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

Anonymous Author(s)

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 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 real-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 sustain consistently low user-to-cache latency for global LSN users.

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 of cloud platforms (*e.g.,* Amazon CloudFront [17]) and Internet service providers (ISP, *e.g.,* Verizon[18]), web content providers can push their web replicas to network edges close to users. User's HTTP request can get a faster response from these cache servers instead of from the source. Hence, distributed web cache is widely deployed and used in existing terrestrial networks, including wired networks [39, 43] and WiFi/cellular networks [46].

Parallel to the swift progression of web technology, Internet infrastructure also advances at an unprecedented pace. The recent satellite Internet constellations, such as Starlink [53], OneWeb [4] and Amazon Kuiper [3], are deploying thousands of low-earth orbit (LEO) broadband satellites to revolutionize the way people access the Internet. For example, as of the date of Oct. 2024, SpaceX's Starlink, the largest operational LEO satellite network (LSN) today, has provided Internet services for more than 3 million global subscribers [54], especially for those in remote and rural areas [27, 54]. 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 [19]. Since LSNs make web services more "worldwide", it should be important for both web content providers and satellite ISPs to understand how existing web infrastructures perform 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 (§2). Our analysis based on 76 probes with Starlink terminals in 5 continents recruited from the RIPE [1] platform 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 client-to-cache RTTs, causing high web browsing latency (*e.g.,* about 40.79% LSN users experience RTTs higher than 100ms 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 (§3,§4). 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 is continuously changing, 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 when an LSN entity (*e.g.,* a user terminal, a satellite or a ground gateway) should communicate to another entity. Concretely, SPACHE adopts a schedule-driven prefetching mechanism to efficiently warm cache in space, and uses 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 guarantee low user-to-cache latency in all considered continents.

[1] SPACHE indicates integrating space and cache.

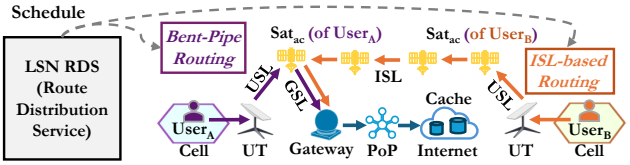


Figure 1: Architecture of existing LSNs (*e.g.*, Starlink).

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 complement the limitations of terrestrial web cache, we present SPACHE, a novel caching system to deploy and manage web cache 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.

We will open source our datasets to stimulate further study.

2 BACKGROUND AND MOTIVATION

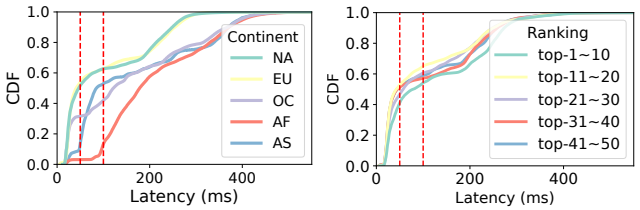
2.1 How LSN Users Access Current Web Cache?

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. Figure 1 illustrates the LSN architecture used by current satellite network operators like SpaceX. To access the Internet, satellite users connect their devices (*e.g.*, laptops, 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). In addition, satellites can interconnect to each other 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 for users in a certain cell to access the Internet is decided by the route distribution service (RDS) [14] in advance and distributed as scheduled to satellite routers along the route. 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 [36]). Otherwise, user traffic needs to be forwarded to the gateway via multi-hop satellite forwarding (*i.e.*, ISL-based routing [34]).

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 [41] and Google's Global Cache [24]) 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 [1] probes equipped with Starlink terminals, we identify three limitations of existing terrestrial web cache.



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

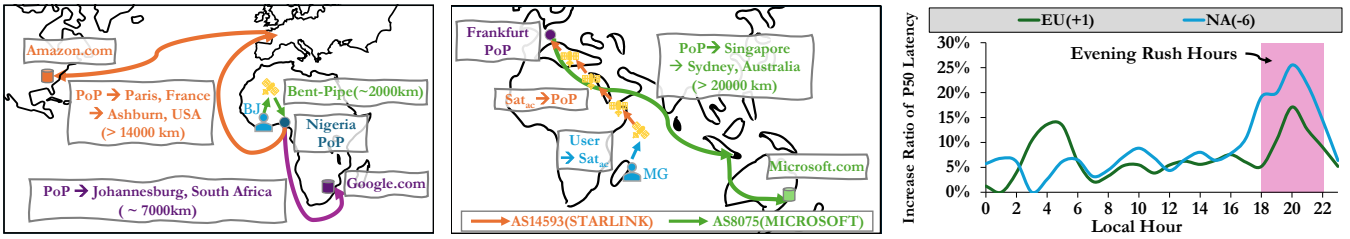
Figure 2: Web cache RTTs perceived by LSN users. NA: North America. EU: Europe. OC: Oceania. AF: Africa. AS: Asia. 50 ms / 100 ms are marked with red dashed lines.

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 Alexa's top-50 websites [50] 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 [37]. To build our geo-distributed LSN vantage points, we recruit 76 Starlink probes in 5 continents based on the RIPE platform.

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 48.7%/56% of our vantage points in NA/EU and all vantage points in OC/AF/AS, the average user-to-cache RTT is higher than 100ms. To uncover the root causes of the high RTTs perceived by LSN users, we use `traceroute` and IP geo-location service [29] 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



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

Figure 3: Dissecting the main sources of latency to web caches.

cloud providers (*e.g.,* CloudFront [17], Microsoft Azure [7]) or terrestrial ISPs (*e.g.,* Verizon [18]). Thus, an LSN user needs to cross multiple autonomous systems (ASes) to access the cache. Note that in current LSN architecture, traffic between the satellite and non-satellite ISP must be aggregated and exchanged at certain 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 PoPs at Frankfurt, and then be forwarded to the assigned cache in Sydney via terrestrial paths in opposite direction (and vice versa). 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 LEO 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 *LEO satellite links*. Figure 3c plots the RTT variations in 7 days measured from our vantage points. We have made several observations. First, the user-to-cache RTTs fluctuate over time, and reach the highest value in the peak hours of every day. Second, the gateway-to-cache RTTs are stable, indicating that the LEO link is likely the bottleneck. Finally, the above observations suggest that even though the baseline delay between an LSN user and the assigned web cache is low, the network congestion in LEO links 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.

Technical feasibility of on-satellite web cache. In recent years, on-board storage capacity keeps evolving [26, 51], 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 1TB (on CubeSats below 20 kg) to 40TB (on medium satellites ~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 [56]. The evolution of space storage hardware has paved the way for deploying web cache in space.

Benefits of on-satellite web cache. Caching web contents in space potentially 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-range web access globally.
- **(ii) Web caching in space can avoid meandering routes between different ISPs' ASes.** Different from existing terrestrial 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 possible 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.** Deploying web cache solely on the ground causes web contents to aggregate at gateways and PoPs, which are reported as the bottlenecks of LSNs [19, 45, 54]. With LEO cache, the content can be served directly from the access satellites, after the first cache miss. Thus, LEO cache could potentially cut down the traffic converged at gateways and PoPs, and then alleviate congestion on LEO last mile that hinders the quality of web services.

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

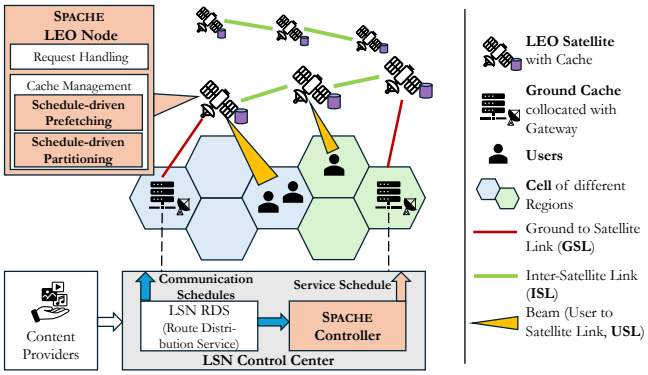


Figure 4: Overview of SPACHE architecture.

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 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 (Sat_{ac}) is only visible to a terrestrial user for up to about 3 minutes. 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 [47] 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 Bélády's optimal algorithm [9]. In real systems, practical algorithms like LRU [52] 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 (sets of) regions, these algorithms tend to misestimate the value of contents and lead to unexpected cache pollution, *e.g.,* overestimating the value of a popular content from previously served regions and keeping it in cache, which is useless currently. Cache pollution may even prevent the popular contents of the upcoming regions from staying in the cache, as we observe in §5.

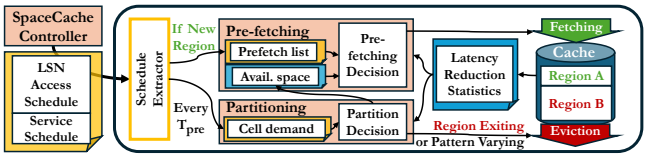


Figure 5: Schedule-Driven Cache Management.

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 the official document published by SpaceX [14, 22], in current Starlink network, a *communication schedule* indicates what entities (*e.g.,* a satellite, a UT or a gateway) should communicate with what other entities, and when certain entities should communicate with other certain entities. 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 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 RN_{c_j}). The SPACHE controller attaches this service information to the communication schedules, transmits them to the LEO cache nodes serving the corresponding cells, and receives calibration feedback from the nodes.

4.2 Schedule-Driven Cache Prefetching

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 to prefetch? The controller monitors the schedules and when it finds that an LEO cache node is about to serve a new region r_{new}, it provides the prefetching content list $\mathbf{\Phi}_{\mathbf{r}_{\mathbf{new}}}$ to the node in addition to the communication schedule. $\mathbf{\Phi}_{\mathbf{r}_{\mathbf{new}}}$ is constructed based on content popularity that the controller keeps monitoring [43], or based on content recommendation [5]. The default strategy is to include the top k popular contents with a cumulative popularity

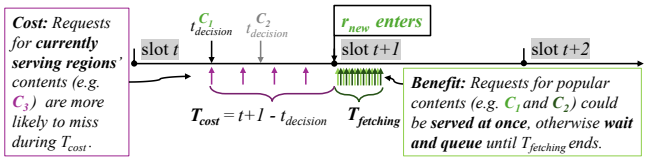


Figure 6: Prefetching decision at $t_{decision}$.

exceeding a threshold β:

$$|\Phi| = \text{argmin}_k \sum_{i=1}^{k} \text{Pop}_i \geq \beta, \quad (1)$$

where Pop_i represents the popularity of content C_i in region r_{new}, estimated with the ratio of the request number of the content C_i to the total request number of the region r_{new}.

Where to prefetch? The controller also provides a prefetching source list $\mathbf{Src}_{\mathbf{r}_{\text{new}}}$. This list includes suitable ground cache nodes and other LEO cache nodes that already possess the content in $\mathbf{\Phi}_{\mathbf{r}_{\text{new}}}$. Each source in $\mathbf{Src}_{\mathbf{r}_{\text{new}}}$ is associated with a fetching latency ($T_{fetching}$) representing the time for the LEO cache node to obtain the content from that source.

When to prefetch? Upon receiving $\mathbf{\Phi}_{\mathbf{r}_{\text{new}}}$, the LEO cache node continuously evaluates whether to prefetch the most popular content C_i in the list. This process continues until the node enters r_{new} or $\mathbf{\Phi}_{\mathbf{r}_{\text{new}}}$ is empty. When the cache node contains the whole prefetching list, it notifies the controller and could be a fetching source for other LEO cache 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_i from the source with the lowest $T_{fetching}$ in $\mathbf{Src}_{\mathbf{r}_{\text{new}}}$. Prefetching C_i allows immediate response of requests upon entering r_{new}, avoiding fetching delays. Here the benefit is calculated as:

$$\text{LatencyBenefit}_{\text{prefetching } C_i} = \frac{T_{fetching}^2}{2\Delta_t} \times \sum_j \mathbf{RN}_{\mathbf{c}_j^{\text{rnew}}} \times \text{Pop}_i, \quad (2)$$

where $\{c_j^{r_{new}}\}$ represents the set of cells in r_{new}, and $\sum_j \mathbf{RN}_{\mathbf{c}_j^{\text{rnew}}}$ is the total request number within r_{new} for the time slot.

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

$$\text{LatencyCost}_{\text{prefetching } C_i} = \frac{T_{cost}}{\Delta_t} \times \sum_j \mathbf{RN}_{\mathbf{c}_j^{\text{revict}}} \times SA - LR_{r_{evict}}, \quad (3)$$

where T_{cost} is the duration of the cost impact, r_{evict} is the region where the evicted item is from. r_{evict} is determined by SPACHE's schedule-driven partitioning strategy, based on the schedule-based average latency reduction ($SA - LR$) 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 burst 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 prevents content requests from one group from evicting content that is frequently accessed by another group, *i.e.,* avoiding cache pollution where valuable content is replaced by less useful content. Cache partitioning was originally designed in CPU cache when it turned into the *shared cache for multi-programs* as multi-core processors arose, to cope with cache pollution due to different programs' disparate cache demands and performance sensitivities [38, 49]. In recent years, cache partitioning is extended to widely used in terrestrial CDNs [57] to improve existing cache replacement algorithms [38].

New challenges of applying cache partitioning in LEO satellite 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 [38, 57]. In LEO satellite cache, where the service regions change rapidly, these methods require additional time after service region switches, to collect new data and gradually evict content no longer relevant to the exited region. This still leads to cache pollution.

Schedule-drive 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 independent sub-caches, with each region served by its own sub-cache using a certain cache replacement algorithm. When serving terrestrial users, a request from a certain region for a particular content set is handled by the corresponding sub-cache and only causes content eviction there.

4.3.1 Schedule-based average latency reduction. First, SPACHE proposes a future-aware utility metric, *Schedule-based Average Latency Reduction* (SA-LR) for each region. For a request, its latency reduction LR_{req} is defined as the miss delay minus the actual request delay. For a region's sub-cache in time-slot t, $LR_{r_i}^t$ sums the latency reductions of all requests. Further, an LEO cache node extracts the cells it will serve in time-slot t based on the schedule information, and then estimates the request rates $\mathbf{RN}_{\mathbf{r_i}}$ for each region in time-slot t by summing the per-cell request rate $\{RN_{c_j}\}$ provided by controller. For newly served regions, the estimated request rate is summed based on the schedule. Collectively, Schedule-based Average Latency Reduction $SA - LR_{r_i}^t$ is calculated as:

$$SA - LR_{r_i}^t = \frac{LR_{r_i}^{t-1}}{P_{r_i}^{t-1}} \times \frac{RN_{r_i}}{\sum_{j=1}^{N} RN_{r_j}}. \quad (4)$$

4.3.2 Schedule-driven adaptive adjustment. At the beginning of each time slot, regions are ranked by $SA - LR$, and the sub-cache of the $(N - k + 1)$-th region is reduced by $D\%$ while adding it to the k-th region's sub-cache, where N is the number of regions served in this slot. Existing works often set D as a fixed value [10, 57], but 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 ΔRN of the regions. Thus, in

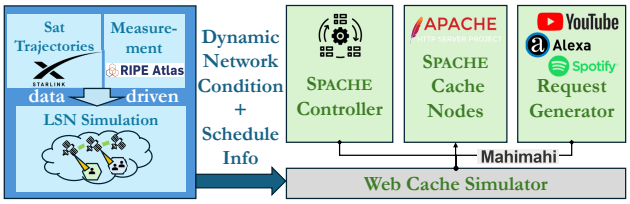


Figure 7: Key components our experiment environment.

case a region exits abruptly, the cache space of the exited region is quickly reallocated to other regions, minimizing cache pollution.

5 EVALUATION

To evaluate SPACHE's improvement for web browsing in various aspects, we implement a prototype of SPACHE, built a trace-driven LSN simulator based on real-world constellation information, and construct an experiment environment as plotted in Figure 7.

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 pre-fetching 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` [52], 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.

LSN simulation. We built an LSN simulator based on the methodology introduced by [31] and the real satellite trajectories of Starlink [22, 23]. The earth surface is divided into different regions and cells. Specifically, regions are partitioned based on the ISO 3166-1 country standard [28], and each region has its own content set. Cells are partitioned using a mesh of 1 degree longitude and 1 degree latitude. Each cell is located in a certain region. We simulate the Starlink's global scheduler [14, 55] which dynamically updates the connections between satellites and UTs in different cells in 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 geo-diversity indicates less overlap of popular contents across regions. We generate 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 the two datasets collected from YouTube (video content)[2] and Spotify (music content)[30]. The two datasets contain the list of Top-100~200 contents of each service region and their number of views. YouTube dataset covers 137 service regions and Spotify dataset covers 71 service regions, with both covering 6 continents.

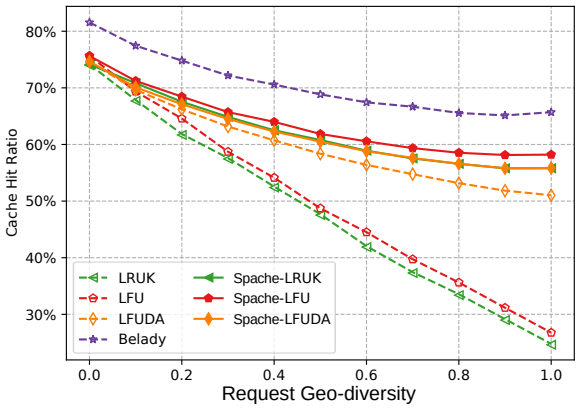


Figure 8: Improvement by SPACHE's schedule-driven partitioning under various request geo-diversity.

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 services; (ii) baseline space caching (denoted as BSC), which deploys web cache servers on LEO satellites, but does not leverages schedule information for content prefetching and cache partitioning like SPACHE; (iii) StarFront [32], 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` [42] to mimic the dynamic LSN connectivity 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 to evaluate the hit rate improvement made by SPACHE's cache partitioning on existing replacement algorithms: LFU [12], LFUDA [6] and LRUK [44]. Figure 8 shows the hit rate achieved by different replacement algorithms with or without SPACHE schedule-driven cache partitioning under different geographical 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

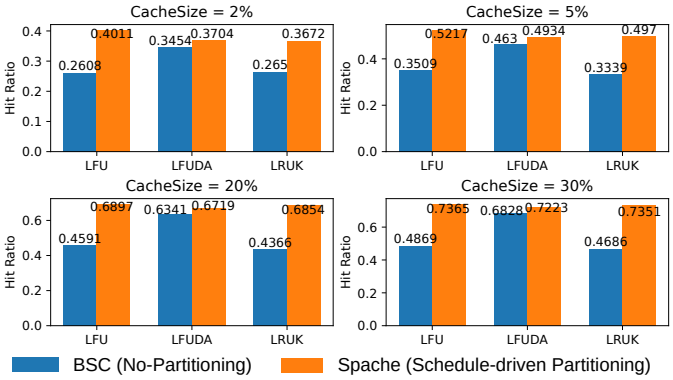
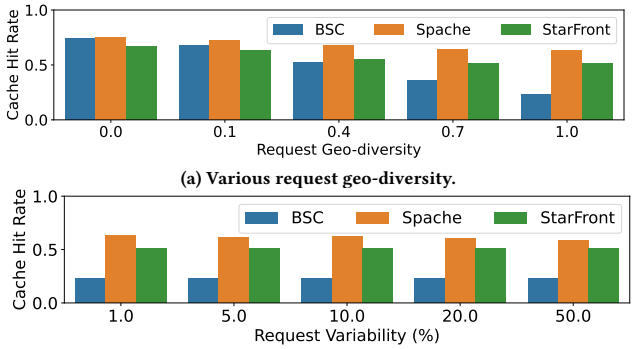


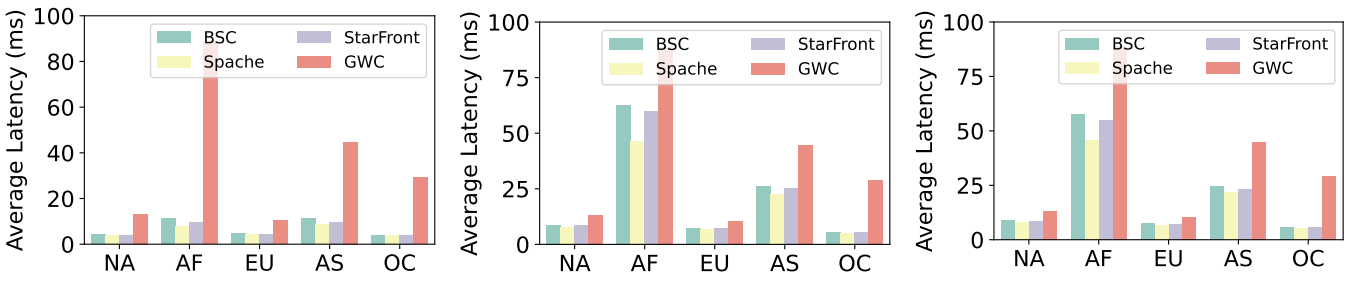
Figure 9: Improvement by Spache's schedule-driven partitioning under various cache capacity. CacheSize = X% means the capability to contain X% of the catalog of one region.



(a) Various request geo-diversity.

(b) Various request temporal variability.

Figure 10: Hit rate achieved by different caching systems.



(a) User-to-cache latency of Alexa Top-50 Websites.

(b) User-to-cache latency of YouTube Datasets.

(c) User-to-cache latency of Spotify Datasets.

Figure 11: User-perceived web experience: uset-to-cache latency.

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 on average 57.68%/19.8% higher hit rate than BSC/StarFront. What's more, as the geo-diversity increases, the improvement of Spache over BSC and StarFront becomes more significant. This is because even under diverse demands from different regions, Spache could effectively utilize the schedule information and prefetch the popular content 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 model temporal variability of V%, we randomly select V% of the content from the top half of the popularity ranking and swap their popularity with content from the bottom half. Figure 10b shows that under different request temporal variability rates, Spache could resiliently achieve better performance than the BSC and StarFront. We notice that BSC and StarFront is not sensitive to the request variability. However, although Spache is slightly affected by the request variability, as the popularity of some prefetched contents of it may be lower than expected, it still outperforms BSC/StarFront by 153.7%/15.6% at high request variability of 50%.

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 three datasets: Alexa Top-50 websites, Youtube and Spodify traces. We observe that by deploying cache on LEO satellite constellations, Spache could achieve low user-to-cache RTTs at different continents. Specifically, Spache improves the user-to-cache latency to the 50 websites by on average 76.9% 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 less portion of global popular contents and higher latency to terrestrial caches. Compared to the StarFront, Spache improves the user-to-cache latency by on average 6.4% and up to 17.7% at different continents.

Further, we compare the results obtained by the overall 50 websites and these two representative applications Youtube and Spodify. We observe that for local popular content like Youtube and Spodify, the overall latency is higher than the global popular content (*e.g.*, all 50 websites), as the local content occuies a smaller portion of

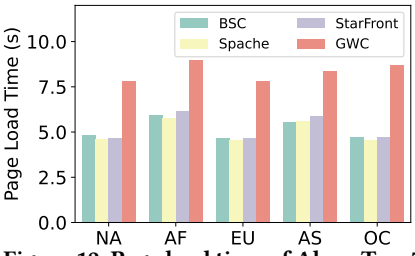


Figure 12: Page load time of Alexa Top-50 Websites.

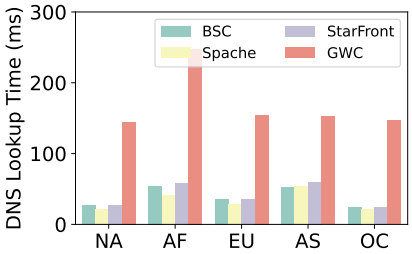


Figure 13: DNS lookup time of Alexa Top-50 Websites.

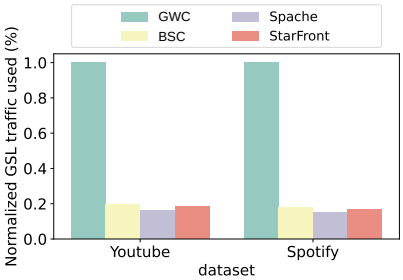


Figure 14: GSL traffic load under different caching solutions.

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 content (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 average traffic consumed by each LEO cache node to deliver the contents under different dataset-driven scenarios. We observe that Spache can achieve 83.65%/84.77% lower GSL traffic consumption than not installing on-satellite cache for YouTube/Spotify dataset. 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

We briefly discuss previous efforts closely related to Spache.

Optimizing content placement for distributed web cache. A considerable amount of research has been conducted on optimizing content placement for distributed web cache [15, 40, 43, 46]. For example, cache coordination [40] 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 [46] 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 consumption 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, 47, 48] (account the contents value with factors including content freshness, access frequency, data size, *etc.* [8]) or learning-based ones [33, 52, 58]. This paper differs from them in that the *cache infrastructure is continuously moving at high speed* all over the globe and the dynamic communication schedules of all the LSN entities could be pre-calculated precisely and provided in advance for LEO caches.

In-orbit processing and storage. Many recent works developed advanced in-orbit processing and storage capabilities for LEO satellites [11, 20, 25, 35, 51]. Kodan [20] is a recent orbital edge computing system that maximizes the utility of saturated satellite downlinks while mitigating the computational bottleneck. Phoenix [35] proposed a sunlight-aware space edge computing framework to optimize the energy efficiency of in-orbit processing. Authors in [51] reported a new storage media for long space missions, which is impervious to ionizing radiation, microgravity, solar (plasma) eruptions. This paper exploits these in-orbit processing and storage capabilities of LEO satellites to extend web caches into space, providing low-latency access to web caches anywhere.

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-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 real-world data-driven LSN simulation. Extensive evaluations demonstrate that Spache can improve hit ratio by 19.8% on average than SoA solutions, reduce latency by up to 17.7% compared to existing solutions, and enable consistently low user-to-cache latency in all considered continents.

REFERENCES

[1] 2024. RIPE Atlas Probes from Starlink (AS14593). https://atlas.ripe.net/probes/public?search=14593, accessed on 2024-06-01.

[2] 2024. YouTube Data API | Google Developers. https://developers.google.com/youtube/v3/.

[3] 2022kuiper [n. d.]. Amazon Kuiper. https://www.geekwire.com/2019/amazon-project-kuiper-broadband-satellite/.

[4] 2022oneweb [n. d.]. OneWeb. https://oneweb.net/.

[5] Xavier Amatriain. 2013. Big & personal: data and models behind netflix recommendations. In *Proceedings of the 2nd international workshop on big data, streams and heterogeneous source Mining: Algorithms, systems, programming models and applications*. 1–6.

[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 2024-09-22.

[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 2024-09-22.

[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] CloudFront [n. d.]. Content Delivery Network (CDN) - Amazon CloudFront - Amazon Web Services. https://aws.amazon.com/cloudfront/.

[17] Content Delivery Network - Amazon CloudFront - AWS 2024. https://aws.amazon.com/cloudfront/, accessed on 2024-09-22.

[18] Content Delivery Network (CDN) Services | Verizon Business 2024. https://www.verizon.com/business/products/security/web-security/web-acceleration-cdn/, accessed on 2024-09-22.

[19] 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.

[20] 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.

[21] ExaTech 2024. ExaTech. https://www.exadevice.com/exaspace.

[22] 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 2024-09-22.

[23] 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.

[24] Google 2024. Google Global Cache. https://peering.google.com/#/options/google-global-cache, accessed on 2024-06-01.

[25] 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.

[26] 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.

[27] IMPROVING STARLINK'S LATENCY. 2024. https://api.starlink.com/public-files/StarlinkLatency.pdf, accessed on 2024-09-22.

[28] 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 2024-09-22.

[29] IP-API.com - Geolocation API. 2024. https://ip-api.com/, accessed on 2024-09-22.

[30] kworb 2024. Kworb.net - All your music data needs in one place. https://kworb.net/.

[31] 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.

[32] 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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

[40] 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.

[41] netflix 2024. Netflix | Open Connect. https://openconnect.netflix.com/, accessed on 2024-06-01.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] Starlink 2024. SpaceX's StarLink. https://www.starlink.com/.
[54] Starlink Progress Report of the last three years. 2024. https://stories.starlink.com/, accessed on 2024-09-22.
[55] 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.
[56] 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 2024-09-22.
[57] 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).
[58] 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.