Distributed Congestion Control in LEO Satellite Networks

Pranav Page
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India

Gaurav Kasbekar
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India

Kaustubh Bhargao
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India

Hrishikesh Baviskar
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India

***Abstract*—Satellite communication in LEO constellations has become an emerging topic of interest. Due to the high number of LEO satellites in a typical constellation, a centralized algorithm for minimum-delay packet routing would incur significant signaling and computational overhead. We can exploit the deterministic topology of the satellite constellation to calculate the minimum-delay path between any two nodes in the satellite network, but that does not take into account the traffic information at the nodes along this minimum-delay path.**

We propose a distributed probabilistic congestion control scheme to minimise end-to-end delay. In the scheme, each satellite, while sending a packet to its neighbour, adds a header with a simple metric indicating its own congestion level. The decision to route packets is taken based on the latest traffic information received from the neighbours. We build this algorithm onto the Datagram Routing Algorithm, which provides the minimum delay path, and the decision for the next hop is taken by the congestion control algorithm. We compare the proposed congestion control mechanism with the existing congestion control used by the DRA, and show improvements over the same.

I. Introduction

With the advent of cost-effective space launch systems, the feasibility of space-based communication networks has turned into a reality. Dense Low Earth Orbit constellations such as Starlink and OneWeb have joined sparse constellations like Iridium in orbit and are operational. Communication using LEO satellite constellations is favoured over GEO satellites due to the much lower ground-to-satellite propagation delay.

The challenges faced by satellite constellations are very different from those encountered by terrestrial networks. The nodes in a satellite constellation are constantly moving relative to the ground, so association and handover in a ground-to-satellite link are non-trivial problems. The satellites typically deployed in a constellation are small in size (about 150 kg), which results in limited on-board processing and storage capacity. In addition, the small size of the satellites leads to difficulties in antenna pointing. The inter-satellite links are also characterized by high propagation and transmission delays and high BER. Limited on-board storage capacity leads to packet drops when the nodes get congested, thus degrading the flow of packets. The network has two types of inter-satellite links (ISLs), namely *intra-plane* ISLs, which are the ISLs between two neighbouring satellites in the same orbital plane and *inter-plane* ISls, which are the ISLs between two neighbouring satellites in different orbital planes. The inter-plane ISLs are difficult to maintain in the polar regions due to the rapid movement of satellites and switching of relative positions.

Due to the dynamic nature of the satellite constellation, paths computed at a central location and sent to the nodes in the network would need a lot of transmissions and computations involving a dense network. Thus, a distributed routing and congestion control algorithm is preferred. The DRA takes advantage of the spherical geometry of the network and calculates the optimum minimum delay path using the relative positions of the nodes. After that, it is the job of the congestion control algorithm to pick the next hop for a packet to reach its destination with the minimum queueing and propagation delay. The problem of choosing the next hop for the packet in the presence of congestion is the problem that this work focuses on. The choice has to be made locally, and without knowledge of the congestion level of every node along the minimum delay path.

A distributed congestion control algorithm that can deal with uneven node congestion levels to route packets from source to destination with low packet drops and end-to-end delay would be easy to implement on-board, and would offer better QoS.

The problem of choosing an optimum schedule which minimizes the total end-to-end delay for a given set of packets has been shown to be NP-hard in [1]. Thus, heuristics-based approaches have been used to tackle this problem. In particular, [2] [3] use a basic threshold on the outgoing buffer to determine whether the link is congested. The DRA does not use local congestion information to reroute packets. We address this problem by using the packet headers as a way of conveying traffic information in the form of a single metric indicating the congestion level, and then probabilistically choosing the next hop for a packet. We simulate a typical LEO satellite constellation, and compare the performance of the DRA and our own algorithm in terms of end-to-end delay

and packet drops.

II. Related works

Werner et al. [4] proposed a routing algorithm based on the Asynchronous Transfer Mode of operation of data transmission, a connection-oriented mode of transmission, with packets from the source node to the destination node following the same path (packet sequence is maintained). The authors propose that for meeting QoS requirements and dealing with the highly dynamic (yet predictable) topology of the satellite constellation, a connection-oriented routing scheme would perform well. The algorithm makes use of the deterministic nature of the constellation topology and generates a fixed number of virtual topology slices on which routing is performed. An iterative modified Dijkstra's algorithm is used to choose a fixed number of distinct shortest paths between any two source destination pairs.

Ekici et al. [2] [3] proposed a distributed routing algorithm which assumes satellites to be located at virtual locations in space.The algorithm focuses on datagram routing between virtual nodes, as described in [2] which is a connection-less framework. The algorithm exploits the geographical locations of the source and destination, and the fact that inter-plane inter-satellite links are shorter closer to the poles. This is a distributed algorithm, and thus, has low computational overhead. Congestion control is employed by using a simple threshold for the queue to a particular node.

Korçak [5] deals with making a choice between the multiple shortest paths available in the network. A priority mechanism is proposed, which takes into account the past utilization and buffering at each link. Liu et al [6] also involves topology snapshots, i.e, it is a time virtualization algorithm. It aims to reduce the computational complexity of routing using Dijkstra's algorithm.

Bertsekas [7] discusses an asynchronous distributed algorithm for a broad class of dynamic programming problems. The paper studies a computation algorithm to find the length of the shortest path between a source node and destination node, which involves nodes transmitting their estimates of the length of the shortest path to their neighbours, which then reevaluate their own estimates and transmit.The paper provides proof for the claim that as the process continues infinitely, the estimate stored in each node approaches the optimal solution.

Clement et al. [1] look at the computational complexity of finding an optimal schedule to send messages from a source node to a destination node in a store-and-forward network. In this paper, a schedule refers to a strategy of assigning free buffers (the next nodes for the hop) to tokens (messages). The decision problem which consists of deciding whether a schedule exists whose completion time is bounded by some fixed given value $k \geq 0$ is proved to be NP-complete in [8]. The paper shows that the above problem, when modified to have the parameter k to be not part of the input, is still NP-complete. Since the problem of finding a schedule with a minimum completion time is not solvable in polynomial time, the authors look at approximation algorithms. It is proven that the problem is non-approximable. The paper [9] is a follow-up to the paper [1], and focuses on the hardness of approximating the schedule with the minimum end-to-end delay. The authors prove that the minimum end-to-end delay cannot be approximated within an error of $k^{\frac{1}{10}}$, where k is the number of messages that are being sent. The paper then studies algorithms in which routing is performed with every node having local knowledge only (neighbouring nodes). It is proven that approximating the minimum end-to-end delay with respect to local strategies within an approximation error bounded by $f(k)$ where $f(\cdot)$ is any sublinear function, is NP-hard. the authors then conclude with some polynomial time heuristics which could inspire some heuristics which approximate the solution better, in an average sense.

Dai et al. [10] introduce a modification of the DRA algorithm as proposed in [2] [3]. The modified algorithm, DRAW, focuses on two main failings of the DRA algorithm, namely routing failure in the case of satellite failures, and absence of counter-seam links in most LEO networks. It achieves a better tolerance to failure than DRA by sacrificing propagation delay (routing packets along longer paths) for lower packet loss rate.

In [11], routing is done based on comparing the locations of the source and destination satellite. The authors tackle the problem of continuously evolving link states (e.g. when satellites go into the polar region) by mapping the sub net of satellites onto a plane and determining the real time inter satellite link state by using the link state broadcast, on which the breadth first tree search is performed. Thus, this technique uses routing tables (similar to [3]), but also uses the real time link states to perform routing.

Dai et al. [12] propose a Distributed Congestion Control Routing protocol based on traffic classification in LEO satellite networks. The routing is performed (as in previous works) using the geographical locations of the source and destination nodes, and their relative positions to the seam and polar regions. The data traffic is divided into three types, namely delay-sensitive, throughput-sensitive, and ordinary, and different routing strategies are devised for each type. It is observed that as higher latitude hops are preferred for their shorter path lengths, the traffic at higher latitudes will experience congestion. The path calculations, after judging the congestion levels at each node, are done by using Dijkstra's algorithm.

Qi et al. [13] focuses on performing distributed routing in constellations where the satellites are in inclined orbits (as opposed to polar orbits). Inclined orbits do not pass over the poles, and thus, inter plane ISLs do not need to be switched off, which makes this constellation have a near-constant topology. The approach taken by this work makes use of the predictable mesh-like topology to route packets based on the logical location of the source-destination pair (as done in previous works). A restricted flooding approach is used to tackle link and node failures.

The Dynamic Source Routing algorithm [14] is a connection-based routing scheme, and thus, suffers from the typical drawbacks such as path handover, initialisation time, etc. The Low Complexity Probabilistic Routing [15] describes a

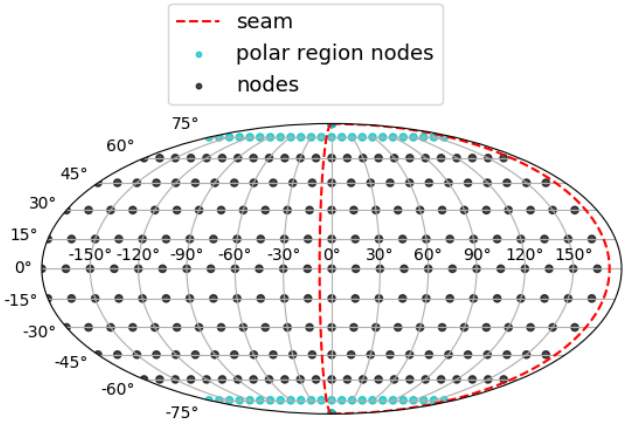

Fig. 1. Satellite constellation with $N = 12, M = 24$ and counter-rotating seam indicated

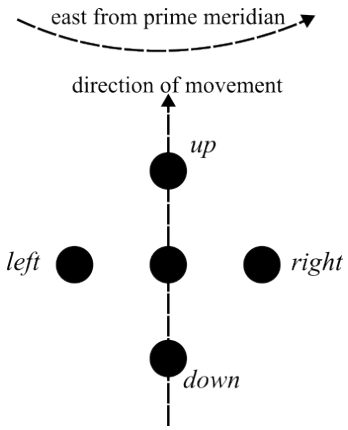

Fig. 2. Neighbours of a node with give orientation with respect to prime meridian and given direction of movement

probabilistic routing technique for polar orbits with a simpler priority metric.

III. System Model

The Low Earth Orbit satellite constellation considered is a Walker star constellation, with satellites in polar orbits. The setup and terminology used is based on the terminology used in [2]. There are N orbital planes, with M satellites per plane. Thus, the angular spacing between the planes is $360^\circ/2N$. All satellites are at a fixed altitude h from the ground, thus forming an orbital shell. We consider the network to be comprised of *virtual nodes*, as in [2], with different satellites occupying the virtual nodes at different instances of time. The virtual nodes are fixed in location with respect to the ground, and are filled up by the nearest satellite. Each virtual node location can be expressed in terms of the plane number p and the satellite number s, $0 \leq p \leq N-1$, $0 \leq s \leq M-1$. Each node in the network can be represented as a tuple (p, s). This model does not deal with the mobility of satellites, and can be used to perform routing based entirely on the position of the virtual nodes.

As this is a Walker star constellation, each node has two neighbours in the same orbital plane, i.e, *up* and *down*, as well has two neighbours in different orbital planes, i.e, *left* and *right*, as seen in Figure 2. The direction *up* is the direction of movement of the satellites in the orbits, while the direction *right* is the direction East from the prime meridian (longitude 0°). Due to the circular arrangement of orbital planes, there exist counter-rotating seams where a satellite moving north has have a neighbour to its left who is moving south. Some works operate with these inter-plane ISLs disabled, with high Doppler effects and difficulties in antenna pointing as justifications, but we have considered advances in beamforming and antenna tracking, and have decided to use these links across seams as operational links. Inter-plane ISLs in the polar regions, however, are considered to be shut off due to the change in the orientation of neighbours. Polar regions are defined using a latitude threshold θ_{polar}, with $\theta_{\text{polar}} = 75^\circ$ taken as the default boundary. The latitudes above θ_{polar} are considered to be in the polar regions.

Each node has four output buffers, corresponding to its four neighbours. We assume that on one link, reception and transmission can take place simultaneously. Thus, all four antennas can be transmitting and receiving simultaneously.

The lengths of the intra-plane ISLs are the same, as nodes are equally spaced in the orbital planes. For an orbital radius of R, the length of an intra-plane ISL L_v is given by (1), while that of an inter-plane ISL situated at latitude θ is given by (2)

$$L_v = R\sqrt{2(1-\cos(360^\circ/M))} \tag{1}$$

$$L_h = R\cos\theta\sqrt{2(1-\cos(360^\circ/2N))} \tag{2}$$

As seen in Figure 1, the inter-plane ISLs are shorter towards the poles and longer towards the equator.

IV. Problem Formulation

Now that the system model has been defined, the problem statement reduces to the following : Given the topology of the network of nodes and a destination node, without any global knowledge of the congestion level at each node, pick the next hop for the packet from the source node in a way that the end-to-end delay ($d_{\text{prop}} + d_{\text{queueing}}$) is minimized.

V. Algorithms

A. Routing

1) DRA direction estimation: DRA has three steps, namely direction estimation, direction enhancement and congestion control. In direction estimation, the path from source to destination with the minimum number of hops is chosen. Two different paths are considered, P^H and P^V, where P^H is the optimal among all paths which do not cross the polar regions, and P^V is the optimal path among all paths which do cross the polar regions. This phase chooses which path (P^H or P^V) would be best by comparing the number of hops in both. Each path can be described by four metrics, namely d_h, n_h, d_v, n_v where (d_h, n_h) describe the direction

in the horizontal direction and the number of horizontal hops respectively. In the end, the two paths are compared, and the path with minimum $(n_v + n_h)$ is returned.

2) DRA direction enhancement: This phase actually takes into account the unequal interplane ISLs in the network, and the transport of packets in the polar region. In this phase, the directions given by d_h and d_v are labelled primary or secondary, depending on the position of the packet source and destination. The decision of whether to name a direction as primary or secondary is influenced by factors like the latitudes of the two nodes, the number of horizontal hops needed, the possibility of sending the packet through a path closer to the polar regions, thus taking the horizontal hops closer to the poles. The primary and secondary directions are passed to the congestion control algorithm.

B. Congestion control

1) DRA congestion control: In DRA, the decision to send packets in the primary or secondary directions is taken on the basis of the congestion level of the node's output buffers in the respective directions. If the output buffer in the primary direction has less than $N_{threshold}$ packets, then the packet is sent in the primary direction. Otherwise, if the secondary direction exists (might not exit for nodes in polar regions), and the output buffer in the secondary direction has less than $N_{threshold}$ packets, then the packet is sent in the secondary direction. If the buffers in both primary and secondary directions have buffers of size greater than $N_{threshold}$, then the packet is sent in the primary direction.

2) New congestion control: The proposed congestion control algorithm takes advantage of the dense network of nodes to transmit traffic information about the node to its neighbours. With each packet sent to its neighbour, the node adds a traffic metric to the header of the packet. Similarly, when a packet is received, the traffic metric in the header is extracted and stored in each node. Thus, each node maintains an output buffer and the latest traffic metric received for each of its neighbours. The decision to be taken involves the lengths of the output buffers as well as the traffic metric in each of the two directions.

The traffic metric to be sent has to be indicative of the congestion level of the node. A weighted sum is used in (3)

$$m_{node} = \sum_{i=1, i\neq j}^{4} (w_{neighbour} m_{node,i} + (1 - w_{neighbour}) N_{node,i}) \quad (3)$$

where m_{node} is the traffic metric to be sent from $node$ to the neighbour j, $w_{neighbour} \in [0, 1]$ is the weight given to the values of the traffic metric of the neighbours of the $node$, $N_{node,i}$ is the length of the output buffer in $node$ towards neighbour i, and $m_{node,i}$ is the latest traffic metric received by $node$ from neighbour i

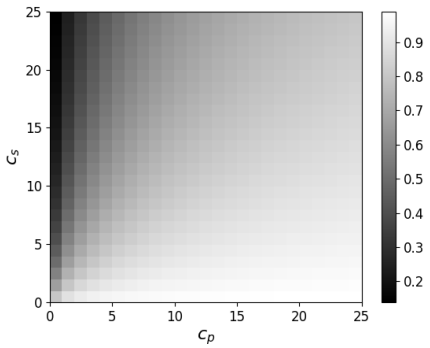

Fig. 3. $P(primary)$ with $p_{preference} = 0.8$

Let $primary$ and $secondary$ be the primary and secondary directions given by the direction enhancement algorithm. If $N_{node,primary}$ and $N_{node,secondary}$ are the output buffer lengths in the respective directions, and $m_{node,primary}$, $m_{node,secondary}$ are the traffic metrics received from the primary and secondary directions and stored in $node$, then the following congestion level metrics are calculated

$$c_i = w_{buffer} N_{node,i} + (1 - w_{buffer}) m_{node,i} \quad (4)$$

for $i \in \{primary, secondary\}$, $w_{buffer} \in [0, 1]$. A probability distribution is chosen such that

$$P(\text{choose primary}) = \frac{(c_s + 1) p_{preference}}{c_p + 1 + (c_s - c_p) p_{preference}} \quad (5)$$

where $c_s = c_{secondary}$, $c_p = c_{primary}$, and $p_{preference}$ is the probability that primary is chosen when $c_s = c_p$. The probabilistic nature of this choice ensures that a single direction is not clogged, which can happen in the DRA congestion control algorithm. Also, this ensures that the primary direction is not completely abandoned if it is clogged, the algorithm sends packets to it at a reduced rate. This leads to lower average delay, but slightly higher packet drops. If c_p increases, then the probability of choosing the secondary direction increases. Preference is still given to the primary direction.

VI. Simulation setup

The simulation was done in Python 3. A constellation with 12 polar orbital planes and 24 satellites per plane was used. Thus, $N = 12$ and $M = 24$ were the parameters for the constellation, with an inclination of 90° and an altitude of 600 km. A discrete event simulator was built which executed events in order of lowest time of execution.

A portion of the network was simulated, bounded by the nodes $(2, 3), (7, 3), (7, 9), (2, 9)$, consisting of 42 nodes. The simulation was fed with Poisson arrivals with rate λ_{in}. After every t_{step} seconds, n_{pairs} source-destination pairs were chosen uniformly randomly from the 42 nodes, and $n_{packets}$ packets were queued up for these pairs with exponential inter-arrival times. Thus, packets were fed into the network for some time,

TABLE I
DEFAULT PARAMETERS FOR SIMULATION SETUP

Parameter name	Default Value
Time step (t_{step})	10 ms
Feed stop time (t_{stop})	50 ms
Number of pairs (n_{pairs})	20
Input flow rate (λ_{in})	1.5×10^4 packets/s
Number of packets per flow ($n_{packets}$)	100
Transmission rate (r_{tx})	25 Mbps
Packet size	1 kB
Buffer size (N_{buffer})	200
Buffer threshold ($N_{threshold}$)[1]	150
$p_{preference}$[1]	0.9
$w_{neighbour}$[1]	0.25
w_{buffer}[1]	0.8

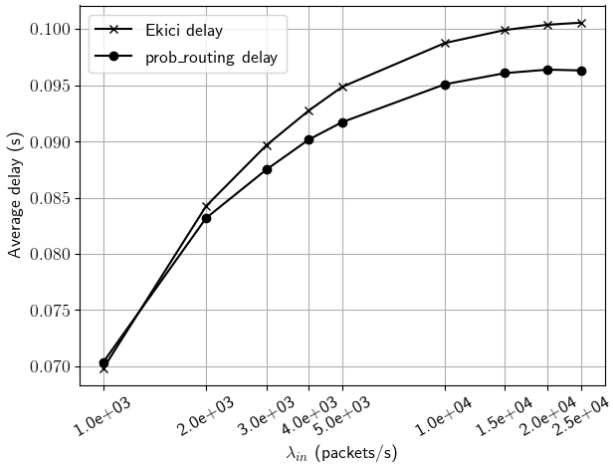

Fig. 4. Comparison of end-to-end delay variation with λ_{in}

to congest most nodes in the network. Then transmissions were stopped, and the network was allowed to decongest. The default values chosen for the simulation are shown in Table I

VII. RESULTS

The probabilistic routing algorithm performs better than the DRA in terms of average end-to-end delay for higher input rates (λ_{in}) to the network (Fig. 4). The tradeoff is that average fraction of packet drops also increases, due to not completely shutting down sending packets in the primary direction of each node, as seen in Fig. 5.

When N_{buffer} is changed, the buffer sizes in each node are increased. The ratio $N_{threshold}/N_{buffer}$ is fixed to be 0.75, to study the effect of changing N_{buffer} primarily. As expected, the packet drops decrease with increasing buffer sizes for both algorithm (Fig. 7). For large buffers, both algorithms perform similarly as no node gets congested. For lower buffer sizes, the probabilistic routing scheme performs better than the DRA in terms of end-to-end delay (Fig. 6).

A different experiment was also performed, with only packets in a single flow analysed for end-to-end delay. This was done

[1]Chosen via simulations to minimize delay and packet drops

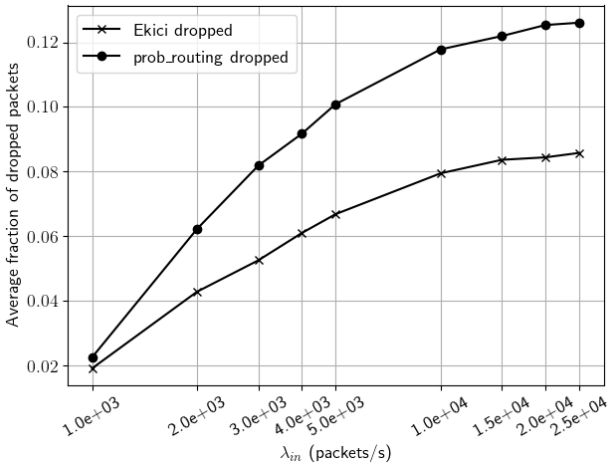

Fig. 5. Comparison of average fraction of packet drops with λ_{in}

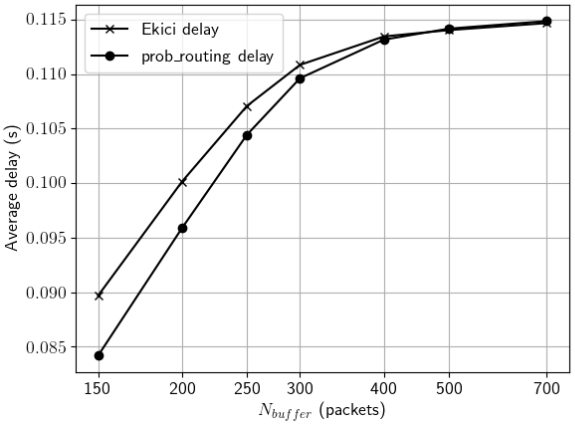

Fig. 6. Comparison of end-to-end delay variation with N_{buffer}

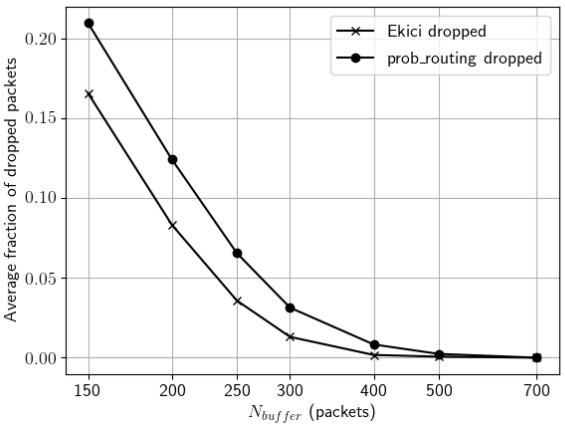

Fig. 7. Comparison of average fraction of packet drops with N_{buffer}

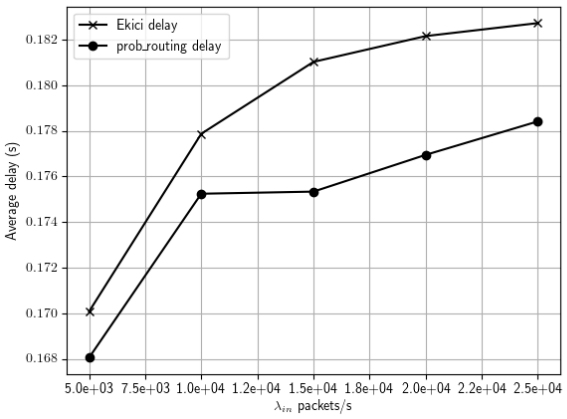

Fig. 8. Comparison of end-to-end delay variation with λ_{in} for a single flow

to check the difference in average end-to-end delay for packets which originated from the same source node and ended in the same destination node. It is seen that probabilistic routing offers improvements in average end-to-end delay for the single flow too (Fig. 8)

VIII. Conclusions and Future Work

The new probabilistic congestion control algorithm offers lower average end-to-end delay in congested networks, as compared to the existing DRA congestion control algorithm. It uses the regular nature of the topology to exchange information, and effectively route packets in a probabilistic manner. For now, this scheme does not account for satellite or link failures. This could be done by flooding the network, or by incorporating a different header into the packets being sent by neighbouring nodes to indicate local link failures. This scheme also does not ensure loop-free routing, which can be incorporated by maintaining a partial list of nodes visited by the packet in the header. Since each node can be represented by a virtual location in the form of (p, s), the overhead would not be very large. The traffic metric could be improved by using an exponentially weighted moving average instead of the latest value. The $p_{preference}$ value can be adjusted dynamically depending on the congestion level in the network and packet drops.

References

[1] A. Clementi and M. Di Ianni, "Optimum schedule problems in store and forward networks," in *Proceedings of INFOCOM '94 Conference on Computer Communications*, pp. 1336–1343 vol.3, 1994.

[2] E. Ekici, I. Akyildiz, and M. Bender, "Datagram routing algorithm for leo satellite networks," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, pp. 500–508 vol.2, 2000.

[3] E. Ekici, I. Akyildiz, and M. Bender, "A distributed routing algorithm for datagram traffic in leo satellite networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 137–147, 2001.

[4] M. Werner, C. Delucchi, H.-J. Vogel, G. Maral, and J.-J. De Ridder, "Atm-based routing in leo/meo satellite networks with intersatellite links," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 1, pp. 69–82, 1997.

[5] O. Korcak and F. Alagoz, "Analysis of priority-based adaptive routing in satellite networks," in *2005 2nd International Symposium on Wireless Communication Systems*, pp. 629–633, 2005.

[6] Y. Liu and L. Zhu, "A suboptimal routing algorithm for massive leo satellite networks," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–5, 2018.

[7] D. Bertsekas, "Distributed dynamic programming," *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 610–616, 1982.

[8] D. Bovet and P. Crescenzi, "Minimum-delay schedules in layered networks," in *Acta Informatica*, vol. 28, pp. 453–461, 1991.

[9] A. Clementi and M. Di Ianni, "On the hardness of approximating optimum schedule problems in store and forward networks," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 272–280, 1996.

[10] H. Liu, F. Sun, Z. Yang, and F. Long, "A novel distributed routing algorithm for leo satellite network," in *2012 International Conference on Industrial Control and Electronics Engineering*, pp. 37–40, 2012.

[11] C. Liu and Y. Liu, "A real-time distributed algorithm for satellite constellation routing," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, pp. 745–749, 2018.

[12] S. Dai, L. Rui, S. Chen, and X. Qiu, "A distributed congestion control routing protocol based on traffic classification in leo satellite networks," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 523–529, 2021.

[13] X. Qi, B. Zhang, and Z. Qiu, "A distributed survivable routing algorithm for mega-constellations with inclined orbits," *IEEE Access*, vol. 8, pp. 219199–219213, 2020.

[14] W. Peng, Y. Jian, C. Zhi-gang, and W. Jing-lin, "Dynamic source routing algorithm in low-earth orbit satellite constellation," in *2006 International Conference on Communication Technology*, pp. 1–4, 2006.

[15] X. Liu, Z. Jiang, C. Liu, S. He, C. Li, Y. Yang, and A. Men, "A low-complexity probabilistic routing algorithm for polar orbits satellite constellation networks," in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–5, 2015.