RedHAT: Receiver-driven Handover Adaptation in TCP for Starlink Communication

Jaehyeong Lim, Minjun Choi, Mingyu Park, and Jeongyeup Paek

Department of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea
a391040065@cau.ac.kr, alswns2631@cau.ac.kr, hello0922@cau.ac.kr, jpaek@cau.ac.kr

***Abstract*—Low Earth Orbit (LEO) satellite networks such as Starlink have shown promising potential to provide high-speed low-latency Internet service all over the world. However, rapid and continuous movement of LEO satellites inevitably leads to frequent handovers which can cause service interruptions and degrade user experience. In particular, TCP's congestion control mechanisms may misinterpret handover-induced losses and delays as congestion signals, leading to unnecessary reductions in the sending rate and underutilization of the available network capacity. Although recent studies have proposed mechanisms to mitigate the problem, their solutions are focused on sender-side which is not always possible to modify. In this paper, we propose *RedHAT* (*receiver-driven handover adaptive TCP*), a receiver-side solution that uses only the receive window (*rwnd*) to mitigate the handover effect and can be easily deployed without requiring any changes on the sender-side. Through precise handover timing estimation and dynamic *rwnd* adjustment, *RedHAT* allows the receiver to effectively adapt to handover events, improving performance and user experience in LEO satellite networks. Evaluation results show that *RedHAT* improves the average throughput by 10.29% for CUBIC and 20.75% for BBR, while reducing the 99^{th} percentile RTT by 11.07% for CUBIC and 19.44% for BBR, demonstrating the effectiveness for both loss-based and delay-based TCP congestion control algorithms.**

***Index Terms*—TCP, Low-Earth-Orbit (LEO) Satellite Networks, Handover, Congestion Control, Flow Control**

I. Introduction

Low Earth Orbit (LEO) satellite networks, such as Starlink, Amazon LEO, and OneWeb[1], have achieved remarkable advances in recent years [1]. Compared to traditional geostationary satellite systems, LEO constellations operate at significantly lower orbital altitudes at approximately 550 km, enabling substantially higher throughput and lower latency. For example, Starlink [2] is capable of delivering downlink throughput of hundreds of Mbps and RTTs on the order of tens of milliseconds. As the constellation continues to expand, these performance characteristics are expected to improve further.

This research was supported by the MSIT(Ministry of Science, ICT), Korea, under the National Program for Excellence in SW), supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation) in 2026 (2025-0-00032), by IITP-ITRC (Information Technology Research Center) grant funded by the Korea government (MSIT) (IITP-2026-RS-2022-00156353, 33%), and also by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00359450). J. Paek is the corresponding author.

[1]https://starlink.com, https://leo.amazon.com/, https://www.eutelsat.com/

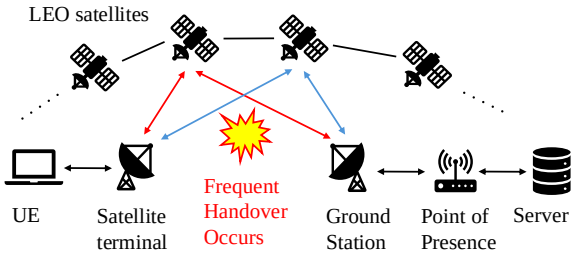


Fig. 1: LEO satellite network architecture and handover.

Despite these advances, LEO satellite networks face unique challenges arising from the high orbital velocity of satellites and the resulting frequent handover events. Fig. 1 shows the handover process in LEO satellite networks. TCP congestion control reacts to handover-induced losses, delay spikes, and temporary service interruptions in a manner similar to congestion signals. It leads to unnecessary reductions in the sending rate and underutilization of the available network capacity.

Prior studies have primarily addressed this issue through modifications to the TCP's congestion control algorithm (CCA) at the sender. For example, Satpipe [3] identifies the approximately 15-second handover periodicity and proactively reduces the congestion window (cwnd) to zero during handover periods, thereby preventing excessive queue buildup. LeoCC [4] detects handover events by monitoring the ping interval to the Starlink point-of-presence (PoP) and dynamically adjusts the sending rate in response to detected handovers. While these approaches demonstrate promising improvements, they require modifications at the sender side. Deploying sender-side transport modifications across the Internet is notoriously challenging. For example, Explicit Congestion Notification (ECN), although standardized more than two decades ago, required many years to achieve limited deployment due to incremental deployment issues and compatibility concerns [5]. Moreover, Internet middleboxes such as NATs and firewalls are known to drop or rewrite TCP options, making sender-side signaling mechanisms difficult to deploy reliably in practice [6]. In contrast, receiver-side solutions can be deployed unilaterally without requiring any modifications to the sender, making them more practical for real-world deployment.

Outside the satellite networking domain, M2HO [7] ad-

dresses a similar problem in 5G networks where handover events can be misinterpreted as congestion signals by TCP. This work proposes a receiver-side solution for 5G that leverages the *rwnd* field in TCP headers to control the sender's transmission rate during handover events. However, the handover and latency characteristics in 5G cellular are different from LEO satellite systems [8], [9].

In this paper, we present *RedHAT*, a receiver-driven solution that mitigates the impact of handover events in LEO satellite networks without requiring modifications to the sender. By shifting handover-aware control mechanisms to the receiver side, our approach enables effective adaptation to periodic handover events while making it deloyable in practical Starlink deployments. Our solution, *RedHAT*, significantly improves the average throughput, by 10.29% for CUBIC and 20.75% for BBR, and reduces the average RTT by 5.22% for CUBIC and 1.63% for BBR, while reducing the 99^{th} percentile RTT by 11.07% for CUBIC and 19.44% for BBR. These results demonstrate that *RedHAT* is effective for both loss-based and delay-based TCP CCA.

II. Measurement of Starlink Handover

Since *RedHAT* is a receiver-side solution, it can influence the sender's transmission rate only through the receiver's feedback such as TCP *rwnd* control and ACK pacing. Unlike prior sender-side approaches, *RedHAT* requires accurate knowledge of handover timing to proactively adjust the receiver's advertised receive window. To this end, we first collect the handover events of Starlink through real-world experiments, and analyze whether the pattern of handovers is predictable.

LeoCC [4] showed that Starlink handovers can be identified through anomalies in the point-of-presence ping response interval. When a handover occurs, the ping interv al increases significantly showing a clear spike in the response time. Furthermore, Satpipe [3] reported that, after NTP synchronization, Starlink handovers occur within narrow time windows around the 12^{th}, 27^{th}, 42^{nd}, and 57^{th} second of each minute, demonstrating that handover timing is not only periodic but also highly deterministic.

We adopted the same methodology in our experiments; we sent PoP ping requests at 10 ms intervals and monitored handover events for 10 minutes. Fig. 2 plots the part of response intervals of the PoP probes for 10 minutes. We observed that handovers occur with a strong periodicity of approximately 15 seconds, a same pattern reported in LeoCC, which can be exploited by *RedHAT* for proactive adaptation.

Fig. 3 shows the CDF of the gap between consecutive spikes in the PoP ping response interval of the entire 1-hour measurement. These results support the feasibility of proactive receiver-side adaptation based on predicted handover timing rather than reactive handover detection.

In addition, we observed discrepancies between the TCP packets captured at the sender and those received at the receiver, including different RTT and TTL values. Although the exact cause remains unclear, these observations indicate that sender-side TCP RTT measurements may not accurately reflect the end-to-end RTT. Thus, we measure the end-to-end RTT using a separate application-level ping request between the sender and receiver.

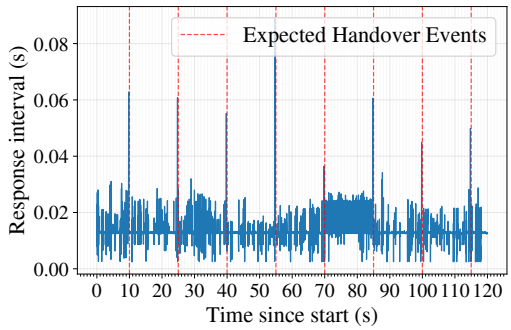


Fig. 2: Response intervals of probes sent to the Starlink PoP, showing 15 second periodic spikes corresponding to handover events.

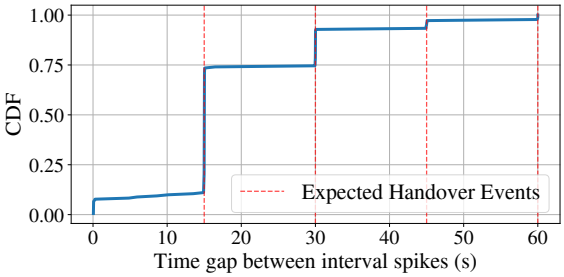


Fig. 3: CDF of the time gap between consecutive spikes in the PoP ping response interval, confirming the regular 15 second periodicity of handover events.

III. Design

In this section, we present the design of *RedHAT*, a receiver-driven approach to mitigate the impact of handover events in Starlink networks. Since the receiver cannot directly control the sender's congestion window, transmission rate adaptation must be achieved indirectly through TCP flow control.

A. Three-Phase Handover Model

By confirming the deterministic and periodic nature of handover events in Starlink networks, we propose a three-phase handover model that enable proactive receiver-side adaptation to handover events.

Fig. 4 illustrates how the three phases of the handover model works with: Normal, Pre-handover, and Outage. In Normal phase, the receiver operates under normal conditions without any handover-related adjustments. In Pre-handover phase, the receiver anticipates an upcoming handover event based on the predicted handover timing and proactively adjusts its *rwnd* to prepare for the handover, preventing excessive queue buildup and potential packet loss during the handover. Additionally, the receiver's ACK needs at least half of the round-trip time (RTT) to reach the sender, so the receiver should initiate each phase RTT/2 faster than the actual handover timing to ensure that the sender receives the *rwnd* adjustment in time. In Outage phase, the receiver maintains

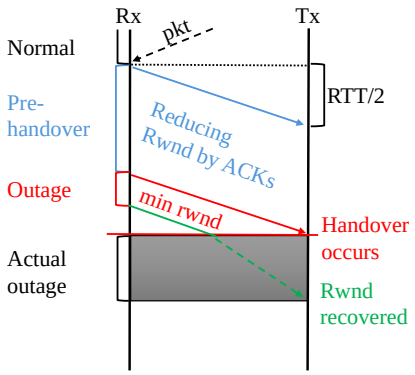


Fig. 4: Action of the three-phase handover model for proactive receiver-side adaptation.

a reduced *rwnd* to prevent unnecessary congestion control reactions from the sender. After the OUTAGE phase is completed, the receiver transitions back to NORMAL phase and immediately recompute the *rwnd* to allow the sender to fully utilize the available network capacity.

B. Receive Window (rwnd) Control Mechanism

Based on RFC 793 [10] and RFC 9840 [11], which describes the TCP *rwnd* control mechanism, *rwnd* reduction should be performed gradually to avoid sender-side abrupt window shrinking. It means that the advertised *rwnd* must not be reduced faster than the amount of data already received. As a result, *rwnd* reduction cannot be performed instantaneously and must instead be applied gradually over time. During the PRE-HANDOVER phase, the receiver progressively decreases the advertised *rwnd* from its current value to a predetermined minimum value, which is set to 4 Maximum Segment Size (MSS). This gradual reduction allows the sender to drain inflight packets before the handover occurs, while remaining compliant with TCP flow control semantics.

C. Parameter Selection

The duration of PRE-HANDOVER phase must be sufficiently long to acheive the desired *rwnd* reduction before the handover occurs. Let $rwnd_{\text{current}}$ be the current advertised *rwnd*, $rwnd_{\text{min}}$ be the minimum advertised *rwnd*, and R be the receiver's data receiving rate. The minimum duration of PRE-HANDOVER phase, T_{pre}, can be calculated as follows:

$$T_{\text{pre}} = \frac{rwnd_{\text{current}} - rwnd_{\text{min}}}{R} \tag{1}$$

This duration ensures that *rwnd* can be reduced solely through normal receiver-side buffer consumption.

Our *RedHAT* starts the PRE-HANDOVER phase RTT/2 before the expected handover timing to ensure that the sender receives the *rwnd* adjustment in time. But, actual RTT before handover may be different from the pre-computed RTT/2, which may lead to start sender-side OUTAGE phase too early or too late. For the former case, the duration of OUTAGE phase must be longer than RTT/2 - RTT_{min}/2 to ensure that the sender never get recovery ACKs before the handover occurs.

Fig. 5: Starlink terminal used for experiments.

For the latter case, the duration of OUTAGE phase must be short enough to prevent the sender from stalling due to the late recovery ACKs. So, we can choose the duration of OUTAGE phase, T_{outage}, as follows:

$$T_{\text{outage}} \geq \frac{RTT - RTT_{\text{min}}}{2} \tag{2}$$

If the OUTAGE phase ends before the handover is completed, it does not cause any problem because the sender will not receive any ACKs until the handover is completed.

D. Implementation

We implemented *RedHAT* as a modification to the Linux 6.8.0 kernel by modifying the tcp_output.c file to adjust the advertised *rwnd* according to the three-phase handover model. In our current experimental environment, the average *rwnd* is around 4 MB, the average receiving rate is around 220 Mbps. the average RTT is around 74 ms, and RTT_{min} is around 50 ms Thus, T_{pre} is approximately 145 ms and T_{outage} is at least 12 ms.

IV. EVALUATION

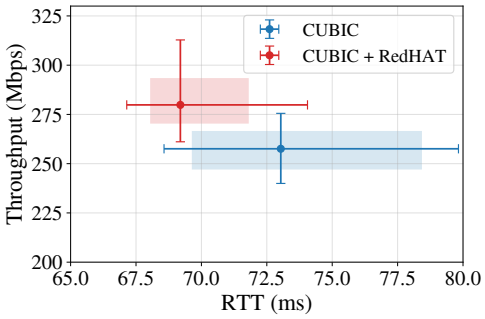
In this section, we evaluate the performance of *RedHAT* through real-world experiments in Starlink networks and compare it with serveral TCP congestion control algorithms.

A. Experimental Setup

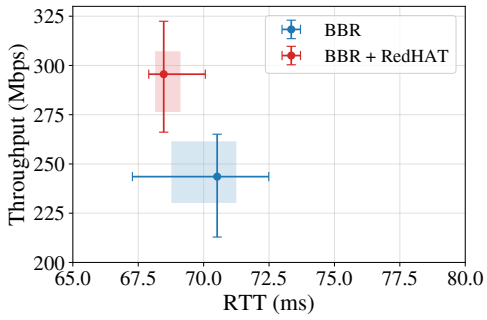
Experiments are conducted using a real Starlink network environment, where the client is connected to the satellite network via a residential Starlink standard (Gen 4) dish and a terminal, and the server is located in our university campus (Fig. 5). Throughput is measured using iperf3, and RTT is measured using application-level ping requests. Each experiment consists of 2 flows to saturate the full downlink capacity and each flow is measured for 3 minutes by repeating 8 times containing 96 handover events in total. The experiments are conducted under different TCP CCA, both loss-based and delay-based, exactly CUBIC and BBRv1, to evaluate the effectiveness of *RedHAT* across various congestion control mechanisms.

B. Performance Evaluation

Fig. 6(a) and Fig. 6(b) plot the performance comparison between original TCP CCA and using *RedHAT* in terms of average throughput and RTT, for CUBIC and BBR respectively. The inner box represents the 25th–75th percentiles, the whiskers indicate the 10th–90th percentiles, and the dot

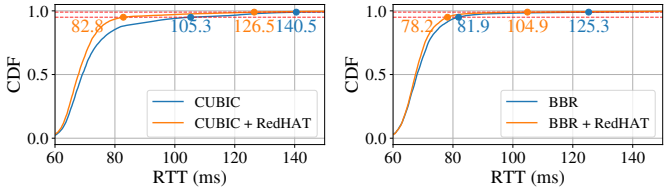


(a) CUBIC



(b) BBR

Fig. 6: Throughput and RTT with and without *RedHAT*. *RedHAT* improves both throughput (upper) and latency (left).



(a) CUBIC **(b)** BBR

Fig. 7: CDF of RTT with and without *RedHAT*. (95th and 99th percentile latency values annoated)

denotes the median. In terms of throughput, *RedHAT* improves the average throughput by 10.29% for CUBIC and 20.75% for BBR, demonstrating its effectiveness in enhancing data transfer. In terms of RTT, *RedHAT* reduces the average RTT by 5.22% for CUBIC and 1.63% for BBR, indicating its ability to mitigate the impact of handover events on latency. Furthermore, Fig. 7(a) and Fig. 7(b) plot the CDF of RTT for each CCA with and without *RedHAT*. In terms of tail latency, *RedHAT* significantly reduces the 99^{th} percentile RTT by 11.07% for CUBIC and 19.44% for BBR, highlighting its capability to improve the worst-case latency experienced by users during handover events.

C. Discussion

Both CUBIC and BBR show significant improvements in throughput and tail latency when using *RedHAT*. For CUBIC, which is a loss-based congestion control algorithm, the proactive adjustment of the receive window during handover events helps to reduce packet loss and prevent unnecessary queue buildup, leading to improved throughput and reduced tail latency. For BBR, which is a delay-based congestion control algorithm, the proactive adjustment of the receive window helps to prevent slowing down by queue buildup and maintain a more stable sending rate during handover events, leading to improved throughput and reduced tail latency.

V. Conclusion

In this paper, we have presented a receiver-side solution to mitigate the handover effect in LEO satellite networks using receive window adjustments. Experimental results demonstrate that the proposed approach can alleviate the adverse effects of handovers and improve communication performance in Starlink networks. *RedHAT* improves the average throughput by 10.29% for CUBIC and 20.75% for BBR, and reduces the average RTT by 5.22% for CUBIC and 1.63% for BBR, while reducing the 99^{th} percentile RTT by 11.07% for CUBIC and 19.44% for BBR. Future work includes extending the proposed mechanism to operate independently of the sender's congestion control algorithm (e.g. César [8] for 5G) and validating its effectiveness across a wide range of server and client environments.

References

[1] O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 70–109, 2020.

[2] S. Ma, Y. C. Chou, H. Zhao, L. Chen, X. Ma, and J. Liu, "Network Characteristics of LEO Satellite Constellations: A Starlink-Based Measurement from End Users," in *IEEE Conference on Computer Communications (INFOCOM)*, 2023, pp. 1–10.

[3] D. Zhao, X. Zhang, and M. Lee, "SatPipe: Deterministic TCP Adaptation for Highly Dynamic LEO Satellite Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2025, pp. 1–10.

[4] Z. Lai, Z. Li, Q. Wu, H. Li, J. Li, X. Xie, Y. Li, J. Liu, and J. Wu, "LeoCC: Making Internet Congestion Control Robust to LEO Satellite Dynamics," in *Proceedings of the ACM SIGCOMM*, 2025, pp. 129–146.

[5] B. Trammell, M. Kühlewind, D. Boppart, I. Learmonth, G. Fairhurst, and R. Scheffenegger, "Enabling Internet-Wide Deployment of Explicit Congestion Notification," in *Passive and Active Measurement*. Springer International Publishing, 2015, pp. 193–205.

[6] H. Lim, S. Kim, J. Sippe, J. Kim, G. White, C.-H. Lee, E. Wustrow, K. Lee, D. Grunwald, and S. Ha, "A Fresh Look at ECN Traversal in the Wild," *arXiv preprint arXiv:2208.14523*, 2022.

[7] Z. Liu, Q. Deng, Z. Tan, Z. Qian, X. Zhang, A. Swami, and S. V. Krishnamurthy, "M2HO: Mitigating the adverse effects of 5G handovers on TCP," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 1089–1103.

[8] J. Shin, G. Lee, J. Paek, and S. Bahk, "César: Cellular Resource Scheduling-Aware Congestion Control," in *Proceedings of The IEEE International Conference on Computer Communications (INFOCOM)*, May 2025.

[9] Y. Cho, M. Kim, S. Cho, and J. Paek, "Analysis of TCP Packet Transmission Characteristics in 5G Networks," in *The 15th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Oct 2024, pp. 448–451.

[10] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc793

[11] M. Bagnulo, A. Garcia-Martinez, G. Montenegro, and P. Balasubramanian, "rLEDBAT: Receiver-Driven Low Extra Delay Background Transport for TCP," RFC 9840, Sep. 2025. [Online]. Available: https://www.rfc-editor.org/info/rfc9840