JMR: Jitter-Minimizing Routing Algorithm for LEO Satellite Networks

John Nowinski
Michigan State University
East Lansing, MI. USA
nowins15@msu.edu

Jude Hansen
Michigan State University
East Lansing, MI. USA
hanse237@msu.edu

***Abstract*—This paper proposed the jitter-minimizing routing algorithm for LEO satellite networks (JMR). JMR combines a forward look-ahead with anchor-based routing in order to efficiently create routes that are stable across time. Simulation results show 13% improved jitter compared to localized location-based algorithms and nearly 50% speedup compared to a similar look-ahead routing method.**

***Index Terms*—LEO Networks, ISL routing, Jitter, Satellites**

I. Introduction

A. Overview

Satellite constellation networks are a rapidly growing pillar for global connectivity, increasing network availability and reliability in areas the traditional internet backbone can not reach. Constellation networks, like SpaceX's Starlink, rely on dense low earth orbit (LEO) satellites that can communicate both with ground stations on the earth and between each other utilizing inter-satellite links (ISL). ISL's allow satellites to efficiently move traffic through the constellation without relying on the "bent-pipe"-style of satellite to ground station to satellite communication. Satellite constellations are separated into orbital planes, each containing satellites with the same orbit around the earth. Intra-plane ISLs connect adjacent neighbors within the same plane, while cross-plane ISLs bridge satellites between different planes providing global reach across the network. The inherently time-varying nature of LEO systems are highly dynamic. As satellites in cross planes move, the slant distance between them and pointing angles vary drastically, shifting propagation delay on the order of 10 to 100s of milliseconds. Offered traffic load varies heavily depending on geographic demand and cross-plane visibility, resulting in queuing delay spikes as traffic is shifted between paths. Additionally, ISL retargeting, maintenance, and debris avoidance maneuvers can temporarily remove nodes from the network. Altogether, such factors often translate into unpredictable latency fluctuations which manifest as jitter.

B. Jitter

Jitter is defined as the variation in packet inter-arrival times (end-to-end delay). For realtime applications like video call, video streams, telecommunication, and gaming, large jitter can massively degrade perceived quality, increasing artifacts and forcing large media buffers, even when the average latency is low. Thus, minimizing jitter while keeping mean latency near the shortest path baseline should be considered an important objective in any large-scale satellite communication network. Within the network, this manifests as an ISL-aware, topological based routing algorithm that reliably determines paths in the small decision window provided using a jitter-aware objective function. We plan to evaluate such a routing design in simulation against the current standard shortest-path baselines, comparing jitter, mean latency, route churn and throughput

C. Contributions

We propose jitter-minimizing routing algorithm for LEO satellite networks (JMR), a predictive routing protocol that creates more stable paths to reduce jitter. JMR is based on finding stable paths to a specified look-ahead horizon. To reduce the computational cost of doing so, certain satellites are designated as anchor nodes through which candidate paths are generated.

We implement JMR using simulations based on the Hypatia simulator framework [1]. According to the results, JMR experiences an average 13% improvement in jitter over location-based routing at the cost of a 15% increase in the mean latency.

The rest of the paper is organized as follows. We review existing literature in Section II, and Section III discusses the methodology used to implement and test JMR, including the simulator used, comparison to existing protocols, data collection, and satellite constellation used. Section IV discusses the design of the JMR protocol, discussing the predictive routing, jitter-aware path selection, anchors, and hysteresis. Experimental results are presented and analyzed in Section V. Potential directions for future work are presented in Section VI, and Section VII provides a summary of the paper and the lessons learned therein.

II. Related Work

Satellite networks have been a subject of study for decade, but the recent development and continual deployment of LEO satellite mega constellations has caused a recent spike in work. Routing is a crucial area of that research, as many traditional routing methods do not function well in satellite networks.

A. General Prioritization of Latency

Research on satellites has been ongoing for decades, with many advancements and the start of large-scale commercial

deployments in that time. Many attempts have been made to improve network latency, whether by dividing satellites into local regions [2], applying heuristic path-finding algorithms [3], or a myriad of other methods. However, relatively little has been done to minimize jitter through these routing methods, with some proposed local routing algorithms that the routes generated may be non-optimal and redirected several times based on the immediate state of nearby satellites [4]. Such redirections can have significant impacts on jitter by potentially creating a completely different, non-optimal route that takes longer. As mentioned previously, even in networks with a low average latency, individual spikes can create large jitter that renders realtime applications unusable. Consequently, the main body of work fails to meet certain quality of service (QoS) requirements.

One direction of study that often indirectly addresses jitter is that of congestion control algorithms, which adjusts sender behavior based on network congestion and link overutilization. Of these, the recently proposed LeoCC [5] is a transport-layer congestion control algorithm designed for the rapidly-changing nature of LEO satellite networks. It leverages the predictability of scheduled network reconfigurations to look ahead and detect bottlenecks and adjust the sending rate accordingly, resulting in steadier queue and packet reception rates. One measure in the evaluation of LeoCC was the observed range of RTTs (Round trip times - a proxy for jitter) which was found to be significantly lower than traditional congestion control algorithms used in terrestrial networks. However, LeoCC does not conduct routing selection or interact with the network layer. Its primary aim is to increase the overall throughput of a network, which can complement, but not necessarily replace, jitter-aware path selection at the routing layer. Nonetheless, its insights on predictable reconfigurations and the identification of future bottlenecks can be applied at the routing level to selectively route paths to avoid congestion and decrease jitter.

B. Focus on Jitter

Some recent work has directly attempted to address jitter and related variations in network performance. PP-IBM [6] utilize a priority mechanism that accounts for jitter in conjunction with the Bhandari algorithm for finding the best path between two satellites. However, this method only considered LEO ISLs with no other network elements and assumed topology changes occurred at regular intervals, which is often not the case in practice. They handle time by performing periodic recomputations, not by looking ahead and optimizing how paths will evolve over time. As another approach, LMSR [7] uses software-defined networking with ground-based controllers to precompute the network topology for multiple timeframes. Based on the topology, the shortest path through uncongested nodes is found for each time interval, and the most congested links are iteratively rerouted so the number of hops between source and destination satellites in all intervals is as close as possible to the one with the most hops in its shortest path. This method is effective in reducing jitter and has a beneficial effect on overall link utilization, but the centralized computation of routes for all services for entire constellations over multiple time slots simultaneously is likely unscalable as proposed.

III. Methodology

In this section, the methods used to implement and test JMR are presented. Most importantly, the simulator framework used for experimentation is specified.

A. Hypatia

In order to conduct experiments, a satellite network simulator was needed. After some investigation, the Hypatia simulator [1] was selected as a comprehensive and open-source simulation framework that could meet all anticipated needs. The framework is primarily built as an extension for the popular packet-level network simulator, NS-3. Specifically, the Hypatia simulator implements a library that adds support for satellite networking with ISLs and GSLs. Additionally, the framework comes with some built-in visualization and data gathering tools. Specifically, scripts to use both the command-line graphing utility, gnuplot, and JavaScript 3D mapping library, Cesium, are included in the framework. The framework also comes with two-line elements (TLE) for some real satellite networks for testing deployed satellite networks. In the paper where Hypatia was developed and tested [1], it was found that the simulator was scalable with constellation sizes and simulation interval, allowing testing with larger constellations or more precise simulation as necessary.

In order to implement JMR in Hypatia, it was necessary only to modify the route generation function of the simulator. Specifically, the simulator is implemented using predictive routing, so all possible forwarding tables are pre-generated before the simulation and updated in NS-3 as the simulation progresses. As such, the network state at every timeframe is computed then forwarding states are generated based on the network graph. In particular, the Floyd-Warshall algorithm is used to find the shortest path length between every pair of nodes. In order to implement a look-ahead function, the graph generation algorithm was slightly modified to allow calling repeatedly and storing the results for the entire window. Through the use of a sliding window array of graphs, only the graph for the next timeframe needs to be generated each step after the initial window is computed. Then, the actual routing decisions could be modified by changing the function used to find the neighbor node should forward to in order to reach a specific target. By default, the function finds which neighbor is the closest to the destination and routes to that, but modifications were made to select the neighbor that produces the least jitter by having the most consistent length across the entire look-ahead window. To ensure Hypatia functions as intended, a comprehensive test was run and validated with the unmodified simulator and original routing methods.

B. Comparison to Others

For comparison with other similar routing mechanisms it was determined to use the LMSR [7] and PP-IBM [6]

algorithms. The implementation of LMSR was able to utilize the same look-ahead function used for JMR itself, as both algorithms have similar approaches around the use of future states. However, the use of predictive routing meant that congestion was not able to be utilized as a factor in routing. As such, the algorithm's implementation is not able to be a perfect replica of the original. Rather, it had to be assumed that links had equivalent congestion, so the removal of links was done arbitrarily prioritizing the immediate link. The PP-IBM algorithm [6] could be implemented using the anchor mechanism developed for JMR, but this was not able to be done for a lack of time to implement a priority mechanism and run additional simulations.

C. Data Collection

Data collection was relatively simple, as Hypatia [1] already has some data collection using NS-3 functions. Notably, the RTTs are measured and graphed by the framework for every data flow in the simulation. However, in order to get a more holistic view of the network performance, a function to collect and compile the RTTs as a combined average was implemented. Some other metrics were also collected, such as the data rates, progress, and congestion window of each flow.

D. Constellation Choice

For evaluation, the Kuiper constellation with 630 nodes included with Hypatia [1] was selected, but it was believed that other similar constellations, such as the included Starlink constellation, would yield similar results. The Kuiper constellation was chosen due to its relatively large size and coverage as well as its support of ISLs. Additionally, the Kuiper constellation was used elsewhere in the Hypatia paper [1] and framework tests, making for easier regression testing and comparisons.

IV. Design Rationale

In this section, we present the JMR routing algorithm that seeks to minimize jitter while remaining near the optimal mean latency. JMR takes inspiration from both PP-IBM [6] and LMSR [7] utilizing four key concepts: Predictive routing, key anchor nodes, path-switch hysteresis, and a scoring heuristic that prioritizes paths with minimal jitter.

A. Predictive Routing

A common pitfall of many algorithms that seek to minimize jitter is the consideration of only the current-time snapshot. Jitter, however, is a time-varying quantity that measures the variance in delay between packets at different time steps. Because of this, any routing algorithm seeking to minimize jitter should consider not just the current time snapshot but also how routes and their corresponding delays may evolve over time.

Large LEO satellite networks are highly predictable with regard to geometric shifts in the topology, such shifts induce predictable variance within slant distances and shifting node adjacencies. We can harness this inherent predictability to perform a time-forward look ahead of the network over future time steps. In practice, this translates to a duplication of the network graph over many time slots, adjusting node adjacencies and edge costs depending on the topology of the network at that time. Similarly to LMSR [7] we partition a horizon interval $H = [0, t_n]$ into discrete time slices (epochs) of length Δt. For the start of each epoch, we create a graph representing the current state of the network. Each node corresponds to a satellite in the ISL network, while the edge costs between them are computed by calculating the sum of the propagation and queuing delays for the ISL communication:

$$c_{ij} = prop_{ij} + queue_j \tag{1}$$

The propagation delay is the time taken to send a packet from a satellite, i, to its ISL neighbor, j:

$$prop_{ij} = \frac{s_{ij}}{c} \tag{2}$$

where s_{ij} is the distance between satellites i and j, and c is the speed of light.

The queuing delay is a simple model of the congestion at satellite j based on the number of connections through j at t_0 (See 3.B):

$$queue_j = w_d \mu \tag{3}$$

where μ is the number of anticipated connections through j at t_0.

B. Jitter-aware Path Selection

Our algorithm will determine candidate paths by examining how paths selected at the present time, t_0, evolve throughout the time horizon. For each connection within the network, we calculate a set, S, of k shortest path candidates. We then consider how the cost of those paths evolve over the horizon by constructing a delay sequence, D_p for each candidate, P:

$$D_p = \{d_0, d_1, ..., d_n\} \tag{4}$$

Our algorithm will score the path based on key metrics of this latency sequence including mean latency, standard deviation, and maximum step size. Each parameter will be weighted and summed to determine the fitness of the candidate path:

$$Score(P) = w_\mu \mu_{D_p} + w_\sigma \sigma_{D_p} + w_{\delta_{max}} \delta_{max_{D_p}} \tag{5}$$

Where μ_{D_p} is the mean latency of the delay sequence, σ_{D_p} is the standard deviation, and $\delta_{max_{D_p}}$ is the max step change, with corresponding weights w_μ, w_σ, $w_{\delta_{max}}$.

While we seek to prioritize jitter minimization, we do not want that to come at the cost of mean latency. For this reason, we will be implementing a latency guardrail to ensure the chosen path remains within a set range of the optimal mean latency. We define a parameter α (0.1-0.2) that ensures candidate paths remain close to optimal latency by selecting the minimum mean latency within the candidate set, S, and ensuring all other candidates adhere to:

$$P \leq min_\sigma(S) \times (1 + \alpha) \tag{6}$$

C. Anchors and Tunnels

The jitter-aware scoring system as proposed above is prone to massive computational overhead. For F connections within the network k candidate paths are generated, resulting in a worst-case time complexity of $O(Fk * E\log(V))$ which is not practical for any large-scale constellation. To minimize computational complexity, we adapt from PP-IBM [6] the notion of anchors and tunnels. By strategically placing a set of key anchor nodes, A, throughout the constellation, ensuring there are a few anchors within each orbital plane, we can reduce computation by only performing path searches between the anchors. At each timestep, we run a shortest path search from every anchor to compute a matrix M where M_{ij} is the cost to travel from anchor i to anchor j. We then route all candidate flows through these anchors, moving from the source node to the nearest access anchor, than performing a lookup of the anchor matrix to generate candidate routes to the nearest egress anchor to the destination, before finally routing to the destination node.

This massively reduces the computational complexity, as for each epoch we only need to perform A global path searches with constant time hash-table look ups to determine the access, egress, and candidate paths. Overall this results in a much improved complexity of $O(A * E\log(V) + F * k)$. An additional benefit of routing paths through anchors is that the nearest access and egress anchors will allow for greater ground area coverage than individual nodes, reducing the number of required handovers and path switches.

D. Path-switch Hysteresis

Implemented as proposed, our algorithm is susceptible to constant switches between paths as the network topology changes. At each epoch small shifts in the network across the horizon may cost the optimal path to oscillate between many candidates. Path-switch hysteresis will introduce inertia into the path selection algorithm, only switching to a new path if it is significantly and persistently better than the current path.

We plan to implement path-switch hysteresis in a few ways, the first being a score margin. We will only switch paths if the new path improves upon the total score by more than a threshold ΔS, which is a small margin of the mean score over all candidate paths. Additonally, we also plan to explore the concept of folding the path-switch hysteresis directly into the scoring heuristic by making path switches self penalizing. The jitter scoring is modified to account for this:

$$Score'(P) = Score(P) + w_s \cdot \mathbb{I}(P \neq P_{prev}) \tag{7}$$

where w_s is a switching penalty weight, and $\mathbb{I}(P \neq P_{prev})$ is an indicator function that equals 1 if the candidate path differs from the previously selected path, and 0 otherwise.

V. Results

A. Simulation Setup

The simulator was configured to use the Amazon Kuiper constellation with 21 orbital planes and 30 satellites per plane, for a total of 630 satellites orbiting at an altitude of 630 km and an inclination of 51.9°. The first simulation was run with a total of 6 flows (Manila to London, Manila to Los Angeles, Sydney to Nairobi, Sydney to Sao Paulo, and Tokyo to New York). The other simulations were all ran with 100 ground stations were selected based on estimates of the 100 most populous cities as of 2025. All flows were based on TCP New Reno at 10 Mbps throughout 100 seconds. JMR was run using 1/10 nodes as anchors. All random aspects were seeded to ensure the flows represented the same connections in every experiment and could be directly compared between routing algorithms.

B. Tokyo to New York

To indicate the viability of JMR as a routing method, a relatively small simulation was ran with a small number of flows, as described above. The RTTs of the flow between New York and Tokyo is presented for LMSR and JMR in Fig. 1 and 2, respectively. From the figures, it is apparent that JMR did induce a higher overall RTT, with an increase of about 15% over LMSR. While not ideal, an increase to mean latency was anticipated in order to achieve the jitter reduction JMR aims for. While the increase is more significant than had been initially hoped for, the increase to mean latency is not enough to render connections unusable in most cases.

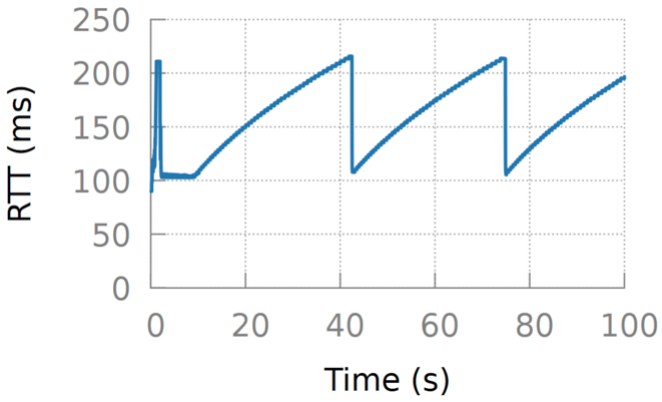

Fig. 1. Tokyo to New York TCP flow using LMSR

Besides the increase to mean latency, it was also noted that JMR achieved an essentially flat RTT between approximately 5 and 30 seconds. Such a flat RTT represents a minimal jitter near 0, which is the optimal result of JMR. As such, the flow displays promising results for the viability of JMR. However, after 30 seconds, the same sawtooth pattern is observed as in LMSR, but with a slightly slower climb. This pattern is most likely an effect of routing staying with a single satellite until another comes into range. Ideally, JMR would switch to paths preemptively to minimize the sudden drop off of RTTs, which appears to happen in the first 30 seconds. It was not clear why after that period, the sawtooth pattern appears. However, it was proposed that the partial deployment of the Kuiper constellation used meant that there was not always a

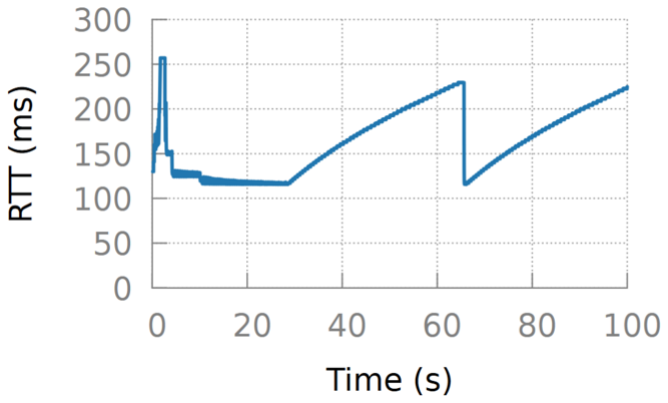

Fig. 2. Tokyo to New York TCP flow using JMR

better satellite to route to. As a final note, limitations in the simulation caused this experiment and all others to have an initial spike in latency within the first few seconds, but this disappears within approximately the first 5 seconds.

C. Runtime Analysis

As discussed in Section IV, the computational complexity is the major downside of using look-ahead algorithms, since path-finding algorithms need to be run for the complete network graph at every timestep within the window. The use of anchor points is intended to reduce this complexity without significantly increasing jitter. According to the benchmark results displayed in Table 1, it was found that JMR only took 97 seconds to run compared to the 143 seconds for LMSR. As JMR took approximately 2/3 the time of LMSR, it was clear that the anchor-based approach significantly reduced the complexity of routing. Such a reduction in time equates to a roughly 50% speedup for JMR over LMSR. Furthermore, these results would be expected to be more pronounced as the proportion of anchors is decreased or the look-ahead horizon increased.

TABLE I
ROUTING GENERATION TIMING

Algorithm	Complexity	Time (s)
All-Pair Shortest Path	$O(V^3)$	85
LMSR	$O(TV^3)$	143
JMR	$O(A(V \log V^3 + 3))$	97

D. Consistent Flow

Moving to the large scale experimentation with 100 flows described at the start of the section, it was found that some flows were extremely consistent. Fig. 3 and 4 show such a consistent link in LMSR and JMR respectively, but the graphs appear identical. This result was believed to be a side-effect of using LMSR as the basis for the look-ahead function of JMR, which is the primary route determination method. As such, JMR is intended to approach the performance of LMSR, with the benefit of a reduced runtime from the use of anchors. While it was not possible to verify the exact nodes used as anchors and for routing the graphed flow, it was believed that such consistent results were likely the result of the routing between anchors matching the routing otherwise. Notably, JMR has the same mean latency as LMSR in this case, despite the faster routing.

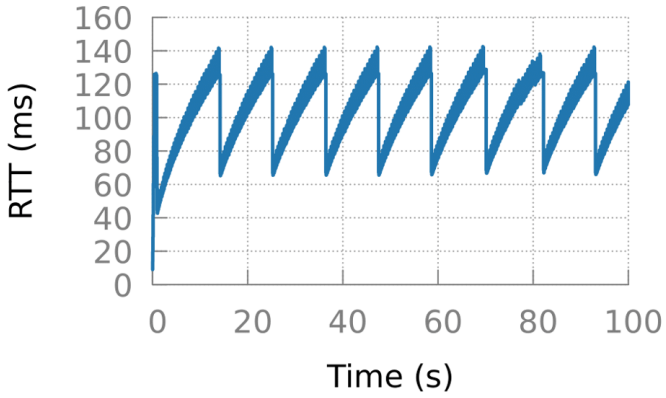

Fig. 3. Consistent flow using LMSR

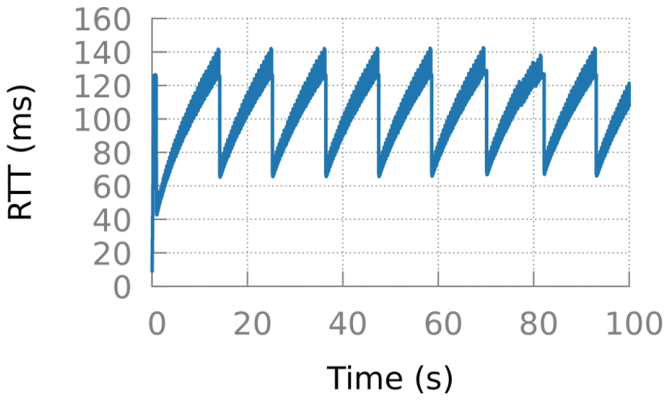

Fig. 4. Consistent flow using JMR

E. Large-scale Experiment

In cases where the flows are not identical between LMSR and JMR, it was found that JMR reduced the jitter compared to LMSR. Fig. 5 and 6 show such a case for LMSR and JMR respectively. Most notably, JMR experiences a prolonged period of completely stable RTTs between approximately 30 and 62 seconds. Since jitter is the variance in RTT, this represents a period of approximately 0 jitter. As the minimization of jitter is the goal of JMR, such an outcome is ideal for the algorithm. However, most flows are not able to achieve such an ideal, as can be seen outside that flat period. This is primarily due to the rapidly-changing topology of LEO satellite networks, which prevent stable connections over long periods of time.

Outside the zero-jitter period, it was observed that there was less fluctuation in RTT for JMR than LMSR. While not

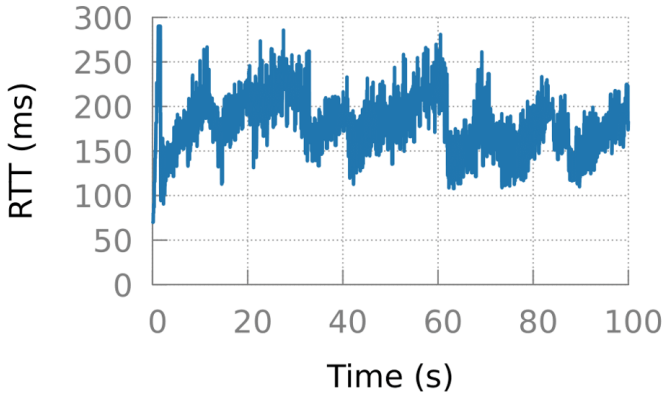

Fig. 5. Regular TCP flow using LMSR

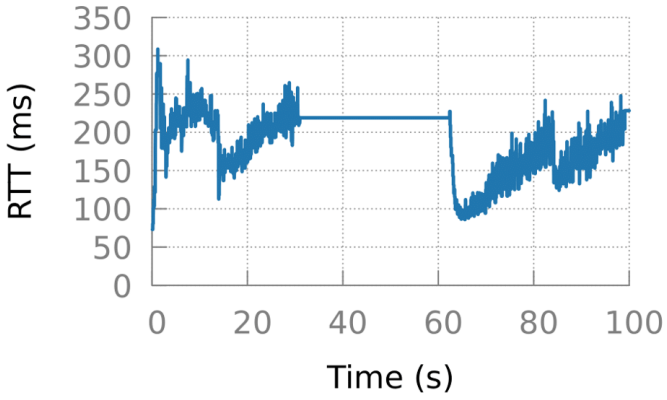

Fig. 6. Regular TCP flow using LMSR

immediately apparent from the figures, the thickness of Fig. 5 compared to Fig. 6 is caused by the smaller fluctuations of RTT in JMR. Corresponding to less variance in the RTTs, less fluctuation is an improvement in jitter. Specifically, JMR achieved an average of approximately 13% jitter reduction over LMSR. Much of the jitter that was observed was believed to be caused by congestion, which itself was likely a side-effect of the anchor-based routing. The predictive routing used for Hypatia was unable to use congestion in routing decisions, so the routes produced could have over utilized certain paths between anchors. A reactive routing approach, as seen in the original LMSR implementation [7], may have been able to address this issue.

F. Loss

It was found that JMR produced a slightly higher number of failed flows than the other routing methods. In particular, Hypatia's default routing method would consistently have data transferred in every flow, with few exceptions, whereas approximately 1/10 to 1/5 flows would have 0 or nearly 0 data transferred throughout the entire simulation. Additionally, several flows in JMR would only have a window of successful communication. This outcome is almost certainly the result of JMR being unable to identify a steady path throughout the look-ahead window. It was intended that JMR would fall back to the best available path in such a case, but the handling was likely misimplemented, causing some inconsistent paths to fail. However, it is also possible that at least some failures were from the Kuiper constellation snapshot used not having adequate coverage to reliably connect the cities used as the endpoints of the failed flows, as the cities were randomly selected and paired from the 100 most populous cities. Additional testing with a focus on the coverage of the constellation and specific flow endpoints could reveal if that was the source of the issue.

VI. Future Work

As mentioned previously, it had been planned to implement PP-IBM [6] as another point of comparison. However, there was insufficient time to do so with appropriate care to faithfully recreate the original algorithm, so the secondary algorithm for comparison had to be scrapped. However, using the anchor-based routing from JMR should be the majority of the implementation, so comparison in the future should be relatively straightforward. The main challenge would likely be identifying the optimal proportion and placement of anchor points, which was not well addressed in the original paper and would be an interesting point of study that could also be applied to JMR.

Otherwise, issues encountered during the implementation of JMR meant that little time was able to be spent optimizing the look-ahead horizon and hysteresis threshold parameters. Consequently, running more simulations with a variety of parameter choices, constellation sizes, and simulation timestep size could produce valuable insights. Running simulations was a fairly resource and time consuming process, but is largely automated.

Future projects could also consider implementing JMR in a congestion-aware reactive routing scenario for more real-time routing with additional information. Knowledge about congestion could provide a better comparison to LMSR [7], but it may also be a valuable component of path selection in JMR to improve performance. Another approach could be the integration of ground-based routes into routing considerations. Grounded links are generally much more reliable and stable, minimizing jitter compared to ISLs or even GSLs.

Lastly, one of the biggest limitations with look-ahead routing was the inability to parallelize the process. In the future, effort could be put into finding an alternative implementation or other method altogether that is able to be parallelized for much faster computation. Alternatively, reactive routing methods would not run into parallelization issues, so a reactive version, in addition to the previously mentioned congestion avoidance, may also be more performant.

VII. Summary and Lessons Learned

A. Summary

This paper presented the jitter-minimizing routing algorithm for LEO satellite networks (JMR). JMR is a routing algorithm

aimed at minimizing jitter in the rapidly-changing topology of LEO satellite networks, which has not been a well-studied area of research. To create JMR, the anchor-based routing of PP-IBM [6] and the look-ahead routing of LMSR [7] were combined. In addition, JMR adds hysteresis to ensure consistent pathing with minimal jitter by reducing the amount of flipping between routes. As a result, JMR was anticipated to have minimal jitter compared to other satellite routing algorithms while also being more performant than other look-ahead routing methods as a result of the simplification of anchor-based routing.

In order to test JMR, the Hypatia simulator framework [1] was used. The simulator adds satellite routing to the popular NS-3 network simulator, and it also provides TLE and route generators as well as graphing utilities. In particular, JMR was implemented as a new routing algorithm that could be plugged into the framework's routing generator. As such, the network simulation and data collection aspects of the framework required minimal modification.

By running multiple simulations using JMR, LMSR, and Hypatia's default routing method, it was found that JMR achieves an average of approximately 13% less latency than LMSR with about 15% higher latency. However, better results could likely be achieved by further refining JMR and optimizing its parameters. In ideal cases, JMR was able to achieve near 0 jitter. In others, the routes generated by JMR matched those of LMSR almost perfectly, but with less time required for routing. However, JMR experienced a alightly greater number of failed flows compared to LMSR. Overall, the performance of JMR generally surpassed that of LMSR when jitter is the primary concern.

B. Lessons Learned

The main lesson learned about the subject itself is that routing is more complicated than it may first appear. While it may seem like the best route is the one that gets packets to the destination the fastest, the initial research and project formulation revealed how that is not always the case. Other factors, like jitter, computational complexity, or energy efficiency may be more important in some scenarios. Furthermore, when satellites specifically are considered, routing becomes much more complicated. Most traditional routing methods simply do not scale and adjust effectively to the complexity of LEO satellite networks, particularly due to the large number of low-resource nodes with the rapidly changing topology of such networks.

Figuring out a simulator was also a major challenge. It was difficult to find one that was useful, but it was critical to testing satellite routing algorithms. A few otherwise unmentioned simulators were evaluated, with some even being deployed and tested, before settling on Hypatia [1]. Even with a viable simulator, issues and limitations were encountered that required compromises and significant reworks. While it would have been interesting to try making a simulator from scratch, satellite networking is complex enough that creating a simulator likely would have taken the entire duration of the project. One other lesson from the simulator was how to effectively share a virtual machine between the group for collaborative development and adequate resources to run simulations.

In less technical areas, it quickly became apparent that finding an area with sufficient work to build off but gaps to be filled could be challenging. Satellite network jitter proved to be a good area for finding gaps, but the existing body of knowledge was generally lacking in terms of repeatable applications and methods to build on. At best, some pseudocode was found for LMSR [7], which was enough to guide our own implementation but that was still significantly more complicated and less accurate than using the original version outright, had it been available. This likely made the JMR implementation much more difficult as well, at least compared to other algorithms with a better documented basis. Regardless, the documentation search was a major time sink that could likely have benefitted from more time.

Individual Contributions

A. John Nowinski's Tasks

- Literature search
- Introduction and Methodology sections for proposal report
- Proposal report formatting
- Methodology section for proposal presentation
- Anchor-based routing implementation
- Initial JMR implementation
- PP-IBM [6] anchor point and candidate path selection
- Algorithm refinement and bug fixing
- Methodology and results sections for progress presentation
- Design Rationale section for final report
- Final report formatting
- Code reproduction preparation

B. Jude Hansen's Tasks

- Literature search
- Previous Work, Plan, and References sections for proposal report
- Proposal report formatting
- Satellite Overview, Jitter Overview, and Plan sections for proposal presentation
- Hypatia simulator [1] setup and hosting
- Graph look-ahead implementation
- Initial LMSR [7] implementation
- Algorithm refinement and bug fixing
- Motivation and previous work sections for progress presentation
- Abstract, Introduction, Related Work, Methodology, Future Work, Technical Results and Analysis, Summary and Lessons Learned, Individual Contributions, and References sections for final report
- Final report formatting

ACKNOWLEDGMENT

We thank our professor, Dr. Li Xiao, for her support on this project and for the CSE 824 course in general. We thank Griffin Klevering for the inspiration to do a project on satellite networking.

REFERENCES

[1] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the "internet from space" with hypatia," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 214–229. [Online]. Available: https://doi.org/10.1145/3419394.3423635

[2] X. Zhang, Y. Yang, M. Xu, and J. Luo, "Aser: Scalable distributed routing protocol for leo satellite networks," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 65–72.

[3] X. Sun, Y. Shen, and H. Zhang, "A stable satellite routing algorithm based on a-star algorithm in dynamic networks," *Journal of Physics: Conference Series*, vol. 2807, p. 012026, 07 2024.

[4] Q. Shan, Z. Wang, S. Zhang, Q. Meng, and H. Luo, "Routing in leo satellite networks: How many link-state updates do we need?" in *2023 IEEE International Conference on Satellite Computing (Satellite)*, 2023, pp. 7–12.

[5] Z. Lai *et al.*, "Leocc: Making internet congestion control robust to leo satellite dynamics," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 129–146. [Online]. Available: https://doi.org/10.1145/3718958.3750491

[6] C. Tong *et al.*, "A multipath satellite routing to enhance networking performance for leo constellations," in *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2024, pp. 2248–2249.

[7] S. Sun, R. Zhang, K. Liu, Z. Sun, Q. Tang, and T. Huang, "Lmsr: A low-jitter multiple slots routing algorithm in leo satellite networks," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.