Distributed Congestion Control in LEO Satellite Networks

***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

What is the problem?

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.

Why is it interesting and important?

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.

Why is it hard?

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. Add figures of improvement. Add breakdown of the paper(?)

II. Related works

Do last

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 $360^{\circ}/2N$. (Add figure of satellite constellation). 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*. 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°). (Add figure denoting neighbours)

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. (Add figure denoting switching of left and right 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 constellation figure, 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.

Mathematically, (Add mathematical expression for the path)

V. Algorithms

(Briefly explain DRA direction estimation, DRA direction enhancement, DRA congestion control)

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.

calculation of the traffic metric

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

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

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

probabilistic decision making

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

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

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

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

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. If c_p increases, then the probability of choosing the secondary direction increases. Preference is still given to the primary direction. (Add a heat map for probability vs c_p, c_s)

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, to congest most nodes in the network. Then transmissions were stopped,

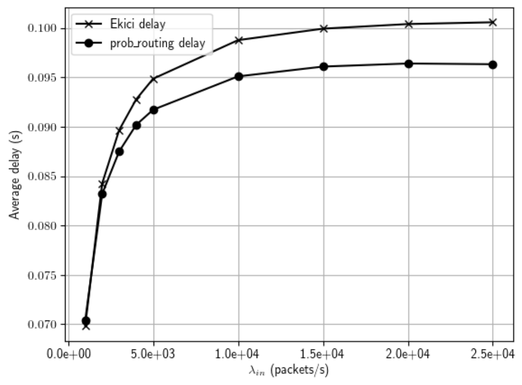

Fig. 1. Comparison of end-to-end delay variation with λ_{in}

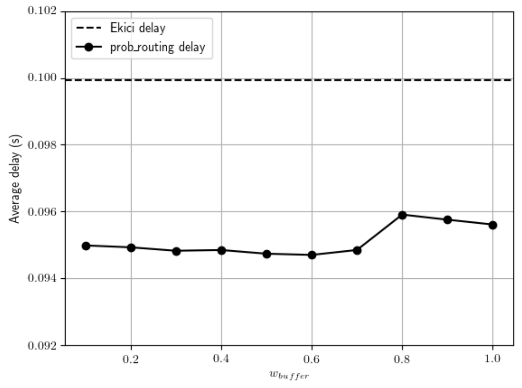

Fig. 2. Comparison of end-to-end delay variation with w_{buffer}

and the network was allowed to decongest.

The inter-satellite links were chosen to have a transmission rate of $r_{tx} = 25$Mbps. The packet size was chosen to be 1kB, which made the transmission delay to be 0.327 ms. Each output buffer was chosen to be of the same size, $N_{buffer} = 200$ packets. The threshold $N_{threshold}$ was chosen to be $N_{threshold} = 150$.

(Add random seeds)

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 (Figure 1). (Add interpretation of results).

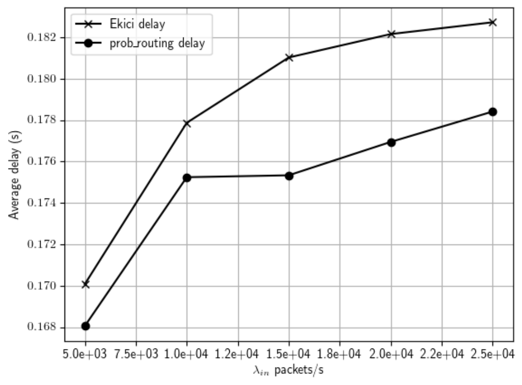

Fig. 3. Comparison of end-to-end delay variation with λ_{in} for a single flow

A different experiment was also performed, with only packets in a single flow analysed for end-to-end delay. This was done 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. The source node was $(2, 3)$ and the destination node was $(7, 9)$

VIII. Conclusions and Future Work

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.