining more stable segment endpoints and hence the benefit of SR sustained for a longer duration.

But the key concern with the CCHstable technique is the duration required to reach the break-even point. The FTSmax is being badly affected because of the unusual mix of flows with respect to their characteristics at very few devices. A majority of them might be unusually tiny or elephant-sized and short-lived. The duration of about 13 seconds required to reach the break-even point on implementing the CCHstable necessitates the introduction of the Knapsack technique. In the Knapsack approach, the devices at which the FTSmax is badly affected are identified and the Knapsack technique is applied only at those devices for further reduction in FTS. The Knapsack size is chosen to be proportional to the difference between the FTS at that device and the average FTS. The Knapsack technique allows picking flows more aggressively at devices with higher FTS for SR, thereby maximizing chances for the reduction in FTSmax. The results show a significant reduction in the duration required to reach the break-even point after applying the Knapsack technique. The break-even point in the Knapsack approach is just above 1 second which is far better compared to the 13-second duration required in the CCHstable technique.

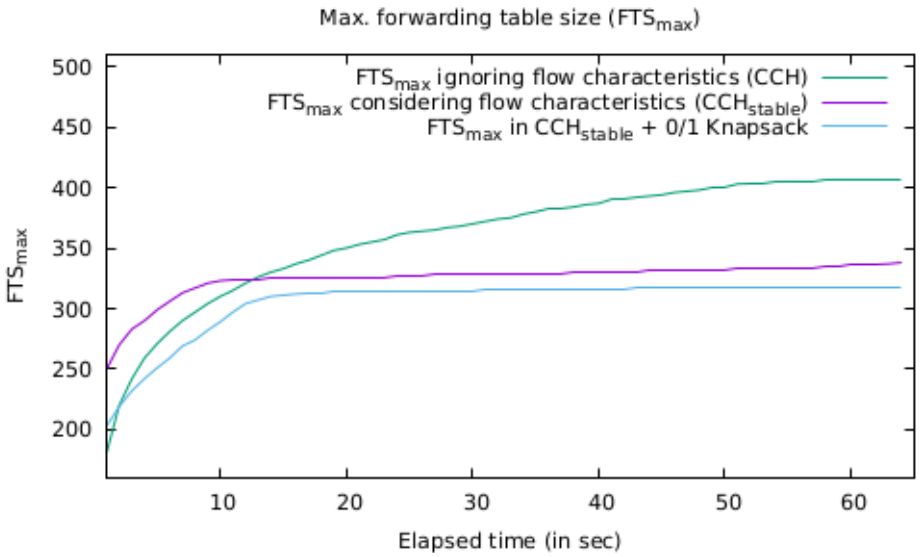

Fig.3. The variation in FTSmax over time

The segment endpoint distribution in the aforementioned Inet topology with 80 devices and 5*k* flows is presented in Fig. 4. The results show that approximately 21% of flows have chosen the same device as segment endpoint in CCH, whereas 18% of flows have chosen the same device as segment endpoint in the CCHstable technique. The higher the number of flows that choose the same node as segment endpoint, the higher are the chances for multiple flows to share a common forwarding table entry, which eventually leads to a significant reduction in FTS at that device. So as far as SR is concerned, it is good to have more flows choose the same node as the segment endpoint. The CCH version resulted in a higher percentage of flows choosing the same segment as endpoint initially, but later on, huge volatility in the co-flow count is observed in the CCH because the short-lived flows considered for determining segment endpoints terminate in very short duration and the newly generated flows are likely to result in a different segment endpoint. The overall volume of flows considered for SR in the CCHstable technique is less compared to the number of flows considered in the CCH. The number of flows considered for SR show a clear impact on the heuristic performance.

A considerable reduction in FTSmax can be observed when devices at the 90^{th} percentile FTS and above are considered for applying the Knapsack technique. Yet another notable observation is that applying the Knapsack technique at devices with 90^{th} percentile FTS and above resulted in a sharp rise in segment endpoint distribution at normalized node ID 1. After applying the Knapsack technique, close to 20% of flows have chosen the same device as the segment endpoint. Further, the resultant FTSmax on applying the Knapsack technique on devices through quartiles starting from the highest FTS is presented in Fig. 5. Fig. 5 shows that considering more devices from Quartile classes Q_3, Q_2, through Q_1 has no significant impact on reduction in FTSmax.

The variation in segment endpoint distribution after applying the Knapsack technique on devices at different levels of FTSmax is presented in Fig. 6. The resultant segment endpoint distribution exhibited minimal impact when more devices with lesser FTS are considered for the Knapsack technique. However, the key observation to be noted from Fig. 6 is that compared to the volume of flows considered for determining segment endpoints in CCHstable with the number of flows considered in CCH, the variation in segment endpoint distribution is negligible.

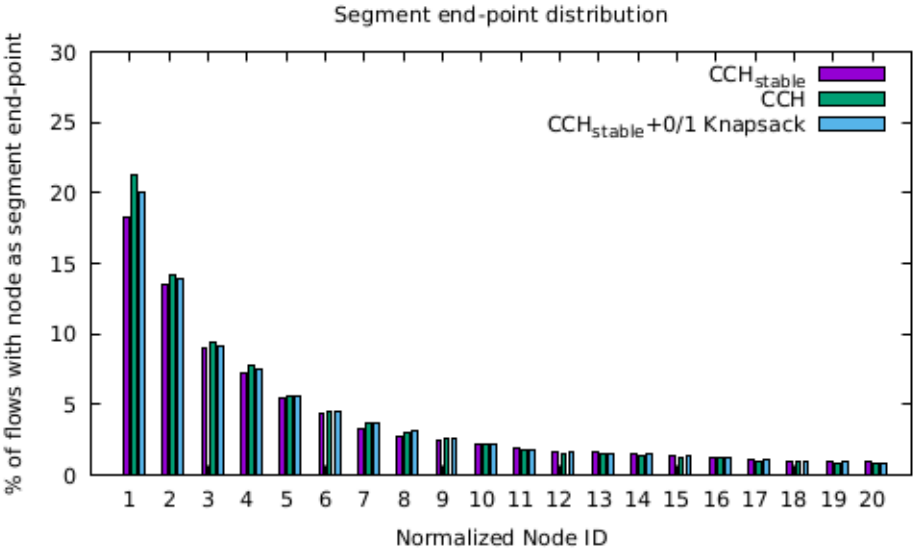

Fig.4. Segment endpoint distribution using different techniques

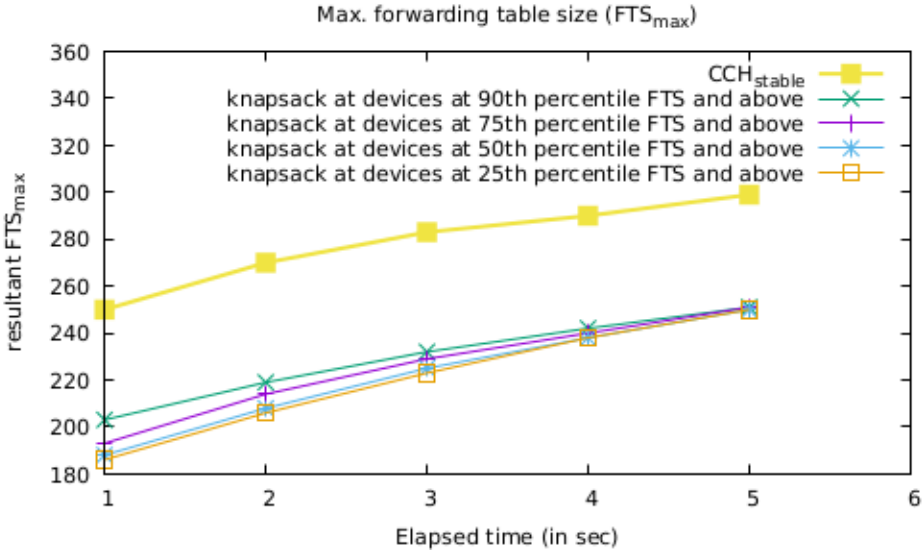

Fig.5. Resultant FTSmax

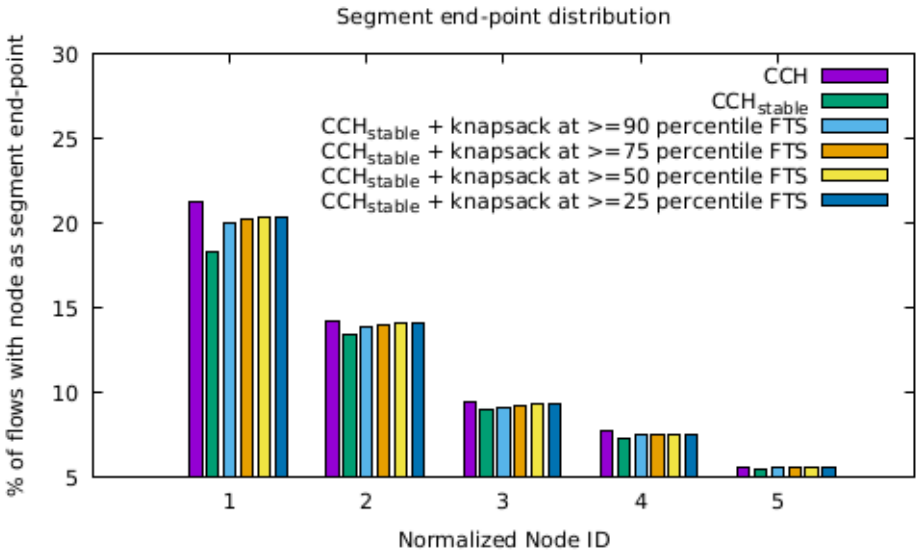

Fig.6. Segment endpoint distribution after applying the Knapsack technique

Due to the innovative traffic engineering practices that are being adopted in modern networks and the scale at which enterprise networks are growing, having complete information of network traffic is a challenge these days. However, due to the centralized view and fine-grained control of data traffic offered in SDNs, better accuracy can be attained through analyzing flows in terms of size and duration. Fig. 7 illustrates the variation in FTSmax with respect to input accuracy.

The confusion matrices considered for different levels of accuracy in flow classification are presented in Table 11 and Table 12 (actual vs predicted). The results show that with a considerable change in accuracy of the input data, the impact on the resultant FTSmax is apparent.

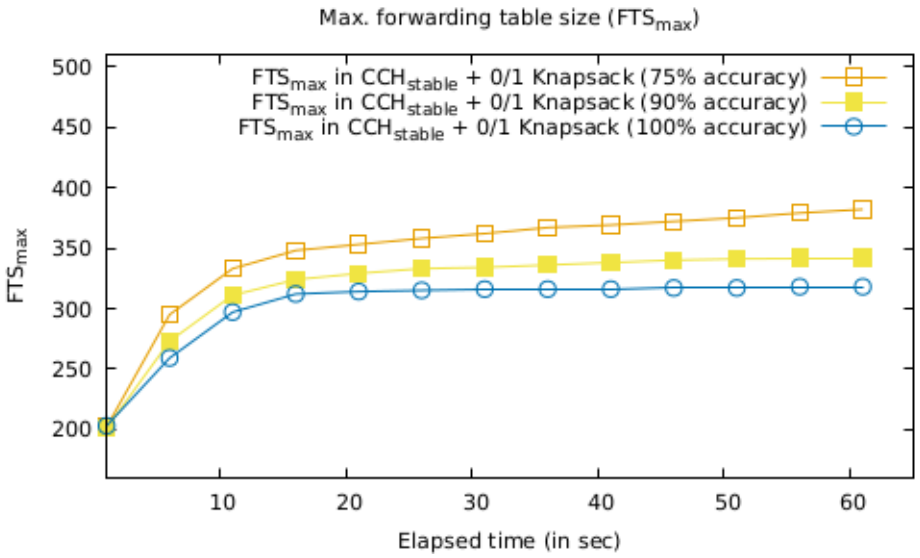

Fig.7. Sensitivity analysis of the proposed CCHstable technique

Table 11. Confusion matrix for 90% input accuracy

		Output class					
		tiny-short	**tiny-long**	**med-short**	**med-long**	**ele-short**	**ele-long**
Actual class	**tiny-short**	90%	2%	2%	2%	2%	2%
	Tiny-long	2%	90%	2%	2%	2%	2%
	Med-short	2%	2%	90%	2%	2%	2%
	Med-long	2%	2%	2%	90%	2%	2%
	Ele-short	2%	2%	2%	2%	90%	2%
	Ele-long	2%	2%	2%	2%	2%	90%

Table 12. Confusion matrix for 75% input accuracy

		Output class					
		tiny-short	**tiny-long**	**med-short**	**med-long**	**ele-short**	**ele-long**
Actual class	**tiny-short**	75%	5%	5%	5%	5%	5%
	Tiny-long	5%	75%	5%	5%	5%	5%
	Med-short	5%	5%	75%	5%	5%	5%
	Med-long	5%	5%	5%	75%	5%	5%
	Ele-short	5%	5%	5%	5%	75%	5%
	Ele-long	5%	5%	5%	5%	5%	75%

To realize the variance across the resultant initial FTSmax, the experiment is repeated 10 times with 80 devices and 5k flows. The descriptive statistics of the initial FTSmax and the confidence intervals in all three scenarios are presented in Table 13. The statistical data establishes the consistency of the proposed technique by delivering a similar initial FTSmax repeatedly.

Table 13. Descriptive statistics of initial FTSmax (Experiment repeated 10 times with 80 devices and 5k flows)

	FTS_before_ minimization	FTS_CCH	FTS_CCH_stable	FTS_CCH_stable +Knapsack
Mean	456.7	185.8	250.8	207.3
Standard Error	5.9778294091	2.1123972690	4.2993539566	2.9327650964
Mode	-	183	-	-
Median	455.5	187	253.5	207.5
First Quartile	449	183	242.25	202.25
Third Quartile	467	188.75	260.5	212.5
Variance	357.34444444	44.622222222	184.84444444	86.011111111
Standard Deviation	18.903556396	6.6799866932	13.595750970	9.2742175471
Kurtosis	-0.434868088	1.529278978	-0.36643887	-0.373516413
Skewness	-0.04552622	-0.55120198	-0.42741384	-0.165750316
Range	59	25	45	30
Minimum	426	172	226	191
Maximum	485	197	271	221
Sum	4567	1858	2508	2073
Count	10	10	10	10
Confidence Interval(95%)	[444.984, 468.416]	[181.66, 189.94]	[242.373, 259.227]	[201.552, 213.048]
Confidence Interval(90%)	[446.867, 466.533]	[182.325, 189.275]	[243.728, 257.872]	[202.476, 212.124]

7. Conclusions

A smaller initial FTSmax can be achieved in the CCH technique because a better flow convergence can be attained when a large population of flows is considered for computing the heuristic value. The heuristic performance improves when a large number of network flows are considered for determining the segment endpoints. However, considering too many flows for SR is a trade-off because it involves significant overhead.

In the CCH technique, since a large number of flows are considered for SR, the number of flows allowed to share the forwarding rules is also high which eventually led to a better reduction in FTSmax. Despite a smaller initial FTSmax, a rapid rise can be observed while implementing the CCH technique because many short-lived flows that are considered for determining segment endpoints have terminated, and the newly established connections led to a surge in the FTSmax. The newly initiated flows mandated exclusive forwarding table entries leading to a rapid increase in the size of the forwarding tables.

Compared to CCH, the FTSmax in the CCHstable technique is initially high because in the CCHstable only long-lived medium-sized flows are considered for SR. The key observation is that the FTSmax in CCH technique increased rapidly due to ignoring the flow characteristics. CCH underperformed the CCHstable technique over time. Further, applying the analogy of the Knapsack technique for selecting flows at devices with higher FTS allowed the CCHstable technique to significantly reduce the time required to reach the break-even point. The proposed CCHstable technique helped in determining more stable segment endpoints, and eventually, a smaller FTSmax can be witnessed proving that the benefit of SR is sustained for a longer duration.

References

[1] Yaoqing Liu, Beichuan Zhang, and Lan Wang. Fifa:Fast incremental fib aggregation. In *2013 Proceedings IEEE INFOCOM*, pages 1–9, 2013.

[2] Gagandeep Garg, Roopali Garg, "Accurate Anomaly Detection using Adaptive Monitoring and Fast Switching in SDN", International Journal of Information Technology and Computer Science, vol.7, no.11, pp.34-42, 2015.

[3] Kshira Sagar Sahoo, Sambit Kumar Mishra, Sampa Sahoo, Bibhudatta Sahoo,"Software Defined Network: The Next Generation Internet Technology", International Journal of Wireless and Microwave Technologies, Vol.7, No.2, pp.13-24, 2017.

[4] Luca Davoli, Luca Veltri, Pier Luigi Ventre, Giuseppe Siracusano, and Stefano Salsano. Traffic engineering with segment routing: Sdn-based architectural design and open source implementation. In *2015 Fourth European Workshop on Software Defined Networks*, pages111–112, 2015.

[5] Liaoruo Huang, Qingguo Shen, Wenjuan Shao, and Cui Xiaoyu. Optimizing segment routing with the maximum sld constraint using openflow. *IEEE Access*,6:30874–30891, 2018.

[6] Francesco Lazzeri, Gianmarco Bruno, Jeroen Nijhof, Alessio Giorgetti, and Piero Castoldi. Efficient label encoding in segment-routing enabled optical networks. In *2015 International Conference on Optical Network Design and Modeling (ONDM)*, pages 34–38, 2015.

[7] Olivier Dugeon, Rabah Guedrez, Samer Lahoud, and Géraldine Texier. Demonstration of segment routing with sdn based label stack optimization. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 143–145, 2017.
[8] Alessio Giorgetti, Piero Castoldi, Filippo Cugini, Jeroen Nijhof, Francesco Lazzeri, and Gianmarco Bruno. Path encoding in segment routing. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2015.
[9] Rabah Guedrez, Olivier Dugeon, Samer Lahoud, and Géraldine Texier. Label encoding algorithm for mpls segment routing. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 113–117, 2016.
[10] Antonio Cianfrani, Marco Listanti, and Marco Polverini. Translating traffic engineering outcome into segment routing paths: The encoding problem. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 245–250, 2016.
[11] Randeep Bhatia, Fang Hao, Murali Kodialam, and T.V.Lakshman. Optimized network traffic engineering using segment routing. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 657–665, 2015.
[12] Eduardo Moreno, Alejandra Beghelli, and Filippo Cugini. Traffic engineering in segment routing networks. *Computer Networks*, 114:23–31, 2017.
[13] Anix Anbiah and Krishna M. Sivalingam. Sr domain partitioning in segment routed sdns. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pages 153–156, 2019.
[14] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), April 2009.
[15] Anix Anbiah and Krishna M. Sivalingam. Optimal segments for forwarding table size minimization in segment-routed sdns. *International Journal of Network Management*, n/a(n/a):e2142. e2142 nem.2142.
[16] Pier Luigi Ventre, Stefano Salsano, Marco Polverini, Antonio Cianfrani, Ahmed Abdelsalam, Clarence Filsfils, Pablo Camarillo, and Francois Clad. Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results. *IEEE Communications Surveys Tutorials*, 23(1):182–221, 2021.
[17] Zahraa N. Abdullah, Imtiaz Ahmad, and Iftekhar Hussain. Segment routing in software defined networks: A survey. *IEEE Communications Surveys Tutorials*, 21(1):464–486, 2019.
[18] Google. ClusterData, 2019 (accessed in February, 2020). https://github.com/google/cluster-data/blob/master/ClusterData2019.md.
[19] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *Proceedings of the 2015 ACM Conference on SpecialInterest Group on Data Communication*, SIGCOMM'15, page 183–197, New York, NY, USA, 2015. Association for Computing Machinery.
[20] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. Scaling memcache at facebook. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, page 385–398, USA, 2013. USENIX Association.
[21] Jared Winick and Sugih Jamin. Inet-3.0: Internet topology generator, 08 2002.

Authors' Profiles

Bommareddy Lokesh received Bachelor of Technology degree in Information Technology from PVP Siddhartha Institute of Technology, Vijayawada in 2011 and Master of Technology degree in Computer Science and Technology from Andhra University College of Engineering in 2013. He is currently a research scholar in the department of Computer Science and Engineering at National Institute of Technology Puducherry, Karaikal. His research interests include software-defined networking, Internet of Things, and network function virtualization, 5G communications.

Narendran Rajagopalan received Bachelor of Engineering degree in Information Science and Engineering from Vidyavikas Institute of Engineering and Technology, Mysore in 2004 and Master of Technology in Networking and Internet Engineering from Sri Jayachamarajendra College of Engineering, Mysore in 2007 and PhD degree in Computer Science and Engineering from National Institute of Technology Trichy, Trichy in 2013. He is currently working as an Assistant Professor and Head in the department of Computer Science and Engineering at National Institute of Technology Puducherry, Karaikal. His research interests include wireless sensor networks, software-defined networking, data center networks, 5G communications, and Blockchain.

How to cite this paper: Bommareddy Lokesh, Narendran Rajagopalan, "Flow-aware Segment Routing in SDN-enabled Data Center Networks", International Journal of Computer Network and Information Security(IJCNIS), Vol.15, No.5, pp.50-62, 2023. DOI:10.5815/ijcnis.2023.05.05