SPACHE: Accelerating Ubiquitous Web Browsing via Schedule-Driven Space Caching

Qi Zhang
Zhongguancun Laboratory
Beijing, China
zhangqi@zgclab.edu.cn

Qian Wu
Tsinghua University
Zhongguancun Laboratory
Beijing, China
wuqian@cernet.edu.cn

Zeqi Lai
Tsinghua University
Zhongguancun Laboratory
Beijing, China
zeqilai@tsinghua.edu.cn

Jihao Li
Zhongguancun Laboratory
Beijing, China
lijh@zgclab.edu.cn

Hewu Li
Tsinghua University
Zhongguancun Laboratory
Beijing, China
lihewu@cernet.edu.cn

Yuyu Liu
Tsinghua University
Beijing, China
liuyy23@mails.tsinghua.edu.cn

Yuanjie Li
Tsinghua University
Zhongguancun Laboratory
Beijing, China
yuanjiel@tsinghua.edu.cn

Jun Liu
Tsinghua University
Zhongguancun Laboratory
Beijing, China
juneliu@tsinghua.edu.cn

Abstract

In this paper, we perform a systematic study to explore a pivotal problem facing the web community: *is current distributed web cache ready for future satellite Internet?* First, through a worldwide performance measurement based on the RIPE Atlas platform and Starlink, the largest low-earth orbit (LEO) satellite network (LSN) today, we identify that the uneven deployment of current distributed cache servers, inter-ISP meandering routes and the last-mile congestion on LEO links jointly prevent existing terrestrial web cache from providing low-latency web access for users in emerging LSNs. Second, we propose SPACHE[1], a novel web caching system which addresses the limitations of existing ground-only cache by exploiting a bold idea: integrating web cache into LEO satellites to achieve ubiquitous and low-latency web services. Specifically, SPACHE leverages a key feature of LSNs called *communication schedule* to efficiently prefetch web contents on satellites, and adopts a schedule-driven partitioning strategy to avoid cache pollution involved by LEO mobility. Finally, we implement a prototype of SPACHE, and evaluate it based on real-world HTTP traces and data-driven LSN simulation. Extensive evaluations demonstrate that as compared to existing distributed caching solutions, SPACHE can improve cache hit ratio by 19.8% on average, reduce latency by up to 17.7%, and maintains consistently low web browsing latency for global LSN users.

CCS Concepts

• **Networks** → **Network measurement**; **Cloud computing**.

Keywords

LEO Satellite Networks; CDN Performance; Caching.

[1]SPACHE indicates the integration of space and cache.

WWW '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714789

ACM Reference Format:
Qi Zhang, Qian Wu, Zeqi Lai, Jihao Li, Hewu Li, Yuyu Liu, Yuanjie Li, and Jun Liu. 2025. SPACHE: Accelerating Ubiquitous Web Browsing via Schedule-Driven Space Caching. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3696410.3714789

1 Introduction

User-perceived latency is recognized as the core performance issue in web browsing — the dominant Internet application today. *Distributed web caching* is a fundamental technique for reducing interactive latency and improving web experience. Leveraging geo-distributed cache servers by cloud platforms (*e.g.,* Amazon CloudFront [16]) and Internet service providers (ISP, *e.g.,* Verizon[17]), web content providers can push their web replicas to network edges close to users. User's HTTP request can get a faster response from the cache servers instead of from the source. Hence, distributed web cache is widely deployed and used in existing terrestrial networks, including wired networks [47, 51] and WiFi/cellular networks [54].

Parallel to the swift progression of web technology, Internet infrastructure also advances at an unprecedented pace. The recent satellite Internet constellations, such as Starlink [63], OneWeb [2] and Amazon Kuiper [1], are deploying thousands of low-earth orbit (LEO) broadband satellites to revolutionize the way people access the Internet. For example, as of Jan. 2025, SpaceX's Starlink, the largest operational LEO satellite network (LSN) today, has provided Internet services for more than 4.6 million global subscribers spanning all seven continents and oceans [64], especially for those in remote and rural areas [31, 64]. Essentially, an LSN has many unique characteristics different from other forms of terrestrial networks, such as the infrastructure-level dynamics due to the high-speed movement of LEO satellites [18]. Given that LSNs extend the global reach of web services, it is imperative for both web content providers and satellite ISPs to understand the performance of existing web infrastructures for LSN users.

Facing the future trend of *web-LSN integration*, this paper explores a pivotal problem: *is current distributed web cache ready for future satellite Internet?* We carry out our study in three steps.

First, we conduct a global measurement study to understand the real-world web performance through the lens of LSN users.

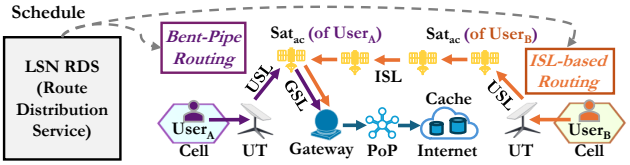
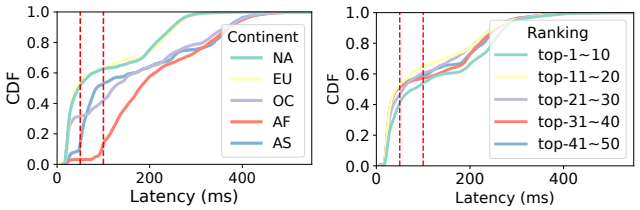


Figure 1: Architecture of existing LSNs (*e.g.*, Starlink).



(a) RTTs grouped by continents. **(b) RTTs grouped by website ranks.**

Figure 2: Web cache RTTs perceived by LSN users. NA: North America. EU: Europe. OC: Oceania. AF: Africa. AS: Asia. 50 ms / 100 ms are marked with red dashed lines.

Our analysis based on data collected from 76 RIPE Atlas [3] probes equipped with Starlink terminals across five continents reveals that: from a worldwide aspect, distributed web caching has not kept pace with the expansion of network boundaries, and there is a large proportion of LSN users suffering from high user-to-cache RTTs, causing high web browsing latency (*e.g.,* about 40.79% LSN users experience RTTs higher than 100 milliseconds (ms) on average when visiting Alexa Top-50 websites). On our further investigation, we identify that the uneven deployment of current web cache servers, inter-ISP meandering routes and the last-mile congestion on LEO links are the culprits preventing existing terrestrial web cache from providing ubiquitous and low-latency web access as expected.

Second, to address the limitations of existing terrestrial web cache, we explore the feasibility and effectiveness of a bold idea: *integrating LEO satellites and web caching systems to directly provide cache services from space, and thus accelerate ubiquitous web browsing*. To this end, we present Spache, a novel space web caching system for terrestrial users. Practically deploying web cache in LEO satellites needs to cope with two specific challenges involved by the unique LEO mobility: (i) since the LEO satellite cache assigned to users dynamically changes, it is challenging to efficiently warm space cache to improve the hit rate; (ii) because the serving region of an LEO satellite cache continuously changes, existing cache replacement algorithms (*e.g.,* Least Frequent Used, LFU) are vulnerable to cache pollution which leads to low hit rate during region switches. Spache addresses these challenges by exploiting a key feature of emerging LSNs called *communication schedule*, which is pre-calculated by satellite operators in advance, and determines the communication timing between LSN entities (*e.g.,* user terminals, satellites, or ground gateways). Concretely, Spache adopts a schedule-driven prefetching mechanism to efficiently warm cache in space, and leverages a cache partitioning strategy to mitigate cache pollution and increase cache hit ratio.

Finally, we implement a prototype of Spache, and further combine the prototype with a recent LSN simulator, real-world web traces and real constellation information to jointly build a data-driven experimental environment for evaluation. Extensive experiments demonstrate that by schedule-driven cache prefetching and partitioning, Spache can improve cache hit ratio by 19.8% on average, reduce latency by up to 17.7% as compared to existing solutions, and attain low web browsing latency in different continents.

The main contributions of this paper can be summarized as follows: (i) through the lens of geo-distributed LSN users, we identify the latency issue of today's terrestrial web caching systems; (ii) to overcome the constraints of terrestrial web caching, we propose Spache, a novel system designed to deploy and manage web caches on LEO satellites; (iii) combining our Spache prototype and real-world trace-driven LSN simulation, we demonstrate Spache can effectively accelerate ubiquitous web browsing for terrestrial users.

The measurement dataset is published to foster further study [62].

2 Background and Motivation

2.1 How LSN Users Access Current Web Cache?

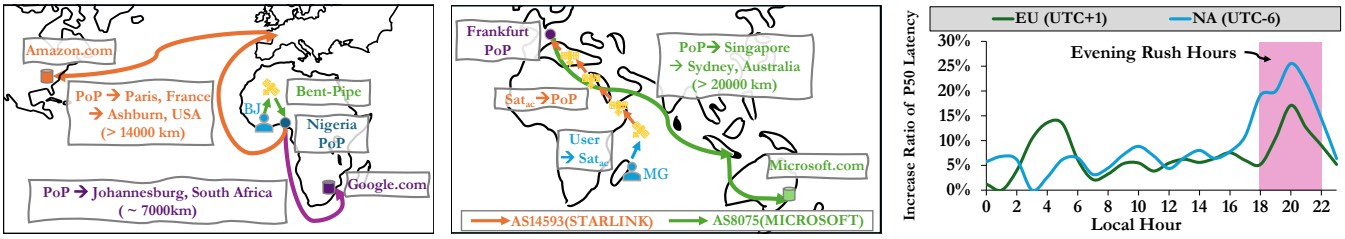
LSN architecture. LSNs provide Internet services to users via satellite constellations which consist of thousands of LEO satellites positioned 300-2000 km above the earth surface. Figure 1 illustrates a typical LSN architecture used by current satellite network operators like SpaceX. Users connect their equipments (*e.g.,* laptops or smartphones) to a satellite user terminal (UT, *e.g.,* Starlink's dish), which then dynamically connects to an access satellite (Sat_{ac}) through UT-to-satellite links (USLs). Satellites can inter-connect via inter-satellite links (ISLs). Essentially, in the current networking architecture, an LSN works as an *access network* of the terrestrial Internet. No matter where the user is, user traffic should be aggregated at a certain ground gateway through ground-to-satellite links (GSLs), and then be exchanged with the Internet through points of presence (PoPs). In practice, global users are mapped into geography-based cells for access management. The route through which users in a specific cell access the Internet is calculated in advance by the route distribution service (RDS) [14] and disseminated as schedules to the satellite routers along the path. If a user is close to an available gateway, traffic from the user can be forwarded to the gateway via transparent forwarding (*i.e.,* bent-pipe routing [43]). Otherwise, user traffic needs to be forwarded to the gateway via multi-hop satellite forwarding (*i.e.,* ISL-based routing [41]).

Web cache servers. To provide scalable and low-latency web services, Internet content providers (ICPs) usually leverage geo-distributed web cache servers (*e.g.,* Netflix's Open Connect [49] and Google's Global Cache [28]) to serve web access requests closer to users. Depending on providers' strategies, they may be set up at different scales, *e.g.,* global or regional. Since today's LSN is mainly used as an access network for the Internet, when a user accesses a website through an LSN, its HTTP(s) request is firstly forwarded to an associated PoP and then redirected—often via DNS- or anycast-based load balancers—to a nearby terrestrial web cache.

2.2 Limitations of Terrestrial Web Cache

Ideally, by pushing contents to edges close to end users, distributed web caches are expected to provide low-latency web access for users anywhere. However, through a global measurement based on a number of RIPE Atlas probes [3] equipped with Starlink terminals, we identify three limitations of existing terrestrial web cache.

Methodology and observations. To quantitatively understand the web performance perceived by LSN users, we use the `ping` and `traceroute` tools to measure the user-to-cache RTT of accessing



(a) Limited coverage of web cache. (b) Routing detours between satellite/terrestrial ISPs. (c) Adverse impacts of LSN link congestion.

Figure 3: Dissecting the primary sources of latency in web caches through the lens of LSN users.

Alexa's Top-50 websites [59] from a number of Starlink terminals in different regions around the world. All these top-50 websites use CDNs to push their web contents to cache nodes close to users to speed up content access [44]. To build our geo-distributed LSN vantage points, we recruit 76 Starlink probes in 5 continents based on the RIPE Atlas platform [3].

Figure 2 plots the user-to-cache RTT between: (i) our vantage points in different continents, and (ii) the web cache server assigned by these top-50 websites, grouped by different continents (Figure 2a) and by website ranking (Figure 2b). Note that when a user visits a website via the URL, the browser first queries DNS to obtain the IP address of the assigned web cache server. On each vantage point, we first resolve the IP address of the web cache server of each website, and then `ping` the corresponding cache server every 4 minutes in one day to obtain the RTT results. We observe that although LSNs like Starlink enable Internet access anywhere, from a global perspective, LSN users still experience long user-to-cache RTTs, *e.g.,* for 40.79% of all the vantage points, 48.7%/56% of the vantage points in NA/EU and all vantage points in OC/AF/AS, their user-to-cache RTT to all the websites is higher than 100ms on average. To uncover the root causes of the high RTTs perceived by LSN users, we use `traceroute` and IP geo-location service [33] to inspect each hop of the user-to-cache path. Concretely, we identify three limitations of existing ground-based web cache as follows.

Limitation (i): limited coverage of web cache in remote areas. Although LSN with ubiquitous connectivity can connect people in remote and rural areas, the deployment of web cache servers has not kept pace with LSN development. As a result, the coverage of existing terrestrial web cache is still limited in remote or underdeveloped areas. For example, Figure 3a plots a concrete case where the high user-to-cache RTT is caused by the long distance between the user and the assigned cache servers. A Starlink user located in Benin (BJ) wants to visit `google.com` and `amazon.com`. In particular, the load balancers of these websites assign web cache servers in Ashburn (US) and Johannesburg (ZA) which are far away from BJ to the user. Based on our `traceroute` analysis, we find that traffic from the users has to aggregate at a PoP at Nigeria, and then it takes tortuous terrestrial routes to reach the destination cache servers. As a result, the user experiences 97.7/224.2 ms user-to-cache RTTs on average for visiting Google and Amazon websites respectively.

Limitation (ii): routing detours between satellite and terrestrial ISPs. Today's web cache servers are typically deployed by cloud providers (*e.g.,* CloudFront [16], Microsoft Azure [7]) or terrestrial ISPs (*e.g.,* Verizon [17]). Thus, an LSN user needs to cross multiple autonomous systems (ASes) to access the cache. Note that in the current LSN architecture, traffic between the satellite and non-satellite ISPs must be aggregated and exchanged at specific ground gateways. We identify that the insufficient gateway deployment can lead to long routing detours between satellite and terrestrial ISPs, causing high user-to-cache RTTs. Figure 3b plots a concrete case where a Starlink user located in Madagascar (MG) visits a web cache in Sydney assigned by `microsoft.com`. Based on our `traceroute`-based investigation, we find that the user's HTTP requests have to go through a multi-hop satellite path to reach a PoP at Frankfurt, and then be forwarded to the assigned cache in Sydney via a terrestrial detour. As a result, the entire end-to-end path passes through the Starlink ISP and other terrestrial ISPs, causing high RTTs up to 510 ms.

Limitation (iii): The adverse impacts of LSN link congestion on web cache effectiveness. In addition, even in developed areas with sufficient cache server and gateway deployments, we observe that the effectiveness of web cache could still be constrained by the network congestion of the *LSN links*. Figure 3c plots the RTT variations versus local hour measured from our vantage points in NA and EU. We observe that the user-to-cache RTTs fluctuate over time, and reach the highest value in the peak hours (evening rush hours) of the day. This suggests that even when the network path between an LSN user and the assigned web cache is of low baseline delay and dominated by LSN links, the network congestion in LSN links [22, 25] can undermine the effectiveness of web cache.

3 SPACHE Overview

To address the above limitations of existing terrestrial web cache and facilitate low-latency web access anywhere, we present SPACHE, a novel cache management system for LSNs. In this section, we introduce the basic idea and overview of SPACHE.

3.1 A Bold Idea: Web Cache in Space

Fundamentally, SPACHE exploits a bold idea: integrating web cache into LEO satellites to serve terrestrial users, which enables a series of benefits as highlighted below.

- **(i) Web caching in space enables short-distance cache access from anywhere.** LEO constellations provide ubiquitous and continuous satellite connectivity for terrestrial users. In other words, for a terrestrial user, at any location and at any time, there will be at least one LEO satellite close to the user that can provide LSN service. In principle, caching web content at these *"LEO edges"* could enable short-distance web access globally.
- **(ii) Web caching in space can avoid meandering routes between different ISPs' ASes.** Different from existing terrestrial

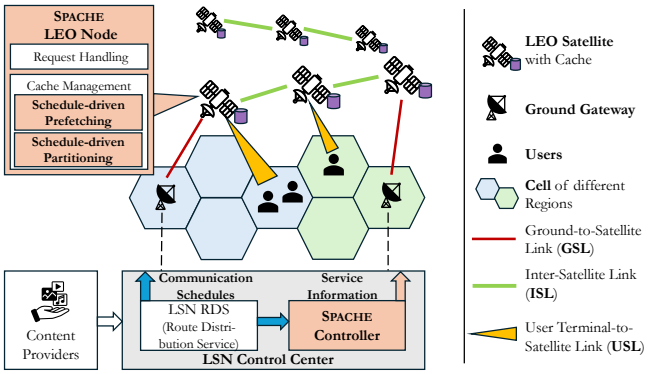


Figure 4: Overview of SPACHE architecture.

ISPs which mainly operate their local network infrastructures and ASes to provide *regional* services, satellite ISPs can provide *global* network services via one AS. For example, the current Starlink mainly uses AS14593 to serve its global users. In other words, directly caching web contents on satellites can avoid the potential meandering routes to other ASes, enabling web content service directly from the "nearest" AS to the users.

- **(iii) Web caching in space can alleviate last-mile congestion.** Ground-only web caching results in web traffic aggregation at gateways and PoPs, which have been reported as bottlenecks in LSNs [18, 53, 65]. With LEO caching, content can be served directly from access satellites when possible. This approach reduces traffic convergence at gateways and PoPs, thereby alleviating congestion on last-mile links that degrade web service quality.

As on-board storage capacity continues to advance [30, 60], the deployment of high-volume caches on LEO satellites is becoming a practical reality. A detailed discussion on the technical feasibility of space caching is provided in Appendix B.

3.2 SPACHE Architecture

Core components. Figure 4 plots the overview of SPACHE, which consists of two core components: *(i) a* SPACHE *Controller* on the satellite ISP's control center to manage and distribute web contents to LEO satellites; and *(ii) a number of distributed* SPACHE *Caches* on satellites. SPACHE exploits the evolving on-board storage of LEO satellites, and equips each LEO satellite of the LSNs with a cache storage to work as an LEO cache node. Each LEO cache node is responsible for serving the requests from the regions it covers.

Baseline workflow. In practice, content providers leverage SPACHE to accelerate their web access as follows: *(i) Content Registration.* A content provider registers at SPACHE Controller with its basic information, *e.g.,* domain names and target service regions. Moreover, the content provider decides what web contents should be cached by SPACHE; *(ii) Cache Warming.* Since LEO satellites are equipped with web caches, when an LSN user visits a website, the cacheable contents (*e.g.,* defined by the content provider) are cached in the access satellite (*e.g.,* Sat_{ac} in Figure 1). For example, for the first time a web page is fetched from the original website, this page is cached on the access satellite, and then be used to serve all users covered by the LEO satellite. In this phase, the satellite cache is filling up with data based on user requests; *(iii) Cache Replacement.* Because a satellite cache has limited volume, SPACHE needs to decide which content needs to be replaced when a new content comes in. Thus, at runtime SPACHE follows a certain *cache replacement* algorithm to update each satellite cache in space.

3.3 Challenges of Practicalizing SPACHE

Realizing the potential of SPACHE in practice still needs to cope with two critical challenges involved by the unique LEO mobility.

"Fast cooling" in space web cache. Fundamentally, LEO satellites are moving at a high velocity related to the earth surface. Due to the LEO mobility, an access satellite is only visible to a terrestrial user for up to several minutes depending on the orbit altitude. Therefore, a terrestrial user has to frequently switch its space cache during a web session, which imposes a significant challenge on cache warming and maintaining high cache hit rate: when a space cache has just been warmed by user requests, the user has to switch to a new and cold cache that contains little to no required data.

Cache pollution by existing replacement algorithms. There are a number of classic cache replacement algorithms [55] that are widely used in existing cache systems (*e.g.,* CDNs). In theory, the most efficient replacement algorithm would be to discard information which would not be needed for the longest time, which is known as Belady's optimal algorithm [9]. In real systems, practical algorithms like LRU [61] and LFU [12] choose the least recently used or the least frequently used item as the one that "might not be used for the longest time". However, as an LEO cache node continuously moves and its serving regions frequently switch between various regions, these algorithms tend to misestimate the value of contents and lead to unexpected cache pollution, *e.g.,* over-estimating the value of a content from previously served regions and keeping it in cache, which is useless in the current region. Cache pollution may also prevent the popular contents of the upcoming regions from staying in the cache, as we will observe in §5.

SPACHE addresses the above challenges by exploiting an important feature called "communication schedule" in LSNs, and adopting a schedule-driven cache management described as follows.

4 Schedule-Driven Cache Management

4.1 Communication Schedule in LSNs

Due to the inherent LEO mobility, frequent handovers and connectivity updates are inevitable between LEO satellites and terrestrial entities (*e.g.,* UTs and gateways in Figure 1). In order to manage these frequent handovers and ensure seamless connectivity for terrestrial users located in different cells, satellite operators (*e.g.,* SpaceX) leverage a global scheduler to make *communication schedules* for their satellites. For example, according to SpaceX's official documents [14, 23], in current Starlink network, a *communication schedule* indicates which entities (*e.g.,* a satellite, a UT or a gateway) should communicate with each other and when. In other words, a schedule determines which satellite the UT in each cell should connect to at different time, *i.e.,* it determines the time-varying access satellites (Sat_{ac}). In practice, the satellite operator's global scheduler calculates the communication schedule of each LEO satellite in the near future, and then distributes the schedules to the corresponding satellite through the Route Distribution Service (RDS) [14]. Thus, satellite operators know: (i) the schedule of each satellite; (ii) which satellite the UT in different cells will connect to and switch to in the

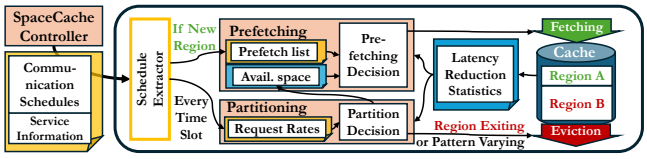


Figure 5: Schedule-driven cache management.

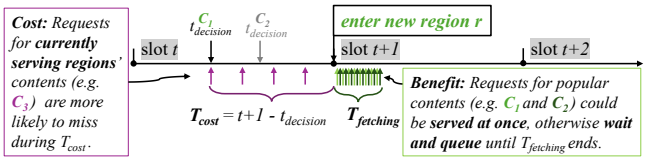


Figure 6: Prefetching decision at $t_{decision}$.

near future. Based on the communication schedules, the SPACHE controller can monitor fine-grained cache service information at the cell level (*e.g.,* request rates of each cell). The SPACHE controller attaches this service information to the communication schedules, sends them to the LEO cache nodes serving the corresponding cells, and receives the feedback of actual value from the nodes.

4.2 Schedule-Driven Cache Prefetching

Following the time-slotted manner of communication schedules [14, 23], SPACHE operates in a time-slotted way with each slot of duration Δ_t. To prevent UTs from frequently accessing a cold space cache and suffering from a low hit rate, SPACHE uses a *schedule-driven prefetching* mechanism to pre-load the contents that may be accessed into the space cache before the UT switches to it. In particular, to ensure a high cache hit rate and avoid wasting precious on-orbit storage resources, SPACHE leverages the schedule information to guide prefetching web contents in space as follows.

What and where to prefetch? The controller monitors the schedules and when it finds that an LEO cache node is about to serve a new region r, it provides the prefetching list $\mathbf{\Phi_r}$ to the node in addition to the communication schedule. $\mathbf{\Phi_r}$ is constructed based on content popularity and performance logs that the controller keeps monitoring, including the top k popular contents such that their cumulative popularity exceeding a threshold β:

$$|\mathbf{\Phi_r}| = argmin_k \sum_{i=1}^{k} Pop_i \geq \beta, \tag{1}$$

where Pop_i represents the popularity of C_i, the content indexed by i, estimated with the ratio of the request number of C_i to the total request number of the region r.

For each content, a fetching source list is included, and each source is associated with a fetching latency $T_{fetching}$ representing the time for the node to obtain the content from that source.

When to prefetch? Upon receiving $\mathbf{\Phi_r}$, the LEO cache node continuously evaluates whether to prefetch the remaining most popular content C_{max} in the list. When C_{max} is completely fetched, it is removed from $\mathbf{\Phi_r}$. This process repeats until the node enters r or $\mathbf{\Phi_r}$ is empty. When $\mathbf{\Phi_r}$ is empty, the node notifies the controller and could be a fetching source for other nodes.

At each decision time $t_{decision}$, this decision is made by comparing the benefit and cost of prefetching in terms of response latency (see Figure 6). If the benefit exceeds the cost, the node fetches C_{max} from the corresponding source with the lowest $T_{fetching}$. Prefetching C_{max} allows immediate response of requests upon entering r, avoiding fetching delays. The benefit is calculated as:

$$\text{LatencyBenefit}_{C_{max}} = \frac{T_{fetching}}{2} \times \frac{T_{fetching}}{\Delta_t} \times \sum_{c \in \mathbf{c^r}} RN_c \times Pop_{max}, \tag{2}$$

where $\mathbf{c^r}$ represents the set of cells in r, $\sum_{c \in \mathbf{c^r}} RN_c$ is the total request number within r for the next time slot and denoted as RR_r.

Prefetching consumes cache storage, potentially evicting content valuable for other regions. This cost is quantified as:

$$\text{LatencyCost}_{C_{max}} = \frac{T_{cost}}{\Delta_t} \times SLR_{r_{evict}}, \tag{3}$$

where T_{cost} is the duration of the cost impact, r_{evict} is the region where the evicted item is from and is determined by schedule-driven partitioning strategy (§4.3), based on the schedule-based average latency reduction (SLR) it maintains for each region.

The above prefetching approach achieves efficiency since: (i) it performs prefetching operations just on time, as the schedule of future service regions allows it to make decisions and fetch contents in advance, and it continuously evaluates the cost involved by prefetching; (ii) it avoids aggressive content pre-loading, as it successively fetches the contents and does not incur bursty traffic.

4.3 Schedule-Driven Cache Partitioning

To avoid the cache pollution between regions involved by LEO mobility, SPACHE adopts a *schedule-driven cache partitioning* strategy.

Cache partitioning is a technique where the total cache space is divided into sub-caches, each dedicated to a specific group of content. This ensures that content requests from one group do not cause the eviction of frequently accessed content from another group, *i.e.,* avoiding cache pollution where valuable content is replaced by less useful content. Cache partitioning was originally designed in CPU cache to cope with cache pollution due to different programs' disparate cache demands and performance sensitivities [46, 58]. In recent years, cache partitioning has been widely adopted in terrestrial CDNs [68] to enhance existing cache replacement algorithms.

New challenges of applying cache partitioning in LEO cache. In traditional caching systems, partitioning can involve high computation overhead due to the much larger scale of content compared to the size of programs in CPU cache [46, 68]. Due to the rapid changes in service regions in LEO satellite caching, existing methods take additional time after a region switch to acquire new data and remove outdated content, which can still result in cache pollution.

Schedule-driven cache partitioning in LEO cache. To solve the above challenges, SPACHE employs a novel partitioning method leveraging the schedule information. At a high level, the entire cache storage is partitioned into multiple sub-caches, with each region served by its own sub-cache using a certain replacement algorithm. P_r^t denotes the space allocated to the sub-cache of region r during slot t. A request from a certain region for a particular content set is handled by the corresponding sub-cache.

First, a future-aware utility metric, *Schedule-based Average Latency Reduction* (SLR) for each region is proposed. For a request, its latency reduction is defined as the delay of fetching from the source minus the actual request delay. For the sub-cache of a region r in

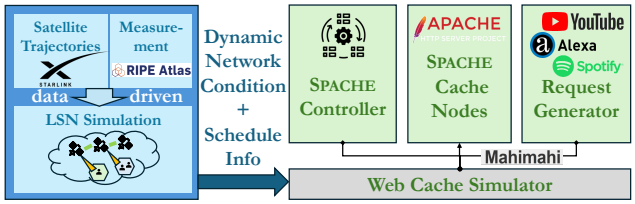


Figure 7: Key components of the experiment environment.

time slot t, LR_r^t sums the latency reductions of all requests. Further, an LEO cache node extracts the cells it will serve in time slot t from the schedules, and then estimates the request rates RR_r for region r in time slot t by summing the per-cell request rate. Collectively, SLR_r^t is calculated as:

$$SLR_r^t = \frac{LR_r^{t-1}}{P_r^{t-1}} \times \frac{RR_r^t}{RR_r^{t-1}}. \quad (4)$$

Then, at the beginning of each time slot, regions are ranked by SLR, and the sub-cache of the $(N-k+1)$-th region is reduced by $D\%$ and reallocated to the k-th region's sub-cache, where N is the number of regions served in this slot. Unlike existing static D settings [10, 68], SPACHE dynamically adjusts D based on the variation of the regions' request rate, to adapt to possible abrupt service region changes. Specifically, D is set in a range of $[D_{min}, D_{max}]$ linearly mapping with request rate variation Δ_{RR} of the regions. This adaptive mechanism enables rapid cache reallocation when a region exits abruptly, thereby minimizing cache pollution.

A list of abbreviations is given in Appendix A.

5 Evaluation

5.1 Experiment Setup

SPACHE prototype. We implement a prototype of SPACHE in around 1200 lines of C++ code. The SPACHE controller is composed of a communication schedule reader, a prefetching list generator, and a cell-level request rate monitor. The controller sends these information to the cache nodes and receives their update of content popularity and cell-level request rate. The cache nodes of SPACHE are implemented based on the `webcachesim` [61], a well-known research-grade simulator for web caching policies. We extend the original version of `webcachesim` by adding a schedule-driven partitioning module and a schedule-driven prefetching module to it. The default threshold β is 50%, D is set in a range of [3%, 20%] for our 15-second interval, and the maximum number of service regions that can be served by one satellite at the same time is 5.

LSN simulation. We built an LSN simulator based on the methodology introduced by [38] and the real satellite trajectories of Starlink [23, 27, 36]. The earth surface is divided into different regions and cells. Specifically, regions are partitioned based on ISO 3166-1 standard [32], and each region has its own content set. Cells are partitioned with a $1^\circ \times 1^\circ$ latitude-longitude grid. Each cell is located in a certain region. We simulate the Starlink's global scheduler [14, 66] which dynamically updates the connections between satellites and UTs in different cells with Δ_t of 15-second interval.

Web request generator. To evaluate SPACHE under geo-varying request patterns, we define the request *geo-diversity* as the proportion of request numbers to local content over the total request numbers for a region, and a content is considered *local* if it is not requested by other regions. A higher average geo-diversity indicates less overlap of popular contents across regions. We use two types of traces for our evaluation: (1) *Synthetic Trace*, which allows us to adjust the request geo-diversity and evaluate SPACHE under various geo-diversity levels. Each region has its own catalog of 1000 files, and the popularity distribution follows a Zipf law with α=1.1. (2) *Dataset-driven Trace*, which generates requests composed of *local popular content* and *global popular content* based on two datasets collected from YouTube (video content)[4] and Spotify (music content)[37]. The two datasets contain the list of Top-100 or Top-200 contents of each service region and their number of views. The YouTube dataset includes 137 service regions and the Spotify dataset includes 71 service regions, both covering 6 continents. The *local* popular contents of each region are extracted from the two dataset, with Alexa Top-50 websites as the *global* popular contents.

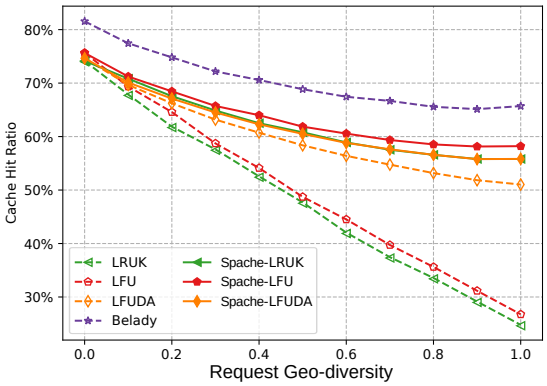


Figure 8: Improvement by SPACHE's schedule-driven partitioning under various request geo-diversity.

Comparisons. We compare SPACHE with: (i) ground-only web caching (denoted as GWC), which only uses terrestrial distributed cache to accelerate web browsing. Specifically, we simulate a GWC system based on the server distribution of Amazon CloudFront [16], a widely used commercial web caching service; (ii) baseline space caching (denoted as BSC), which deploys web cache servers on LEO satellites, but does not leverage schedule information for content prefetching and cache partitioning like SPACHE; (iii) StarFront [39], a recent content distribution system which dynamically builds a delivery tree to distribute contents on satellite edges.

Putting it together. Figure 7 shows how the above components are integrated to construct the experiment environment. First, based on the real Starlink trajectories, the LSN simulator calculates and schedules the time-varying connections between LEO satellites and terrestrial entities. Second, these time-vary network conditions are then used by `Mahimahi` [50] to mimic the dynamic LSN conditions between the web request generator and SPACHE prototype (or other comparisons). Finally, web requests are served by different caching mechanisms through the mimic LSN network conditions.

5.2 Hit Rate Improvement

Hit rate improved by SPACHE's schedule-driven partitioning. We first conduct a micro-benchmark under synthetic traces to evaluate the hit rate improvement made by SPACHE's cache partitioning on existing replacement algorithms: LFU [12], LFUDA [6] and

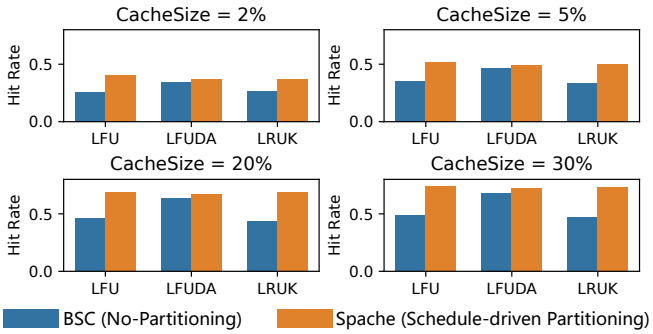
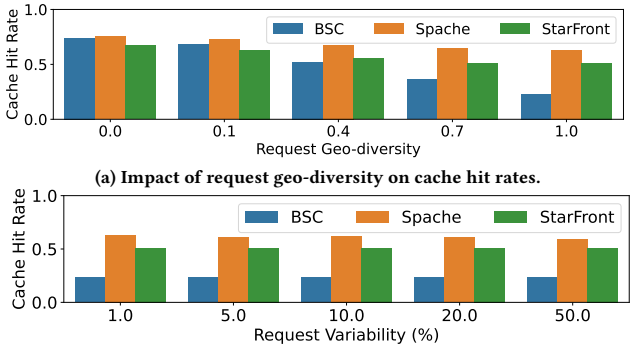


Figure 9: Improvement by Spache's schedule-driven partitioning under various cache capacity. A CacheSize of X% indicates the capacity to store X% of the content catalog for a given region.

LRUK [52]. Figure 8 shows the hit rate achieved by different replacement algorithms with or without schedule-driven cache partitioning under different geo-diversity. We also plot the results of Belady which indicates the theoretical optimal result. We observe that Spache can enhance existing replacement algorithms to achieve higher hit rate. The Spache-enabled improvement compared to the baseline LFU, LFUDA and LRUK is more significant when the geo-diversity increases as the baselines fail to cope with the high dynamics of LEO satellite cache, while Spache dynamically reallocates the cache storage to regions with higher demand and thus mitigates cache pollution. Further, Figure 9 shows the hit rate comparison with different on-satellite cache volumes, where Spache enhances existing replacement algorithms. The improvement on algorithms like LFU is more significant when the cache volume increases, as LFU fails to cope with cache pollution and utilize the cache storage effectively even when a larger capacity is available. By utilizing schedule information to partition the cache storage for contents popular in different regions, Spache enables adaptive and timely content allocation under various cache volumes.

Hit rate achieved by different caching systems. We then evaluate the hit rate improvement by different caching systems, as plotted in Figure 10. For both systems we use LFU as the replacement algorithm. We first compare the hit rate of the strategies under various geo-diversities. Figure 10a shows that Spache could achieve 57.68%/19.8% higher hit rate than BSC/StarFront on average. Moreover, as the geo-diversity increases, the improvement of Spache over BSC and StarFront becomes more significant. This is because under diverse demands from different regions, Spache could effectively utilize the schedule information and prefetch the popular contents of each region. Further, we evaluate the resilience of schedule-driven prefetching facing temporal variability of requests, which results in inaccurate popularity information contained in the schedules. To simulate temporal request variability, we randomly select a portion of the contents from the top half of the popularity ranking and swap their popularity with content from the bottom half. Figure 10b shows that under different temporal request variability, Spache could resiliently achieve better performance than the BSC and StarFront. We notice that BSC and StarFront are not sensitive to the request variability. Although Spache is slightly affected by the request variability, it still outperforms BSC/StarFront by 153.7%/15.6% at a high request variability of 50%.



(a) Impact of request geo-diversity on cache hit rates.

(b) Impact of temporal request variability on cache hit rates.

Figure 10: Hit rate achieved by different caching systems.

5.3 User-Perceived Web Experience

Next, we evaluate the user-perceived web experience by different caching systems, quantified by: (i) the RTTs between the user and the assigned cache server; and (ii) web page load time (PLT).

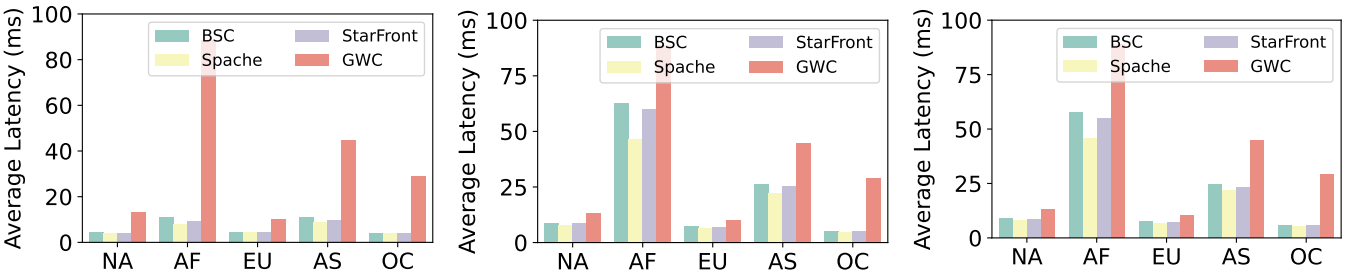
User-to-cache RTTs. Figure 11 shows the geo-distributed user-to-cache RTTs achieved for different datasets under dataset-driven traces. First, we observe that by deploying cache on LEO satellite constellations, Spache could achieve low user-to-cache RTTs at different continents for the global popular contents. Specifically, Spache improves the user-to-cache latency to the 50 websites by 76.9% on average compared to GWC, enabling user-to-cache latency of less than 20 ms in all continents and less than 10 ms in NA, EU and OC. The higher latency in AF and AS is mainly due to their smaller portion of global popular contents and higher latency to terrestrial caches. Compared to StarFront, Spache improves the user-to-cache latency by 6.4% on average and up to 17.7% at different continents.

Further, we look at the results obtained by the local popular contents of two representative applications, YouTube and Spotify. We observe that for local popular contents, the attained latency is higher than that of the global popular ones, as the local content occupies a smaller portion of the requests and the cache schemes aim to optimize the overall performance. Spache can achieve latency lower than 50ms for all datasets, and Spache's average improvement of local contents over StarFront (11.1%) is higher than that of the global popular ones (6.4%). This is because Spache can effectively utilize the schedule information and prefetch the popular content of each region, even when the content is not globally popular.

Web page load time. Further, we evaluate the user-perceived web page load time (PLT) achieved by different caching systems. In particular, we use `browsertime` [13], a web inspection tool to measure the PLT when browsing the Alexa Top-50 websites. Figure 12 shows the PLT results. We observe that Spache achieves lower PLT than others on average, and enables average PLT less than 5 seconds across all continents. The improvement over StarFront is 4.0% on average and up to 7.4%. On further analysis, we found that Spache achieves more significant improvement on DNS lookup time (Figure 13), which is a key factor affecting the overall PLT.

5.4 GSL Traffic Reduction

Finally, we evaluate the GSL traffic reduced by different caching systems. Figure 14 shows the normalized traffic volume consumed by different caching solutions. We observe that Spache can achieve



(a) Alexa Top-50 Websites. (b) YouTube Datasets. (c) Spotify Datasets.

Figure 11: User-to-cache RTT achieved by various solutions across different data traces.

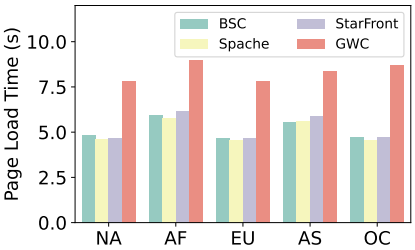


Figure 12: Page load time of Alexa Top-50 websites achieved by different solutions.

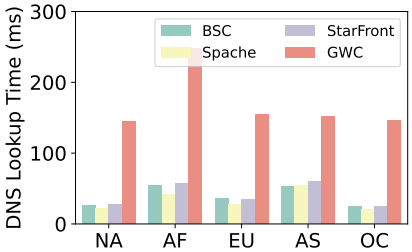


Figure 13: DNS lookup time of Alexa Top-50 websites by different solutions.

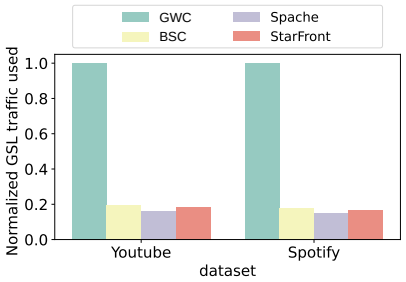


Figure 14: Normalized GSL traffic volume by different caching solutions.

83.65% and 84.77% lower GSL traffic consumption than GWC for YouTube and Spotify datasets. Spache also outperforms BSC and StarFront by 2.8% on average. The results under different datasets are similar, as the global popular content dominates. With lower traffic consumption, Spache could reduce the pressure on the satellite-terrestrial links and mitigate last-mile LEO link congestion.

6 Related Work

Optimizing content placement for distributed web cache. A considerable amount of research has been conducted on optimizing content placement for distributed web cache [15, 48, 51, 54]. For example, cache coordination [48] analyzed how the knowledge of requests in remote serving locations help make better decisions for cooperative caching and reduce serving costs for terrestrial CDNs. Authors in [54] explored the effectiveness of caching systems in 5G and wireless networks. Different from prior efforts for terrestrial caching systems, this paper focuses on content placement upon emerging LEO satellite networks, where the cache faces highly dynamic geo-distributed content request patterns. Spache solves the unique challenges of cache warming up and cache pollution by adopting a schedule-driven cache management.

Cache replacement algorithms. A lot of research efforts have been made on caching replacement algorithms, falling into heuristic-based ones [12, 55, 57] (account the contents value with factors including content freshness, access frequency, data size, *etc.* [8]) or learning-based ones [40, 61, 69]. While Spache does not focus on developing new replacement algorithms, Spache incorporates cache partitioning techniques to mitigate cache pollution issues in existing replacement algorithms for LSNs and addresses the new challenges posed by LEO mobility, where the cache infrastructure moves continuously at high speed.

In-orbit processing and storage. Many recent works developed advanced in-orbit processing and storage capabilities for LEO satellites [11, 19, 29, 42, 60]. Kodan [19] is a orbital edge computing system that maximizes the utility of satellite downlinks while mitigating the computational bottleneck. Phoenix [42] proposed a sunlight-aware space edge computing framework to optimize the energy efficiency of in-orbit processing. Authors in [60] reported a new storage media that is impervious to ionizing radiation and microgravity for long space missions. Based on the advanced hardware capabilities for in-orbit processing and storage, Spache integrates web cache and LEO satellites to accelerate ubiquitous web browsing.

7 Conclusion

This paper performs a systematic study to identify and address the limitations of existing distributed web cache service when serving global users through emerging LSNs. Our RIPE Atlas-based global measurement through Starlink reveals that the uneven deployment of current distributed cache servers, inter-ISP meandering routes and the last-mile congestion on LEO links prevent terrestrial web cache from providing low-latency web access for users in emerging LSNs. We present Spache, a novel web caching system which addresses the limitations of existing ground-only cache by integrating web cache into LEO satellites to achieve ubiquitous and low-latency web services. We implement a prototype of Spache, and evaluate it based on real-world HTTP traces and data-driven LSN simulation. Extensive evaluations show that Spache improves the cache hit ratio by an average of 19.8% reduces latency by up to 17.7% compared to existing approaches, and ensures consistently low user-to-cache latency across different global locations.

Acknowledgements. This research was supported by the National Key R&D Program of China (No. 2022YFB3105202) and National Natural Science Foundation of China (NSFC No. 62372259 and No. 62132004). Zeqi Lai and Hewu Li are the corresponding authors.

References

[1] 2024. Amazon Kuiper. https://www.geekwire.com/2019/amazon-project-kuiper-broadband-satellite/, accessed on 2025-02-05.

[2] 2024. OneWeb. https://oneweb.net/, accessed on 2025-02-05.

[3] 2024. RIPE Atlas Probes from Starlink (AS14593). https://atlas.ripe.net/probes/public?search=14593, accessed on 2025-02-05.

[4] 2024. YouTube Data API | Google Developers. https://developers.google.com/youtube/v3/, accessed on 2025-02-05.

[5] Akamai. 2024. Network Operator | Akamai Network Partnerships. https://www.akamai.com/solutions/industries/network-operator/akamai-network-partnerships, accessed on 2025-02-05.

[6] Martin Arlitt, Ludmila Cherkasova, John Dilley, Rich Friedrich, and Tai Jin. 2000. Evaluating content management techniques for web proxy caches. *ACM SIGMETRICS Performance Evaluation Review* 27, 4 (2000), 3–11.

[7] Azure Content Delivery Network | Microsoft Azure. 2024. https://azure.microsoft.com/en-us/products/cdn, accessed on 2025-02-05.

[8] Nathan Beckmann, Haoxian Chen, and Asaf Cidon. 2018. {LHD}: Improving cache hit rate by maximizing hit density. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 389–403.

[9] Laszlo A. Belady. 1966. A study of replacement algorithms for a virtual-storage computer. *IBM Systems journal* 5, 2 (1966), 78–101.

[10] Daniel S Berger, Benjamin Berg, Timothy Zhu, Siddhartha Sen, and Mor Harchol-Balter. 2018. RobinHood: Tail Latency Aware Caching-Dynamic Reallocation from Cache-Rich to Cache-Poor.. In *OSDI*. 195–212.

[11] Debopam Bhattacherjee, Simon Kassing, Melissa Licciardello, and Ankit Singla. 2020. In-orbit computing: An outlandish thought experiment?. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 197–204.

[12] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. 1999. Web caching and Zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, Vol. 1. IEEE, 126–134.

[13] Browsertime. 2024. https://www.sitespeed.io/documentation/browsertime/, accessed on 2025-02-05.

[14] Chen Chen, Pavel Chikulaev, Sergii Ziuzin, David Sacks, Peter J Worters, Darshan Purohit, Yashodhan Dandekar, Vladimir Skuratovich, Andrei Pushkin, Phillip E Barber, et al. 2023. Low latency schedule-driven handovers. US Patent 11,729,684.

[15] Fangfei Chen, Katherine Guo, John Lin, and Thomas La Porta. 2012. Intra-cloud lightning: Building CDNs in the cloud. In *2012 Proceedings IEEE INFOCOM*. IEEE, 433–441.

[16] Content Delivery Network - Amazon CloudFront - AWS. 2024. https://aws.amazon.com/cloudfront/, accessed on 2025-02-05.

[17] Content Delivery Network (CDN) Services | Verizon Business. 2024. https://www.verizon.com/business/products/security/web-security/web-acceleration-cdn/, accessed on 2025-02-05.

[18] Inigo Del Portillo, Bruce G Cameron, and Edward F Crawley. 2019. A technical comparison of three low earth orbit satellite constellation systems to provide global broadband. *Acta Astronautica* 159 (2019), 123–135.

[19] Bradley Denby, Krishna Chintalapudi, Ranveer Chandra, Brandon Lucia, and Shadi Noghabi. 2023. Kodan: Addressing the computational bottleneck in space. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 392–403.

[20] Bradley Denby and Brandon Lucia. 2020. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 939–954.

[21] ExaTech 2024. ExaTech (in Chinese). https://www.exadevice.com/exaspace, accessed on 2025-02-05.

[22] Hao Fang, Haoyuan Zhao, Feng Wang, Yi Ching Chou, Long Chen, Jianxin Shi, and Jiangchuan Liu. 2024. Streaming Media over LEO Satellite Networking: A Measurement-Based Analysis and Optimization. *ACM Transactions on Multimedia Computing, Communications and Applications* (2024).

[23] Federal Communications Commission (FCC). 2021. PETITION OF STARLINK SERVICES, LLC FOR DESIGNATION AS AN ELIGIBLE TELECOMMUNICATIONS CARRIER. https://www.telepolis.pl/images/2021/02/Starlink-Services-LLC-Application-for-ETC-Designation.pdf, accessed on 2025-02-05.

[24] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. 2013. Pushing CDN-ISP collaboration to the limit. *ACM SIGCOMM Computer Communication Review* 43, 3 (2013), 34–44.

[25] Johan Garcia, Simon Sundberg, Giuseppe Caso, and Anna Brunstrom. 2023. Multi-timescale evaluation of starlink throughput. In *Proceedings of the 1st ACM Workshop on LEO Networking and Communication*. 31–36.

[26] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. 2021. Seven years in the life of Hypergiants' off-nets. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 516–533.

[27] Giacomo Giuliari, Tommaso Ciussani, Adrian Perrig, and Ankit Singla. 2021. {ICARUS}: Attacking low earth orbit satellite networks. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 317–331.

[28] Google. 2024. Google Global Cache. https://peering.google.com/#/options/google-global-cache, accessed on 2025-02-05.

[29] Evan W Gretok, Evan T Kain, and Alan D George. 2019. Comparative benchmarking analysis of next-generation space processors. In *2019 IEEE Aerospace Conference*. IEEE, 1–16.

[30] Huawei Huang, Song Guo, and Kun Wang. 2018. Envisioned wireless big data storage for low-earth-orbit satellite-based cloud. *IEEE Wireless Communications* 25, 1 (2018), 26–31.

[31] IMPROVING STARLINK'S LATENCY. 2024. https://api.starlink.com/public-files/StarlinkLatency.pdf, accessed on 2025-02-05.

[32] International Organization for Standardization. 2023. ISO 3166-1: Codes for the representation of names of countries and their subdivisions — Part 1: Country codes. https://www.iso.org/iso-3166-country-codes.html accessed on 2025-02-05.

[33] IP-API.com - Geolocation API. 2024. https://ip-api.com/, accessed on 2025-02-05.

[34] Nesrine Kaaniche and Maryline Laurent. 2017. Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms. *Computer Communications* 111 (2017), 120–141.

[35] David Karger, Alex Sherman, Andy Berkheimer, Bill Bogstad, Rizwan Dhanidina, Ken Iwamoto, Brian Kim, Luke Matkins, and Yoav Yerushalmi. 1999. Web caching with consistent hashing. *Computer Networks* 31, 11-16 (1999), 1203–1213.

[36] T.S. Kelso. 2024. Celestrak: A Resource for Satellite Orbital Elements. https://celestrak.org Accessed on 2025-02-05.

[37] kworb. 2024. Kworb.net - All your music data needs in one place. https://kworb.net/, accessed on 2025-02-05.

[38] Zeqi Lai, Hewu Li, and Jihao Li. 2020. StarPerf: Characterizing Network Performance for Emerging Mega-Constellations. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[39] Zeqi Lai, Hewu Li, Qi Zhang, Qian Wu, and Jianping Wu. 2021. Cooperatively Constructing Cost-Effective Content Distribution Networks upon Emerging Low Earth Orbit Satellites and Clouds. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. 1–12. https://doi.org/10.1109/ICNP52444.2021.9651950

[40] Suoheng Li, Jie Xu, Mihaela Van Der Schaar, and Weiping Li. 2016. Popularity-driven content caching. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[41] Yuanjie Li, Lixin Liu, Hewu Li, Wei Liu, Yimei Chen, Wei Zhao, Jianping Wu, Qian Wu, Jun Liu, and Zeqi Lai. 2024. Stable Hierarchical Routing for Operational LEO Networks. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. 296–311.

[42] Weisen Liu, Zeqi Lai, Qian Wu, Hewu Li, Qi Zhang, Zonglun Li, Yuanjie Li, and Jun Liu. 2024. In-Orbit Processing or Not? Sunlight-Aware Task Scheduling for Energy-Efficient Space Edge Computing Networks. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 881–890.

[43] Sami Ma, Yi Ching Chou, Haoyuan Zhao, Long Chen, Xiaoqiang Ma, and Jiangchuan Liu. 2023. Network characteristics of leo satellite constellations: A starlink-based measurement from end users. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 1–10.

[44] Yun Ma, Xuanzhe Liu, Shuhui Zhang, Ruirui Xiang, Yunxin Liu, and Tao Xie. 2015. Measurement and analysis of mobile web cache performance. In *Proceedings of the 24th International Conference on World Wide Web*. 691–701.

[45] Edouard Mathieu. 2024. *The price of computer storage has fallen exponentially since the 1950s*. https://ourworldindata.org/data-insights/the-price-of-computer-storage-has-fallen-exponentially-since-the-1950s Accessed: 2025-02-05.

[46] Sparsh Mittal. 2017. A survey of techniques for cache partitioning in multicore processors. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–39.

[47] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning edge research with latency shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*. 182–189.

[48] Kianoosh Mokhtarian and Hans-Arno Jacobsen. 2016. Coordinated caching in planet-scale CDNs: Analysis of feasibility and benefits. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[49] Netflix. 2024. Netflix | Open Connect. https://openconnect.netflix.com/, accessed on 2025-02-05.

[50] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: accurate {Record-and-Replay} for {HTTP}. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 417–429.

[51] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. 2010. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review* 44, 3 (2010), 2–19.

[52] Elizabeth J O'neil, Patrick E O'neil, and Gerhard Weikum. 1993. The LRU-K page replacement algorithm for database disk buffering. *Acm Sigmod Record* 22, 2 (1993), 297–306.

[53] Nils Pachler, Inigo del Portillo, Edward F Crawley, and Bruce G Cameron. 2021. An updated comparison of four low earth orbit satellite constellation systems to

provide global broadband. In *2021 IEEE international conference on communications workshops (ICC workshops)*. IEEE, 1–7.

[54] Georgios S Paschos, George Iosifidis, Meixia Tao, Don Towsley, and Giuseppe Caire. 2018. The role of caching in future communication systems and networks. *IEEE Journal on Selected Areas in Communications* 36, 6 (2018), 1111–1125.

[55] Stefan Podlipnig and Laszlo Böszörmenyi. 2003. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)* 35, 4 (2003), 374–398.

[56] Ingmar Poese, Benjamin Frank, Georgios Smaragdakis, Steve Uhlig, Anja Feldmann, and Bruce Maggs. 2012. Enabling content-aware traffic engineering. *ACM SIGCOMM Computer Communication Review* 42, 5 (2012), 21–28.

[57] Konstantinos Psounis and Balaji Prabhakar. 2001. A randomized web-cache replacement scheme. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, Vol. 3. IEEE, 1407–1415.

[58] Moinuddin K Qureshi and Yale N Patt. 2006. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*. IEEE, 423–432.

[59] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D Strowes, and Narseo Vallina-Rodriguez. 2018. A long way to the top: Significance, structure, and stability of internet top lists. In *Proceedings of the Internet Measurement Conference 2018*. 478–493.

[60] Richard Jay Solomon, Eric Rosenthal, Rodney Grubbs, and Brian D Solomon. 2021. Next Generation Big Data Storage For Long Space Missions. In *2021 IEEE Aerospace Conference (50100)*. IEEE, 1–7.

[61] Zhenyu Song, Daniel S Berger, Kai Li, Anees Shaikh, Wyatt Lloyd, Soudeh Ghorbani, Changhoon Kim, Aditya Akella, Arvind Krishnamurthy, Emmett Witchel, et al. 2020. Learning relaxed belady for content distribution network caching. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 529–544.

[62] SpaceNetLab. 2025. *Global Web Cache Latency Measurement for Low-Earth Orbit Satellite Networks: A Dataset.* https://doi.org/10.5281/zenodo.14835444

[63] Starlink. 2024. SpaceX's StarLink. https://www.starlink.com/, accessed on 2025-02-05.

[64] Starlink Progress Report of the last four years. 2024. https://stories.starlink.com/, accessed on 2025-02-05.

[65] Starlink Progress Report of the last three years. 2023. https://stories.starlink.com/, accessed on 2024-09-22.

[66] Hammas Bin Tanveer, Mike Puchol, Rachee Singh, Antonio Bianchi, and Rishab Nithyanand. 2023. Making sense of constellations: Methodologies for understanding starlink's scheduling algorithms. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*. 37–43.

[67] Thales Alenia Space reveals results of ASCEND feasibility study on space data centers. 2024. https://www.thalesaleniaspace.com/en/press-releases/thales-alenia-space-reveals-results-ascend-feasibility-study-space-data-centers-0, accessed on 2025-02-05.

[68] Peng Wang, Yu Liu, Ziqi Liu, Zhelong Zhao, Ke Liu, Ke Zhou, and Zhihai Huang. 2024. epsilon-LAP: A Lightweight and Adaptive Cache Partitioning Scheme with Prudent Resizing Decisions for Content Delivery Networks. *IEEE Transactions on Cloud Computing* (2024).

[69] Gang Yan, Jian Li, and Don Towsley. 2021. Learning from optimal caching for content delivery. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 344–358.

A List of Abbreviations

Table 1: List of Abbreviations

Abbreviation	Full Form
LEO	Low-Earth Orbit
LSN	Low-Earth Orbit Satellite Network
ISP	Internet Service Provider
AS	Autonomous System
PoP	Point of Presence
UT	User Terminal
USL	User Terminal-to-Satellite Link
ISL	Inter-Satellite Link
GSL	Ground-to-Satellite Link
RDS	Route Distribution Service
ICP	Internet Content Provider
CDN	Content Delivery Network
NA	North America
EU	Europe
OC	Oceania
AF	Africa
AS	Asia
LFU	Least Frequently Used
LFUDA	Least Frequently Used with Dynamic Aging
LRU	Least Recently Used
LRUK	Least Recently Used-K
BSC	Baseline Space Caching
GWC	Ground-Only Web Caching
PLT	Page Load Time

B Discussion

Technical feasibility of on-satellite web cache. (i) On-satellite storage capacity: In recent years, on-board storage capacity keeps evolving [30, 60], and deploying large volume caches on satellites is no longer impossible. For example, ExaTech aims to build an Exa-Byte space data center and has already made many storage units into space, ranging from 1 TB (on CubeSats below 20 kg) to 40 TB (on medium satellites of ~1000 kg) [21]. Similarly, European Space Agency (ESA) is also exploring space data centers and confirmed their technical and economic feasibility through its ASCEND project [67]. The evolution of space storage hardware and the ongoing exponential cost reduction of storage devices [45] have paved the way for deploying web cache in space. (ii) Cacheability of web contents at the edge: Akamai [5], Netflix [49], and Google [28] have been deploying cache nodes *as close as possible* to users, through collaboration with ISPs [24, 26, 56]. Google / Netflix reports that around 65%/95% of its traffic is cacheable and served inside these ISP networks [28, 49].

Practical challenges and future directions. While SPACHE demonstrates the potentials of latency performance improvements, practical deployment faces challenges. (i) Satellite hardware constraints — such as computation capacities, thermal limitations and power consumption — require careful optimization. However, with the enhancement of onboard hardware technology in recent years, emerging satellites can support complex and even intelligent tasks [11, 20]. Also, recent advancements in energy-efficient edge computing for satellites (*e.g.,* sunlight-aware task scheduling [42]) suggest that these barriers are surmountable. (ii) Fault tolerance remains critical. As a "first step" towards integrating LEO and web caching, the current fault-tolerance approach of this work is to fall back to origin servers upon cache misses caused by satellite failures. Future work will integrate redundancy mechanisms inspired by distributed cloud systems (*e.g.,* consistent hashing[35]) to enhance robustness. (iii) Security and privacy concerns. Data sovereignty, privacy and international regulatory considerations faced by SPACHE are nontrivial. A lot could be learned from the existing solutions [34] and the exploration of security and privacy-enhancing techniques for web caching in space requires further efforts.

Adaptability and broader impact. SPACHE is adaptable to different constellations, *e.g.,* Starlink, OneWeb, *etc.* While constellation characteristics (like satellite densities and inter-satellite link designs) can lead to diverse user access strategies and network topologies, SPACHE's policies are not dependent on specific topological features. As long as the LSN operator maintains network connectivity, SPACHE nodes can dynamically adjust their caching strategies based on their service areas derived from the communication schedules and their online status. Additionally, as 5G/6G networks integrate non-terrestrial components, SPACHE could serve as a critical caching layer for global low-latency services.