SaTCP: Link-Layer Informed TCP Adaptation for Highly Dynamic LEO Satellite Networks

Xuyang Cao and Xinyu Zhang
University of California San Diego
Emails: {xuc054, xyzhang}@eng.ucsd.edu

***Abstract*—Low-Earth-orbit (LEO) satellite networking is a promising way of providing low-latency and high-throughput global Internet access. Unlike the static terrestrial network infrastructure, LEO satellites constantly revolve around the Earth and thus bring instability to their networks. Understanding the dynamics and properties of a LEO satellite network and developing mechanisms to address the dynamics become crucial. In this work, we first introduce a high-fidelity and highly configurable real-time emulator called LeoEM to capture detailed dynamics of LEO satellite networks. We then present SaTCP, a cross-layer solution that enables TCP to avoid overly conservative congestion control and improve its performance under high LEO link dynamics. As an upgrade to CUBIC TCP, SaTCP forecasts the time of disruptive events (*i.e.*, satellite handovers or route updates) by tactfully utilizing the predictability of satellite locations, taking into account the prediction inaccuracy, and informs TCP to adapt its decision accordingly. Experiments across various scenarios show SaTCP increases the goodput by multi-folds compared with state-of-the-art protocols while preserving fairness.**

I. Introduction

LEO satellite networks (satnets) hold the potential to fill in the last coverage holes of the broadband Internet. Their advantages are decisive: a highly available, high-throughput, and low-latency network that can largely cater to various delay-sensitive services and demands of underrepresented regions. Industry giants like SpaceX [1], Amazon [2], and OneWeb [3] have been deploying their LEO satnet infrastructures. Among them, Starlink by SpaceX is particularly notable, as it has already provided preliminary yet reliable service for many rural areas lacking cable access.

One crucial and intrinsic challenge for LEO satnets is the dynamics due to satellite mobility. The satellites need to revolve around the earth at an extremely high speed (around 7.5km/s [4]) to combat the gravitational force and stay in their orbits. Such fast movements, along with the corresponding handover and topology dynamics, can cause various types of instabilities on different layers of the network stack. More specifically, the transport layer will be highly vulnerable to such dynamics. TCP in particular, is unaware of satellite handovers, and will misinterpret the resulting surge of RTT and packet loss rate as a sign of congestion. The TCP sender will in turn cut its transmission rate aggressively or even reset it to the minimum level, leading to an underutilization of the network capacity as shown in Fig 1. While MPTCP may mitigate such issues [5], each of its subflows may still experience conservative congestion control and significant throughput drop during handovers. Also, it requires non-trivial network infrastructure support. Prior research also explored TCP over satnets [6], [7], yet focusing on geostationary systems without any handover issues. Whereas certain access networks such as cellular networks also experience handover, the impact on the transport layer is relatively small, because handover occurs only on a single last-hop link, and also because of their relatively shorter RTT (and hence faster recovery) [8].

In this paper, we first conduct an in-depth measurement study to investigate the limitations of TCP in highly dynamic LEO satnets. To facilitate the measurement, we develop a LEO satnet emulator called **LeoEM**. LeoEM allows real-time full-stack emulation of dynamic LEO network paths, and can run unmodified applications on the host OS. We populate the LeoEM link layer with real-world measurement statistics and specifications, to enable accurate representation of the low-level dynamics (*e.g.*, satellite handovers). With LeoEM, we examine Starlink–the currently most promising and representative LEO satellite network. In particular, we analyze the link-level dynamics of the Starlink constellation, and examine the corresponding interruption and performance degradation on the transport layer. Our experimental results show that the popular CUBIC TCP experiences an extremely low bandwidth utilization (around 23.3%) over long-distance LEO satnet paths due to frequent rate fallback in response to the high dynamics and link fragility.

Based on the measurement insights, we propose a mechanism called **SaTCP** that can enhance TCP amid the LEO dynamics. SaTCP builds on two design principles. *First*, it advocates *link-layer informed* TCP adaptation, *i.e.*, the TCP should discriminate link-layer dynamics from real congestion events. *Second*, SaTCP follows *speculative congestion adaptation*. Rather than reacting to link dynamics post hoc, SaTCP leverages the relative predictability of satellite locations and handover events, to proactively prepare for the dynamics. More specifically, SaTCP predicts when an upcoming handover will occur and signals the client node to freeze its congestion window size for a very short instance. It is worth noting that the satellite location is not fully deterministic due to various turbulence and forces [9]–[11], and hence the actual handover event times can deviate from the planned ones. SaTCP takes the prediction inaccuracy into account and properly schedules its reactions. Our evaluation results show SaTCP can effectively prevent unnecessary throughput degradation caused by misjudgments of loss events and varying satenet latency, hence achieving consistent high bandwidth utilization over a wide range of LEO satellite network scenarios.

In summary, the main contributions of this paper include:

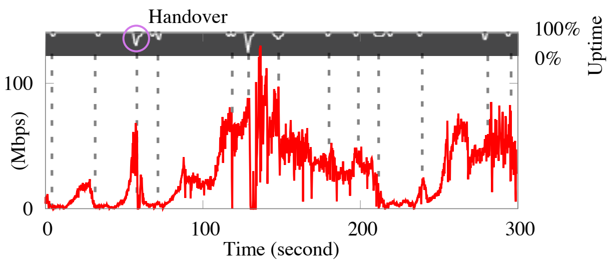

Figure 1: Example downlink TCP throughput using Starlink. Handovers cause network interruptions and TCP transmission rate fallback.

(i) A high-fidelity and highly configurable LEO satellite network emulator. We have developed LeoEM, which allows real-time full-stack emulation of any LEO satellite network. Arbitrary applications and protocols can be used to customize the network and test various scenarios, hence offering a flexible platform for researchers to develop solutions to specific challenges in LEO satellite networking. LeoEM is available at https://github.com/XuyangCaoUCSD/LeoEM.

(ii) A solution to improve TCP performance in dynamic LEO satellite networks. We have designed SATCP, a cross-layer solution that follows a rigorous procedure to predict and report upcoming satellite handovers to clients, whose congestion control will then be temporarily inhibited to prevent unnecessary throughput fallback. SATCP largely improves classical TCP's performance, inherits the fairness property, and is applicable to any mainstream LEO satellite network.

II. Related Work

Experimental tools for LEO satellite networks. Several recent projects have developed tools to investigate mainstream LEO satellite networks. Through mathematical modeling, StarPerf [12] can assess various metrics of a LEO satellite network, including resilience of constellation upon satellite failures, satellite coverage rate, and theoretical latency. However, it lacks the actual network protocol simulation. Hypatia [13] is a ns-3 based LEO satnet simulator that can capture essential network details like RTT and TCP throughput. However, it lacks the flexibility to test arbitrary applications as the traffics are internally simulated rather than generated by native network stacks. Also, Hypatia takes non-trivial time to complete a simulation. In contrast, our LeoEM emulator allows running any real programs over emulated dynamic LEO satnets with real-time observability.

LEO satellite network characterization and evaluation. Existing research mainly profiled LEO constellations with bent-pipe links and inter-satellite links (ISLs), as shown in Fig. 2. Satnet paths formed by ISLs can achieve much lower latency than the bent-pipe versions or terrestrial networks [14], [15]. However, careful constellation design and connection fine-tuning are required in order to effectively utilize ISLs. For example, with the current Starlink topology, paths formed by fixed ISLs can experience a high zig-zag latency variation due to LEO dynamics and occasional lack of optimal path [13], [15]. At the peak, the latency can still exceed its terrestrial counterpart. This paper confirms such phenomenon and takes further steps to characterize the impacts on state-of-the-art transport layer protocols.

Using the Hypatia simulator, [13] examined the impact of LEO constellation dynamics on end-to-end network paths. For both bent-pipe-link and ISL cases, the dynamics lead to frequent route re-selections and thus varying bandwidth-delay product (BDP), which triggers overly conservative TCP congestion windows. However, Hypatia does not address the prominent handover issue that has been widely observed by LEO satellite network users (e.g., [16]–[18] and Fig 1). In contrast, we aim to profile the impact of link-level dynamics, *i.e.*, handover interruptions, and design the corresponding solutions.

Handover in satellite networks. Non-geostationary satellite handover can be categorized into two types: (1) intra-satellite spot-beam handover, and (2) inter-satellite handover. For (1), the ground node will switch its connection to a different spot beam of the same satellite, since a spot beam may fix its angle and only serve a particular cell within the coverage. For (2), the ground node will switch its connection to a different satellite. Various network protocols and resource allocation principles have been proposed to achieve effective handovers [19], [20]. Standardization efforts are underway [21] to incorporate such solutions and integrate the LEO network as an extension to the 5G infrastructure. The present work focuses on the latter due to the limited mobility of ground nodes in the current LEO satnets and the adoption of steerable spot beams, for example, by Starlink [22].

TCP amendments for dynamic networks. Congestion control protocols for dynamic networks have been extensively explored. Earlier work [23] proposed to use single-packet loss in an RTT as signs of handover or transmission errors in LEO satnets. But such approaches cannot distinguish high dynamics produced by thousands of satellites at today's scale from real congestion events. In contrast, SATCP utilizes validated link-layer information to make such distinctions. Another recent work uses link-layer feedback to quickly ramp up TCP throughput upon adoption of high-bandwidth circuit-switch paths [24]. However, it relies on explicit congestion notification (ECN), which requires in-network support and router hardware upgrade. Likewise, to align with the end-to-end principle, SATCP avoids using any active queue management (AQM) techniques, and requires no modification to LEO satnet backbones. Physical or link layer informed TCP enhancements have also been explored in cellular networks [8], [25], but to realize a similar principle, LEO satnet-specific designs are needed, which we explore through SATCP.

III. Preliminary

In this section, we will cover the essentials of LEO satnets.

A. LEO Satellite Network Topology

LEO satnets use dish-like *user terminal (UT)* on the ground to communicate with one satellite overhead (referred to as *ingress satellite*). The ingress satellite further wirelessly connects to the *gateway ground stations* which in turn join the wired backbone. Fig. 2 shows a typical network topology, where the UT, ingress satellite, and nearby ground station together form the access network. End hosts, or *user equipment (UE)*, access the Internet via the UT.

To unlock the full potentials of the LEO satnets, it is crucial to have a *backbone network* consisting of multiple satellite relays and ground stations. The backbone network can adopt either the *bent-pipe (BP)* links or *inter-satellite laser links (ISLs)* as shown in Fig. 2. The former interleave a series of LEO satellite and ground stations using high-frequency radio to form an end-to-end path, whereas the latter leverage high-bandwidth laser beams among satellites. LEO satellites are organized into one or more *shells* which collectively form a *constellation*. Satellites of the same shell have the same altitude.

B. Satellite handovers

Ground equipment in a LEO satnet needs to periodically switch its association from one satellite to another. Such handover can be categorized into two types, both referred to as a *disruptive event*.

End (hard) handover: handover from one ingress satellite to another for a UT. Often equipped with only one antenna, the UT needs to first disconnect from its current ingress satellite before connecting to a new one. Such *hard handover* can cause a short but non-negligible network interruption. The interruption, reasonably quantified by [21], can last for 1.5 RTT between the UT and the old ingress satellite, plus 1.5 RTT between the UT and the new ingress satellite. This is because the UT needs to first query for the spot beam availability in the new satellite, plan to release the current spot beam to the old satellite, reserve the new spot beam, and finally fully disconnect from the old satellite. During the handover, packets can accumulate at the ingress queues of the UT and possibly get dropped. Also, because of the handover, the old end-to-end route can get dismantled due to link breakage or delayed routing updates. Flowing packets in the old route thus can get lost too.

Intermediate (soft) handover: handover from one satellite to another for a gateway ground station for bent-pipe links. To aggregate and redistribute traffics, the ground stations are equipped with multiple antennas which in turn can connect to adjacent satellites simultaneously and allow relatively seamless *soft handover*. Such intermediate handover instructs ground stations to redirect certain traffics to new paths before dismantlement of old routes, during which packet loss can still occur. Note ISLs usually adopt fixed connectivity, so intermediate handovers for satellites in ISL paths often come with only path re-selection, without any link dismantlement.

C. Starlink Shell 1 Properties

To date, Starlink shell 1 is the very LEO satellite network that has provided experimental yet decent Internet service to a good number of users. The shell consists of 72 orbits, each containing 22 satellites [27], with altitude of 550km and an orbital period of only 1.59 hours. Fig. 3 visualizes the shell. Starlink satellites are equipped with steerable spot-beam antennas [22] which are dynamically assigned.

Currently, the default residential Starlink [1] only provides access networks. Nonetheless, periodic (hard) handovers and network interruptions have been experienced by users. We showcase the problem through a measurement benchmark. We deploy a Starlink client on the rooftop of a tall building in San Diego. Then we dump CUBIC TCP traffic from an AWS EC2 in Oregon to the Starlink client using iPerf. Meanwhile, we log the network uptime time series obtained from the client debug console. As shown in Fig. 1, the presence of uptime drops, which essentially denote handovers, causes misinformed TCP CWND and throughput drops. The speed test from debug console indicates 139Mbps available bandwidth, yet the experienced average TCP throughput across the 300 seconds is only 28.8Mbps. The impact of the dynamics on the network is thus nontrivial.

Theoretically, a shell-1 satellite can cover a ground node for at most 2.5 minutes, given its coverage radius of 580km [4]. However, due to practical reasons like line-of-sight blockages, low signal strength, or dismantlement of the current route, UTs may undergo frequent handovers. Specific to Starlink, the network needs to consider handoffs on a 15-second interval [28]. Advanced services like Starlink Maritime [29] or Aviation [30] will have the backbone fully or partially supported by its own infrastructure, adopting bent-pipe links or ISLs [31], [32].

We emphasize that the moving trajectories and schedules of satellites are *not fully deterministic*, as they are constantly subject to turbulence and uneven gravitational forces [9]–[11], causing deviation from their planned orbits. Therefore, each satnet, including Starlink, is required to have a telemetry, tracking, and control (TT&C) subsystem [22], which can fine-tune a satellite's orbit if any trajectory deviation occurs.

IV. The LeoEM Satellite Network Emulator

Existing satnet simulation tools like Hypatia and StarPerf [12], [13] do not support the experimentation of arbitrary protocol suites, particularly at the transport layer or above. LeoEM is built to fill this gap. *It faithfully represents not only the LEO satnets but also the host OS, so any program can be natively run and evaluated over the dynamic links in real time.* Not constrained by application-level simulation, the network has a high degree of real-time observability. For example, video streams can be exchanged between two ends of an emulated satnet path, isolated by two network namespaces. Meanwhile, the media quality can be directly monitored through the playback. Therefore, LeoEM provides a powerful and flexible platform for researchers to experiment their innovations targeting LEO satellite networks.

LeoEM architecture. LeoEM consists of several stages glued by data pipelining. In the first stage, users specify the shell to be analyzed through the constellation parameters (*e.g.* shown in Fig. 3). Then location data of each satellite in the shell are generated at a certain sampling rate. The timeline of location data spans across one orbital period, as they repeat afterwards.

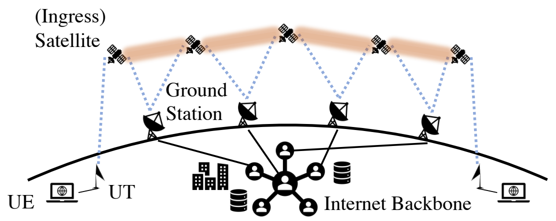

Figure 2: LEO satnet topology. UE (laptop) connects to dish-like UT and reach the Internet or another satnet user through ISLs (glowing line) or BP links (dash line).

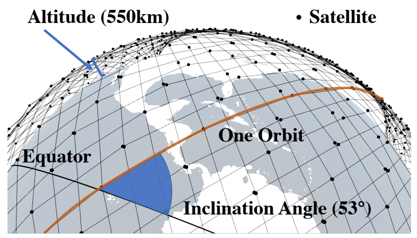

Figure 3: Starlink shell 1 visualization, generated by the CesiumJS framework [13], [26].

The second stage computes the route at each sampled frame between a source node and a destination node, adopting either bent-pipe links or ISLs along the backbone. The former requires the user to provide a list of ground station locations. For the latter, LeoEM uses the fixed +Grid connectivity pattern [33], where each satellite has four laser links connecting to its two adjacent peers on the same orbit and two on the neighboring orbits. To establish the route between the source and destination, we adopt the classical shortest path routing.

Notice LEO satellite networks, as proprietary systems, will likely tunnel user traffics and hide their internal network nodes. For example, tracerouting Starlink does not show any internal hop except the draining points to terrestrial backbones (e.g., [34], [35] and Fig. 7). Given the lack of transparency, we refer satellites and ground stations both into switching nodes without clear layering, in which their network-level information is hidden from users.

The third stage of LeoEM emulates the LEO satnet stack using Mininet [36]. Through Mininet APIs, virtual switching nodes and links are spawned in the host OS to represent satellites, ground stations, UTs, and their connections. The routes precomputed from the second stage are used to continuously update link properties such as propagation latency. In our default configuration, the UTs refresh their ingress satellites every 15 seconds, as suggested by Starlink's FCC filing [28]. To faithfully emulate the end handover, we make the UT seek a new ingress satellite of the shortest distance. During each end handover, LeoEM tears down the link for 1.5 RTT between the UT and the old ingress satellite and 1.5 RTT between the UT and the new ingress satellite, as explained in section III-B. Intermediate handovers are determined by comparing routes at two consecutive time frames, and we turn the corresponding nodes' network interfaces off and on to produce the churn effect and possible packet loss in the dismantled old route. Different Linux namespaces are attached to UTs, where users can run arbitrary programs and evaluate the network in real time.

V. SATCP System Design

Traditional TCP, like the widely deployed CUBIC, is vulnerable to LEO satellite networks' strong dynamics and link fragility, as shown in Fig 1. To enhance TCP, SATCP leverages the predictability of disruptive events in LEO satellite networks to adaptively perform congestion control. By taking satellite link-level information, SATCP manages to distinguish between real congestion events and disruptive events and inform client nodes to avoid unnecessary fallback of congestion window.

A. SATCP *Methodology*

To materialize the design rationale of SATCP, we exploit a few salient properties of LEO satnets.

1. Predictability of satellite locations. Despite the high mobility, LEO satellites usually try to follow predefined orbit paths. Remarkably, *the LEO satellite trajectories are relatively predictable but not fully deterministic*. Due to the uneven gravitational forces over different points of the Earth and various disturbance, a satellite can still deviate from its trajectory and the error can accumulate over time [9]–[11].

2. Computability of a disruptive event time. Given certain satellite locations and the handover scheme, we can compute when a disruptive event will occur for a specific user, namely when the UT or an intermediate switching node will update its next hop and adopt a new route.

Based on the above properties, we introduce 2 key mechanisms in realizing SATCP.

How should the system effectively determine and report the upcoming disruptive event time to the UE in advance?

Intuitively, the UT should be the only ideal candidate for tracking **end** handover times, as it participates in the handover process and periodically collects information from various viewable satellites (*e.g.*, relative locations or signal strengths). Also, since the UT and UEs are co-located, the UT should be able to report end handover times readily and reliably.

In addition, we select the nearby ground station which the ingress satellite connects to as the component for tracking and reporting any **intermediate** handover time to a UE. The ground station may need the dynamic global satellite locations, and constantly tracking them may be unrealistic for an infrastructure except the TT&C system. However, benefiting from the predictability of celestial objects, it only needs to have one snapshot of the constellation and forecast its future state, at trivial computational cost. Note that for either bent-pipe or ISL backbone, a ground station should always be available nearby so the user can connect to the Internet.

After the UE receives the report, what should it do to mitigate the negative impacts of the disruptive events?

A simple relay application in the UE can listen to reports and pass it to the network stack. The TCP logic can be modified such that the congestion window reduction will be inhibited for a short duration upon receiving a report. When

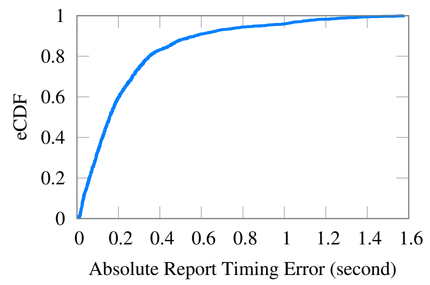

Figure 4: Empirical cumulative distribution of the absolute report timing error.

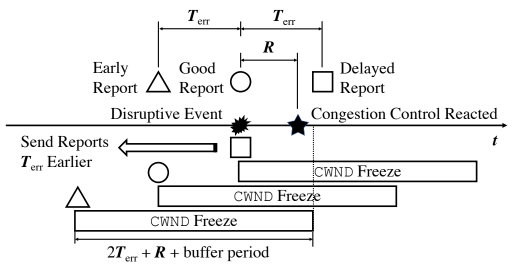

Figure 5: Mitigation of report timing error by sending a report T_{err} earlier.

a disruptive event occurs, it will take some time for the congestion control module to react. Therefore, the duration of the congestion window freeze should cover the point when the congestion control would otherwise start reacting. In this way, the unnecessary throughput drop can be prevented.

We emphasize that, the inhibition period of congestion window reduction is very short, typically on the order of one or two RTTs, even taking into account the additional time needed to tolerate report timing errors (Sec. V-B). Even in the rare case when a real congestion occurs rightly within this period, SATCP can react immediately after this short period, when it reverts to the default TCP behavior.

Note LEO satnets, represented by Starlink [1], provision network resources accordingly. Therefore, the SATCP sender, with preserved CWND value, should not congest the new links after a handover. Resource management components, usually at the edge of the network, will throttle excessive amount of traffics from a particular user. Otherwise it becomes a service quality problem.

B. *Tolerating Error in Orbital Prediction*

The predicted satellite location can be inaccurate, deviating from the actual one. This inaccuracy can negatively impact the effectiveness of SATCP.

We utilize the publicly available two-line element set (TLE) for Starlink from CelesTrak [37] as an example to quantify the error of satellite location prediction and how the inaccuracy actually affects the SATCP effectiveness. A TLE records satellite locations at a certain moment and their trajectories, which can be used to predict their future locations. The public Starlink TLEs are updated every 3 hours.

We use a TLE snapshot to predict the satellite locations 3 hours later, and the results will then be compared against the actual TLE measured 3 hours later. Given the difference between the predicted satellite location and the actual one, we can quantify the amount of time a satellite will take in order for its corresponding predicted coverage area to overlap with that of the actual satellite (or conversely, the time needed for the actual coverage area to catch up the predicted one). This time derivation essentially decides how long the disruptive event report can get delayed or occur ahead of the right moment. We refer to this gap as *report timing error*.

We compute the report timing errors for a total of 1605 satellites, using MATLAB Satellite Communications Toolbox that supports satellite orbit determination. Fig. 4 shows the eCDF of the absolute report timing errors. We can observe that around 90% of satellite location predictions produce a report timing error within the range of ± 0.6 seconds, namely around 90% of disruptive event reports should be sent no more than 0.6 seconds earlier or later than the right moment.

To counteract the report timing error, the ground station can report the disruptive event in advance and the congestion window freeze time can be extended, as shown in Fig. 5. Specifically, given estimated report timing error $\pm T_{\text{err}}$ and congestion control time R (the time TCP sender takes to realize packet loss/link dynamics), the ground station can send the report T_{err} earlier than the estimated disruptive event time (after also considering communication latency). Upon receipt of the report, the UE will also set the duration of congestion control inhibition to slightly greater than $2 * T_{\text{err}} + R$. In this way, any delayed or early report by no more than T_{err} error can be corrected. Based on Fig. 4, setting T_{err} to 0.6 seconds and applying the adjustment should neutralize around 90% effect of the report timing inaccuracy.

We note that the report timing errors illustrated in Fig. 4 should represent the upper bound, for several reasons. First, rather than the 3-hour update interval of the TLEs, practical LEO satellite systems have TT&C, allowing for a much finer-grained location update. Second, the TT&C system can fine-tune the movement of a satellite and maneuver it back to the planned path and location, which should further mitigate the prediction error. Finally, in reality, the satellite may drift away along any direction from the expected location. Given the same drift distance, the parallel drift we consider yields the maximized absolute report timing errors.

C. SATCP *Workflow*

We now introduce the SATCP system architecture and workflow, based on the key design elements in section V-A and section V-B.

As shown in Fig. 6, the UT notifies the UE about upcoming end handovers. The UT also periodically tracks the latency between itself and the UE, denoted as L_{UT} (*e.g.*, with Ping utility or by piggybacking on ongoing packets). For an anticipated handover that occurs at t, the UT will send the report at $t - L_{\text{UT}}$ to ensure timely actions.

Similarly, with the cached global satellite location data, the nearby ground station will be responsible for estimating the upcoming intermediate handover times for its nearby users and notify them. It also periodically receives satellite location updates from the TT&C and tracks L_{GS}–its latency to a UE. To alleviate the effect of unavoidable report timing error, TT&C periodically computes the report timing error

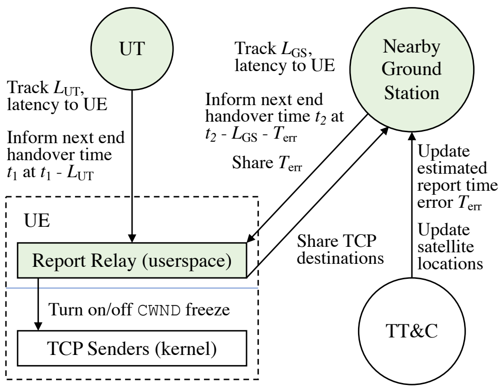
UT
Track L_{UT}, latency to UE
Inform next end handover time t_1 at t_1 - L_{UT}
Track L_{GS}, latency to UE
Nearby Ground Station
Inform next end handover time t_2 at t_2 - L_{GS} - T_{err}
Share T_{err}
Update estimated report time error T_{err}
Update satellite locations
UE
Report Relay (userspace)
Share TCP destinations
Turn on/off CWND freeze
TCP Senders (kernel)
TT&C

Figure 6: SATCP architecture, components that require modifications, and data flows.

distribution using the technique described in section V-B and chooses the aforementioned high-percentile value T_{err}. The ground station uses T_{err} as the estimated report timing error. Therefore, for an intermediate handover at t, the ground station will report it to the UE in advance, at $t - L_{\text{GS}} - T_{\text{err}}$, following the discussion in section V-B.

Upon receiving the disruptive event reports, the UE will instruct the TCP senders to properly inhibit their window reduction. Following the discussion in section V-B, the inhibition period is $2 * T_{\text{err}} + \texttt{TCP\_RTO\_MIN} + 0.1\text{s}$, where T_{err} is shared by the ground station, `TCP_RTO_MIN` is the time a TCP sender waits to retransmit an unacknowledged packet for the first time, and 0.1s is a small buffer period. Notice the presence of an intermediate handover is specific to where the data flows. Therefore, the UE also needs to share the destinations of its current TCP connections with the nearby ground station. The ground station can in turn estimate whether an intermediate handover will impact the end-to-end path and issue a report accordingly.

The workflow is very similar if the TCP connection is from terrestrial networks to a satnet user. The UT and draining ground station can see the sender's IP address and hence notify it about upcoming disruptive events, given the SATCP upgrade in the sender. SATCP largely conforms to the end-to-end design principle [38]. The solution requires no in-network upgrade. Sharing end handover information with UEs can be realized through a small software-level update at the UT. Computing and reporting intermediate handovers can be carried out by auxiliary services at the ground station and TT&C, independent to the network backbone. SATCP enhancements can be added to the UEs through OS patching, and we will show that the modification takes only several lines of code.

D. *Implementing* SATCP *in the network stack*

Without loss of generality, we use the widely deployed TCP CUBIC as an example to explain how existing TCP protocols can incorporate the SATCP mechanisms. Algorithm 1 shows the pseudo-code, where lines annotated with `for-SaTCP` are added to implement SATCP. A netlink socket is initialized through `netlink_create()` in `cubictcp_register()` to allow communications between the kernel and the report relay program. The boolean variable `SaTCP_flag` will determine whether at this moment the congestion window should be frozen. The callback function `on_report` is passed to `netlink_create()` so the relay application can instruct the CUBIC module to turn on and off the congestion control inhibition, based on the inhibition duration $2 * T_{\text{err}} + \texttt{TCP\_RTO\_MIN} + 0.1\text{s}$ mentioned in section V-C. In `cubic_update()`, whenever a packet loss is detected and `SaTCP_flag` is true, we reduce K to its $\frac{1}{9}$ so $cwnd$ reduction will be skipped. Given the CUBIC's $cwnd$ growth model, we set K to its $\frac{1}{9}$ instead of 0 to also avoid increasing $cwnd$ and congesting the UT's ingress

Algorithm 1: SATCP-enabled CUBIC on Linux

```
$C \leftarrow 0.4$, $\beta \leftarrow 0.2$
$satcp\_flag \leftarrow false$                          // for-SATCP
Procedure on_report(report)                             // for-SATCP
    if $report = 1$ then                                // for-SATCP
        $satcp\_flag \leftarrow true$                   // for-SATCP
    if $report = 0$ then                                // for-SATCP
        $satcp\_flag \leftarrow false$                  // for-SATCP
Procedure cubictcp_register()
    ... (initialize CUBIC TCP data structures)
    $netlink\_create(on\_report)$                       // for-SATCP
Procedure on_receive_loss()
    $epoch\_start \leftarrow 0$;
    if $cwnd < W_{last\_max}$ then
        $W_{last\_max} \leftarrow cwnd * \frac{2-\beta}{2}$
    else
        $W_{last\_max} \leftarrow cwnd$;
    $ssthresh \leftarrow cwnd \leftarrow cwnd * (1-\beta)$
// Triggered on each ACK
Procedure cubic_update()
    // If packet loss just occurred
    if $epoch\_start \leq 0$ then
        $epoch\_start \leftarrow tcp\_time\_stamp$
        if $cwnd < W_{last\_max}$ then
            $K \leftarrow \sqrt[3]{\frac{W_{last\_max} - cwnd}{C}}$
            $origin\_point \leftarrow W_{last\_max}$
            if $satcp\_flag$ then                       // for-SATCP
                $K \leftarrow \frac{1}{9}K$             // for-SATCP
        else
            $K \leftarrow 0$;
            $origin\_point \leftarrow cwnd$
    $t \leftarrow tcp\_time\_stamp + dMin - epoch\_start$
    $target \leftarrow origin\_point + C(t-K)^3$
    ... (update $cwnd$ based on $target$ value)
Procedure timeout()
    if $satcp\_flag$ then                               // for-SATCP
        return()                                        // for-SATCP
    $cubic\_reset()$
```

queue in case of end handovers. Similarly, in `timeout()`, if `SaTCP_flag` is true, we skip unnecessary TCP reset.

VI. Evaluation

In this section, we evaluate SaTCP, in comparison with the widely deployed CUBIC TCP and the state-of-the-art BBR [39], across a wide range of scenarios.

A. Experimental Setup

Our experiments focus on Starlink shell 1, emulated through LeoEM on a commodity laptop. The ground station locations are obtained from the live Starlink satellite and coverage map [40]. We experiment with 6 pairs of source-destination locations corresponding to different path lengths: San Diego (SD) and New York City (NY), Seattle (SEA) and New York City, San Diego and Seattle, San Diego and Shanghai (SH), New York City and London (LDN), and Seattle and Buenos Aires (BA). ISLs are available for all 6 pairs. However, only the first 3 intra-continental pairs can adopt bent-pipe links as ground stations are available only along their paths. The different path lengths and links type can lead to different levels of dynamics.

In order to represent the disruptive event report from UTs and ground stations illustrated in Fig. 6, LeoEM notifies upcoming disruptive events with the virtual test hosts at proper time. The CUBIC module in the test hosts with Ubuntu 20.04 and Linux kernel 5.4.143 is modified based on Algorithm 1. To create report timing error, LeoEM adjusts the report time by a random amount based on the eCDF in Fig. 4. According to discussion at section V-C, we set the estimated report timing error T_{err} to 0s, 0.3s, 0.5s, and 1s, and pick a reasonable constant value 0.2s for `TCP_RTO_MIN` to see how effective SaTCP counteracts the report time estimation inaccuracy, using different configurations. To match Starlink's claimed network throughput of 100Mb/s to 200Mb/s [1], we use Linux tc to cap the upload/download bandwidth of emulated links to 150Mb/s. The data streams are generated through an iPerf server and client which reside at the two test hosts. We leave tc's default 1000-packet queue length untouched. Given our 5**KB** iPerf payload size, switching nodes should be able to buffer data up to 260ms, much longer than any handover duration. In this way, packet loss is caused mostly by path dismantlement due to actual link fragility, rather than avoidable queue length inadequacy.

Fig. 7 presents some 200-second time series to illustrate the latency and the occurrences of disruptive events in two test cases. Due to flexible connectivity between ground stations and satellites, bent pipe links tend to be more fragile and dynamic, as manifested by an increasing number of disruptive events. For ISLs, the default +Grid topology fixes the connectivity between every node and its four neighbors, so route updates occur mostly during last-hop handovers. While the relatively low dynamics of +Grid ISLs is desirable, Fig. 7 shows they produce a very problematic, highly varying latency pattern, aligned with the results from a recent study [15]. For a significant amount of time, the RTT of +Grid ISLs far exceeds even the terrestrial counterparts. The root cause is that due to inflexible connectivity, sometimes even the shortest ISL path may need to traverse through the other side of the Earth, leading to very high latency. In contrast, the bent-pipe links offer lower latency than their terrestrial counterparts. Indeed, since satellites and ground stations can freely connect with each other, a relatively straight path can be easily obtained. However, higher flexibility leads to greater dynamics. In Fig. 7 we also presented 200-second RTT between an actual Starlink client in SD and an AWS EC2 in Ohio and the visual traceroute. We can see the current access-only Starlink already demonstrated high latency dynamics similar to those from LeoEM. However, due to the access-only nature, relatively inefficient route, and extra processing time, the latency is still relatively high compared to the terrestrial ping.

B. Performance Evaluation

For the six pairs of locations, traffics of BBR, CUBIC, and SaTCP with different report time adjustments are generated and measured for one entire orbital period using ISLs, and if allowed, bent-pipe links.

As we can observe from the results in Table I, CUBIC performs poorly over the SD $\leftrightarrow$ NYC and SEA $\leftrightarrow$ NYC BP-link paths, achieving only 22.9% and 30.0% capacity utilization. According to Fig. 7, the bent-pipe links exhibit a much higher number of disruptive events than others. The high dynamics lead to frequent packet losses and CUBIC's constant congestion window backoffs, which is further illustrated in Fig. 8.

However, for the SEA $\leftrightarrow$ SD bent-pipe path, CUBIC preforms relatively well, achieving an average of 138.9Mbps with 92.6% utilization. This is because the shorter path involves a smaller number of links, and the occurrences of link breakages are infrequent enough such that TCP can ramp to the capacity limit before the next disruption. Likewise, CUBIC exceeds 135Mbps on average for the six ISL paths. As shown in Fig. 7, there are much fewer disruptive events in ISLs than in their bent-pipe counterparts. The underlying reason again lies in the fixed +Grid connectivity patterns and hence rare intermediate handovers.

As for SaTCP, even without the report timing adjustment (and with the shortest duration of CWND freeze), the throughput has already been enhanced to 1.74$\times$ of CUBIC and 1.47$\times$ for SD $\leftrightarrow$ NYC and SEA $\leftrightarrow$ NYC over the bent-pipe links. However, the absolute throughput, namely 59.8Mbps and 66.3Mbps, is still far away from the link capacity limit. The reason lies in the report timing errors caused by the non-deterministic satellite locations and hence prediction inaccuracy (Sec. V-B). This makes the congestion control inhibition, in many cases, fail to cover the point where the TCP sender actually reacts to packet losses and cuts its window size.

Fig. 8 shows one such case in SD $\leftrightarrow$ NYC bent-pipe path, where the disruptive event report (up triangle symbol) arrives too early at the UE before the actual disruptive event occurs (pentagon symbol) and the inhibition duration is insufficient, leading to pointless congestion window size reduction.

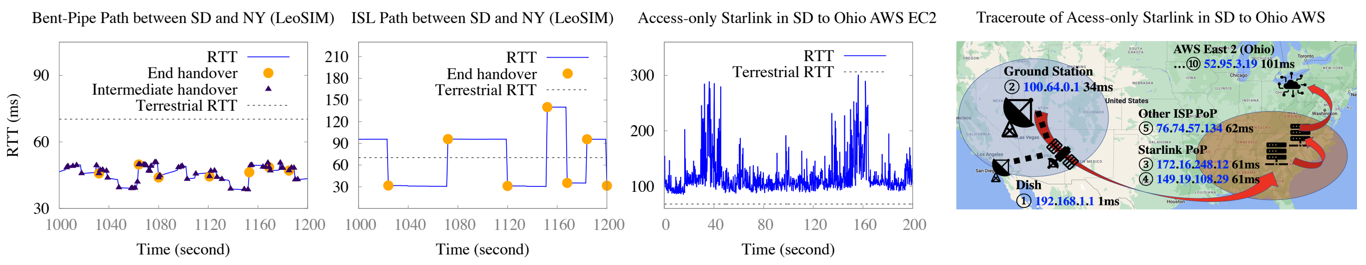

Figure 7: RTTs over 200 seconds between SD and NY. Terrestrial counterparts [41] are presented for comparison. Measured RTT between access-only Starlink client in SD and Ohio AWS EC2 is also presented with the visual traceroute, where the circles denote our estimated PoP geographic ranges.

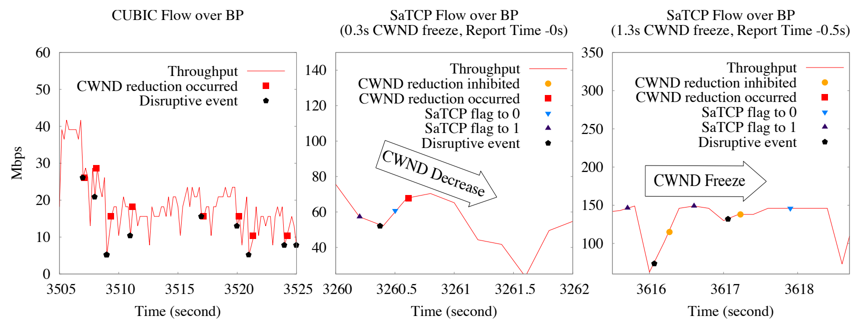

Figure 8: Example throughput time series for CUBIC and SaTCP between San Diego and New York.

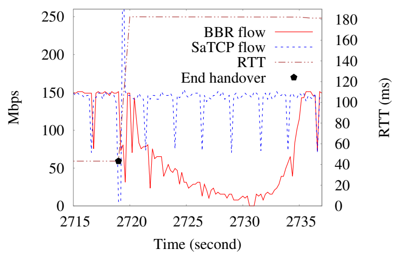

Figure 9: Comparisons of BBR and SATCP flows from NY to LDN over ISLs. SATCP has 0.9s CWND freeze and -0.3s report time adjustment.

The effectiveness of SATCP is largely amplified after taking report timing errors into consideration. For instance, by sending disruptive event reports 0.5s earlier and extending the inhibition duration to 1.3 seconds, SaTCP achieves 135Mbps+ near-optimal throughput over SD $\leftrightarrow$ NY and SEA $\leftrightarrow$ NY bent-pipe paths. We counted 91.7% and 92.1% undesired congestion window reduction have been prevented respectively.

In Fig. 8, we can see that the disruptive events around 3616s and 3617s have been gracefully handled by the two timely report signals, after sending reports 0.5s earlier and extending the congestion window freeze to 1.3s. Because of the preserved CWND value, the throughput quickly ramps up from the unavoidable drop caused by network interruption, recovering to 150Mbps link limit. Further increasing the inhibition duration, *e.g.*, to 2.3 seconds as shown in Table I, leads to little improvement.

As shown in Table I, BBR has very high throughput over all BP-link paths, similar to SATCP with adjusted report time and extended CWND freeze period. Indeed, BDP-based BBR adjusts its throughput based on observed delay gradient instead of packet loss, and thus it is immune to high bent-pipe link fragility. However, for ISL scenarios, the performance of BBR is comparatively mediocre. For example, BBR has only 130Mbps average over NY $\leftrightarrow$ LDN and SD $\leftrightarrow$ SH long-distance ISL paths, around 10Mbps less than their SaTCP counterparts.

While the average seems decent, we found out BBR attains very low transient throughput. The root case, illustrated in Fig. 9, is BBR's delay-based feedback and ISLs' high latency variation. Around 2719s, the NY $\leftrightarrow$ LDN ISL path undergoes an end handover and adopts a new route with much higher latency. BBR, perceiving the latency spike as queue congestion, drastically slows down its pace and under-utilizes the bandwidth for around 15s until its next *ProbeRTT*. The transient throughput drops can negatively impact real-time applications, *e.g.*, resulting in several seconds' freeze in a video telephony session. On the contrary, SaTCP maintains near full bandwidth utilization. Note the throughput spike is because of packet accumulation at the UT during the end handover, along with iPerf's coarse throughput sampling. Overall, across all such transient periods over ISLs, we found out SATCP achieves an average throughput gain of 2.34× over BBR and 1.22× over CUBIC.

Recall that the report timing errors we derive and use should form an upper bound, as explained in section V-B. Therefore, in practical scenarios, the effective duration of congestion control inhibition likely can be further reduced. The SATCP results for some test cases are not presented because CUBIC already achieves a near-optimal throughput and the improvements become insignificant.

SATCP fairness. To see if SATCP can behave in an exceedingly aggressive manner to gain unfair advantages, we initiate two concurrent SATCP flows in highly dynamic SD $\leftrightarrow$ NYC bent-pipe path. The duration of CWND freeze is configured to be 0.3s, 0.9s, 1.3s, and 2.3s respectively, which represents increasing levels of aggressiveness. Table II shows that the two flows achieve very similar average throughput in all settings. Concurrent SATCP and CUBIC flows are also tested. We can see with 0.9s inhibition duration, SATCP flow gives up around 12Mbps and CUBIC flow gives up around 17 Mbps compared with the results in Table. I. With

Protocol	Location Pair	Link Type	Inhibited Congestion Control Duration	Report Timing Adjustment	Avg. Throughput (max 150Mbps)	Throughput Improvement to CUBIC
Intra-Continental Paths						
BBR	SD ↔ NY	Bent Pipe	N.A.	N.A.	136.7Mbps	3.98×
CUBIC	SD ↔ NY	Bent Pipe	N.A.	N.A.	34.3Mbps	1.0×
SATCP	SD ↔ NY	Bent Pipe	0.3s	0s	59.8Mbps	1.74×
SATCP	SD ↔ NY	Bent Pipe	0.9s	-0.3s	123.4Mbps	3.60×
SATCP	SD ↔ NY	Bent Pipe	1.3s	-0.5s	138.1Mbps	4.03×
SATCP	SD ↔ NY	Bent Pipe	2.3s	-1.0s	139.4Mbps	4.06×
BBR	SD ↔ SEA	Bent Pipe	N.A.	N.A.	138.3Mbps	0.99×
CUBIC	SD ↔ SEA	Bent Pipe	N.A.	N.A.	138.9Mbps	1.0×
BBR	SEA ↔ NY	Bent Pipe	N.A.	N.A.	137.6Mbps	3.05×
CUBIC	SEA ↔ NY	Bent Pipe	N.A.	N.A.	45.1Mbps	1.0×
SATCP	SEA ↔ NY	Bent Pipe	0.3s	0s	66.3Mbps	1.47×
SATCP	SEA ↔ NY	Bent Pipe	0.9s	-0.3s	123.9Mbps	2.75×
SATCP	SEA ↔ NY	Bent Pipe	1.3s	-0.5s	135.0Mbps	2.99×
SATCP	SEA ↔ NY	Bent Pipe	2.3s	-1s	139.8Mbps	3.10×
BBR	SD ↔ NY	ISL	N.A.	N.A.	130.1Mbps	0.94×
CUBIC	SD ↔ NY	ISL	N.A.	N.A.	138.3Mbps	1.0×
BBR	SD ↔ SEA	ISL	N.A.	N.A.	138.5Mbps	0.97×
CUBIC	SD ↔ SEA	ISL	N.A.	N.A.	142.8Mbps	1.0×
BBR	SEA ↔ NY	ISL	N.A.	N.A.	136.1Mbps	0.96×
CUBIC	SEA ↔ NY	ISL	N.A.	N.A.	141.8Mbps	1.0×
Inter-Continental Paths						
BBR	NY ↔ LDN	ISL	N.A.	N.A.	131.0Mbps	1.02×
CUBIC	NY ↔ LDN	ISL	N.A.	N.A.	128.5Mbps	1.0×
SATCP	NY ↔ LDN	ISL	0.3s	0s	135.1Mbps	1.05×
SATCP	NY ↔ LDN	ISL	0.9s	-0.3s	140.4Mbps	1.09×
BBR	SD ↔ SH	ISL	N.A.	N.A.	130.4Mbps	0.99×
CUBIC	SD ↔ SH	ISL	N.A.	N.A.	131.3Mbps	1.0×
SATCP	SD ↔ SH	ISL	0.3s	0s	133.4Mbps	1.02×
SATCP	SD ↔ SH	ISL	0.9s	-0.3s	139.7Mbps	1.06×
BBR	SEA ↔ BA	ISL	N.A.	N.A.	136.3Mbps	0.97×
CUBIC	SEA ↔ BA	ISL	N.A.	N.A.	139.9Mbps	1.0×

Table I: Evaluation result of each test case.

Two Concurrent SATCP Flows		
Inhibition Duration	SATCP Flow 1 Avg. Throughput	SATCP Flow 2 Avg. Throughput
0.3s	47.9Mbps	50.0Mbps
0.9s	67.9Mbps	68.5Mbps
1.3s	69.9Mbps	68.7Mbps
2.3s	68.2Mbps	70.1Mbps
One SATCP Flow and One CUBIC Flow		
Inhibition Duration	SATCP Flow Avg. Throughput	CUBIC Flow Avg. Throughput
0.9s	111.9Mbps	17.2Mbps
2.3s	131.3Mbps	8.4Mbps

Table II: Throughput of concurrent flows.

2.3s inhibition duration, the highest level of aggressiveness, CUBIC still achieves a decent 8.4Mbps throughput. Note CUBIC throughput is already capped at around 34.3Mbps due to its unnecessary CWND reductions, as shown in Table. I. Concurrent flows over less dynamic ISLs show even better results.

Indeed, while SATCP momentarily increases its aggressiveness at critical points to avoid unnecessary CWND reduction, the altruism inherited from CUBIC TCP is still well maintained. Two SATCP flows equally divide the bandwidth, and SATCP also gives up a decent amount of throughput for the concurrent CUBIC flow.

VII. Conclusion and Future Work

This paper makes two major contributions: LeoEM, a high-fidelity and highly configurable LEO satellite network emulator, and SATCP, a cross-layer mechanism for improving TCP performance under highly dynamic satellite networks. Using LeoEM, we validate SATCP on an emulated Starlink network, and observe superior performance gains in comparison with the widely deployed loss-based CUBIC, and the state-of-the-art BBR which combines bandwidth and delay metrics. Our experiments adopted standard shortest-path algorithm to select the routes for end-to-end paths. Certain routing protocols may incur less frequent routing table updates, albeit at the cost of suboptimal routes. In our future work, we will explore such routing strategies and evaluate the tradeoffs in depth.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments. This work is partially supported by the NSF under Grants CNS-1901048, CNS-1925767, and CNS-2128588.

References

[1] SpaceX, "Starlink," mar 2022, https://www.starlink.com/.

[2] Amazon, "Project kuiper," mar 2022, https://www.aboutamazon.com/news/tag/project-kuiper.

[3] OneWeb, "Oneweb," mar 2022, https://oneweb.net/.

[4] S. Cakaj, "The parameters comparison of the "starlink" leo satellites constellation for different orbital shells," *Frontiers in Communications and Networks*, vol. 2, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/frcmn.2021.643095

[5] P. Du, X. Li, Y. Lu, and M. Gerla, "Multipath tcp over leo satellite networks," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2015, pp. 1–6.

[6] L. A. Sanchez, M. Allman, and D. D. Glover, "Enhancing TCP Over Satellite Channels using Standard Mechanisms," RFC 2488, Jan. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2488

[7] J. Griner, D. D. Glover, S. Dawkins, D. J. D. Touch, M. Allman, D. Tran, H. Kruse, J. Heidemann, J. Semke, S. Ostermann, K. Scott, and T. Henderson, "Ongoing TCP Research Related to Satellites," RFC 2760, Feb. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2760

[8] X. Xie, X. Zhang, and S. Zhu, "Accelerating mobile web loading using cellular link information," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2017.

[9] Dave, "How accurate can a satellite's orbit be?" Space Stack Exchange, https://space.stackexchange.com/questions/31378/how-accurate-can-a-satellites-orbit-be.

[10] H. Peng and X. Bai, "Improving orbit prediction accuracy through supervised machine learning," *Advances in Space Research*, vol. 61, no. 10, pp. 2628–2646, may 2018. [Online]. Available: https://doi.org/10.1016%2Fj.asr.2018.03.001

[11] J. Najder and K. Sośnica, "Quality of orbit predictions for satellites tracked by slr stations," *Remote Sensing*, vol. 13, no. 7, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/7/1377

[12] Z. Lai, H. Li, and J. Li, "Starperf: Characterizing network performance for emerging mega-constellations," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, 2020, pp. 1–11.

[13] 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

[14] Y. Hauri, D. Bhattacherjee, M. Grossmann, and A. Singla, ""internet from space" without inter-satellite links," in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, ser. HotNets '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 205–211. [Online]. Available: https://doi.org/10.1145/3422604.3425938

[15] M. Handley, "Delay is not an option: Low latency routing in space," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, ser. HotNets '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 85–91. [Online]. Available: https://doi.org/10.1145/3286062.3286075

[16] S. Rican8964, "Anyone else's latency look like this?" 2022, https://www.reddit.com/r/Starlink/comments/takytm/anyone_elses_latency_look_like_this/.

[17] eprosenx, "Starlink latency and packet loss graphs from smokeping," 2021, https://www.reddit.com/r/Starlink/comments/k7mj2x/starlink_latency_and_packet_loss_graphs_from/.

[18] Vertigo103, "After several updates to starlink i'm down to 1 minute of obstruction per day." 2021, https://www.reddit.com/r/Starlink/comments/mav3h5/after_several_updates_to_starlink_im_down_to_1/.

[19] P. K. Chowdhury, M. Atiquzzaman, and W. Ivancic, "Handover schemes in satellite networks: state-of-the-art and future research directions," *IEEE Communications Surveys Tutorials*, vol. 8, no. 4, pp. 2–14, 2006.

[20] J. Li, K. Xue, J. Liu, and Y. Zhang, "A user-centric handover scheme for ultra-dense leo satellite networks," *IEEE Wireless Communications Letters*, vol. 9, no. 11, pp. 1904–1908, 2020.

[21] H.-L. Maattanen, B. Hofstrom, S. Euler, J. Sedin, X. Lin, O. Liberg, G. Masini, and M. Israelsson, "5g nr communication over geo or leo satellite systems: 3gpp ran higher layer standardization aspects," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[22] SpaceX, "Spacex non-geostationary satellite system," 2016, https://fcc.report/IBFS/SAT-LOA-20161115-00118/1158350.pdf.

[23] A. Abdelrahman, M. Abdalla, B. Ali, V. Prakash, and R. Sabudin, "Improving the performance of tcp in leo satellite environment," *Information Technology Journal*, 2002.

[24] M. K. Mukerjee, C. Canel, W. Wang, D. Kim, S. Seshan, and A. C. Snoeren, "Adapting tcp for reconfigurable datacenter networks," ser. NSDI'20. USA: USENIX Association, 2020, p. 651–666.

[25] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "piStream: Physical Layer Informed Adaptive Video Streaming over LTE," in *Proceedings of ACM MobiCom*, 2015.

[26] CesiumJS, "Cesiumjs," 2022, https://cesium.com/platform/cesiumjs/.

[27] SpaceX, "Spacex non-geostationary satellite system," 2019, https://fcc.report/IBFS/SAT-MOD-20190830-00087/1877671.

[28] ——, "Starlink services - spacex," 2021, https://www.globalsecurity.org/space/systems/starlink.htm.

[29] ——, "Starlink maritime," jan 2023, https://www.starlink.com/maritime.

[30] ——, "Starlink aviation," jan 2023, https://www.starlink.com/aviation.

[31] walrus01. (2020, July) The first generations of starlink satellites will serve as bent pipes. [Online]. Available: https://news.ycombinator.com/item?id=22447873

[32] E. Ralph. (2020, July) Spacex starlink 'space lasers' successfully tested in orbit for the first time. [Online]. Available: https://www.teslarati.com/spacex-starlink-space-lasers-first-orbital-test/

[33] D. Bhattacherjee and S. Kassing, "Rethinking networking for an "internet from space"," https://wisr.cs.wisc.edu/ppt/satnets.pdf.

[34] ZealousidealDouble8, "Maybe dumb question - what will we see on traceroute?" 2020, https://www.reddit.com/r/Starlink/comments/hbicej/maybe_dumb_question_what_will_we_see_on_traceroute/.

[35] speedypoultry, "To beta testers: Has any beta tester out there run trace routes to determine if they are always routing through the same ground station?" 2021, https://www.reddit.com/r/Starlink/comments/kclydi/to_beta_testers_has_any_beta_tester_out_there_run/.

[36] Mininet, "Mininet: An instant virtual network on your laptop (or other pc)," 2022, http://mininet.org/.

[37] T. Kelso, "Supplemental two-line element sets," 2022, https://www.celestrak.com/NORAD/elements/supplemental/.

[38] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, p. 277–288, nov 1984. [Online]. Available: https://doi.org/10.1145/357401.357402

[39] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *ACM Queue*, 2016.

[40] "Live starlink satellite and coverage map," 2022, https://satellitemap.space/#.

[41] WonderNetwork, "Global ping statistics - wondernetwork," 2022, https://wondernetwork.com/pings.