Optimize Replica Server Placement in a Satellite Network

Paper # 461, 12 pages + references

ABSTRACT

Satellite communication provides an attractive means of offering Internet connectivity to users in remote locations, such as villages, deserts, mountains, and at sea. However, transmitting content over satellite networks is significantly more expensive than traditional Internet. To address this issue, we propose placing content replica servers within satellite networks and optimizing replica placement for important performance metrics, such as latency, transmission, and storage cost. Our approach can support different types of satellite networks, including Low Earth Orbit (LEO), Medium Earth Orbit (MEO), Geostationary Orbit (GEO), and their combinations. An important challenge for supporting content replicas in such networks is that LEO and MEO satellites are constantly moving. We address this challenge by explicitly considering their moving trajectories and strategically optimizing not only client performance, but also the cost of transferring content from one satellite to another as needed. We demonstrate the effectiveness of our approach using both simulated traffic traces and a prototype system.

1 INTRODUCTION

The first satellite was launched in 1957 [13]. Since then satellite communication has gone through rapid development. We now have Geostationary Orbit (GEO), Medium Earth Orbit (MEO) and Low Earth Orbit (LEO) satellites. GEO, MEO, and LEO are 35,785 km, 2,000 – 36,000 km [48], and less than 2,000 km [40] from the Earth's surface, respectively.

GEO satellite appears stationary to the user on the ground, which simplifies communication as the ground station can connect to the same satellite in the same direction all the time. In comparison, both MEO and LEO satellites appear in motion to the ground station, and require the ground station to steer in different directions to stay connected and need to switch to a different satellite when the previous one goes out of sight [36]. Moreover, they require more satellites to cover the Earth due to their closer distance to the Earth. In return for the larger number of satellites and management complexity, MEO and LEO satellite communication enjoys lower latency and higher bandwidth owing to their proximity to the Earth. LEO satellite communication features the lowest delay and highest bandwidth, and can potentially support challenging real-time applications [33]. In this paper, we consider all these types of satellites but focus most on LEO satellite networks due to its support of real-time applications

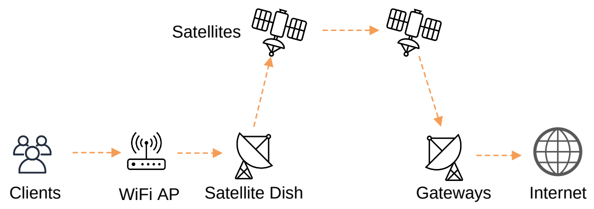

Figure 1: A typical path between a client and the Internet in LEO satellite network.

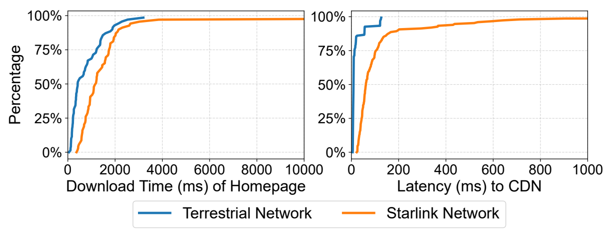

Figure 2: Measured download time of web homepage and latency to CDN in satellite network and terrestrial network.

and increasing commercial interest. For example, Starlink and OneWeb have 3000+ and 542 LEO satellites in orbits, respectively, while Amazon plans to launch over 3000 LEO satellites in the next few years.

As shown in Figure 1, a client using a LEO satellite communicates with an Internet server through the following path: its device $\longrightarrow$ its WiFi router $\longrightarrow$ its satellite dish $\longrightarrow$ a satellite $\longrightarrow$ 0 or more satellites $\longrightarrow$ a gateway closest to the server $\longrightarrow$ the server. This path involves two hops between ground and satellites and 0 or more hops among satellites. The delay can be significant.

We perform Web performance measurement using a Starlink ground station in Texas. We measure the download time of the homepage at the top 100 websites [38] and latency to the top 15 content delivery network (CDN) providers according to [11]. We compare it to the terrestrial network at the same location. Figure 2 shows that the median time to download a website homepage in Starlink is 2.7 times of terrestrial network, with a median latency to a CDN server 7.1 times. In addition, we observe a long tail in both the webpage download time and CDN latency in the Starlink satellite network likely due to satellite movement.

The significant end-to-end delay in LEO satellites motivates our design of CDN for LEO and other satellite systems. In the existing Internet, CDN is widely used to speed up content delivery to the user and reduce delivery cost [9]. Different from the traditional CDN, the relative position of LEO satellites and ground stations are constantly changing, so we can no longer place content on the same satellite to serve the same region. For example, as the satellite serving the region is moving away, its content may need to be replicated to another satellite in order to serve the same region. In general, we should explicitly take into account of the LEO orbits in order to design the CDN for LEO satellite communication.

More specifically, we formulate the replica placement problems for satellites as an optimization problem that selects the server locations to minimize the total cost of (i) delivering the content from replicas to users, (ii) replicating the content to replica servers, and (iii) the storage cost for the content.

We develop efficient algorithms that take advantage of satellite orbits. We evaluate the effectiveness of our approaches through both simulation and emulation in both web browsing and video streaming scenarios. The simulation experiments are based on three real web traces and use real latency values from the Starlink network. In addition, we implement a real-world prototype system that employs separate processes to mimic clients and servers, and uses data from the Starlink network to emulate network connections. Our results show that our methods can not only significantly enhance user experience, but also reduce replication cost and storage costs. We will release our code to the community. **This paper does not raise any ethical issues.**

2 BACKGROUND

2.1 Content Distribution Network (CDN)

CDN replicates content, such as web pages, software, audio, and video, across a distributed set of nodes so that a client can fetch the content from a nearby CDN node, thereby offloading the origin server, improving the latency, and reducing the cost. Some of the popular CDN providers include Akamai, Amazon AWS, Microsoft Azure, Google cloud.

On the research front, a number of theoretical models have been proposed for replica server placement [43], including facility location (i.e., minimizing the total cost of opening facilities and the cost of using the facilities to serve the clients) [42, 49, 54], K-median (i.e., selecting a given number of replicas to minimize the total cost of serving the clients) [27, 28, 39, 50], K-center (i.e., similar to K-median but minimizing the maximum cost instead of the total cost) [21, 22], K-cache (i.e., similar to K-median but considers the cache hit ratio and requires the request to go to the origin server if the replica does not have cached content) [23]. A series of algorithms have been developed to tackle the problems and most of them use greedy or other heuristic algorithms [43].

Current CDNs are designed for stationary networks. However, the network topology is constantly changing in satellite networks. If we place content to satellites, the satellites may need to replicate the content to other nodes when they move away from the interested regions. We need to explicitly take into account replication costs caused by satellite movements when designing a CDN for LEO satellite networks.

2.2 Satellite Networks

Artificial satellites are launched into the space to support a variety of applications, such as communication [19], weather monitoring [41], and navigation [16]. GEO satellites are the earliest communication satellites [15]. Since they have the same orbital period as the Earth, these satellites appear stationary to the ground station so that the ground station can steer in the same direction for communication. However, their large distance from the earth (35,785 km) makes it impossible to support time-critical traffic. To reduce the latency, LEO satellite communication is becoming popular [52]. Starlink has launched over 2,600 LEO satellites [2], OneWeb has over 600 satellites in orbit [5], and Amazon plans to launch 3,236 satellites in the next few years [4]. LEO satellites orbit around the Earth much faster than the Earth's orbital period. For example, the orbital period of a StarLink satellite is 95 minutes. MEO satellites are in between GEO and LEO in terms of their altitude, moving speed, and a number of satellites required to cover the Earth.

The communication cost in satellite networks can be significant, so CDNs are attractive for satellite communication. Unlike traditional Internet, where nodes are typically stationary, LEO and MEO satellites are constantly moving with respect to the end users and we need to dynamically move the content around to satisfy the same region. This motivates us to develop replication algorithms for satellite networks. (

3 OUR APPROACH

3.1 Problem Formulation

Our goal is to develop algorithms to determine the replica placement that minimizes the total cost of replicating the content and delivering it to end users during a given period.

We partition time into discrete slots t and represent the network as a time-dependent graph $G_t =< V, E_t >$. V consists of three types of vertices: V_{user} represents users or merged user regions, $V_{replica}$ represents potential replica servers such as satellites, ground stations, and terrestrial servers, and V_{origin} represents the origin server, which always holds the content. The edge set E_t is associated with time t and captures the dynamic connectivity between nodes. For example, one satellite v_a can be connected to a ground station v_b at time t_1

but disconnected at time t_2. In this scenario, we will always have v_a, v_b in the vertex set V but the edge between v_a and v_b will be in E_{t_1} but absent in E_{t_2}.

Users at different locations generate requests for content during each time slot. We choose popular files and represent them as a content set C. $size_c$ represents the size of file $c \in C$, and $demand_{v,c,t}$ represents the quantified request demand from user v for content c in time slot t.

Our task is to choose appropriate replica servers for each content over a given period. We define the replica set for content c in time slot t as $S_{c,t}$. The algorithm should choose appropriate $S_{c,t}$ given the network topology and user demand. Origin servers are always included in $S_{c,t}$ ($V_{origin} \subset S_{c,t}$). We add additional replica servers $v \in V_{replica}$ to $S_{c,t}$. The potential replica server set $V_{replica}$ consists of satellites, ground stations, and terrestrial servers. The replica set $S_{c,t}$ is time-dependent, allowing it to reflect the dynamic transfer and replication of content between satellites. For example, if a satellite $v_a \in S_{c,t}$ moves away from a serving region, it can be replaced in $S_{c,t+1}$ by another satellite v_b.

To measure different selections of replicas, we use 3 metrics: (1) query cost (the cost of serving user requests), (2) replication cost (the cost of replicating content to the replicas both at the beginning and in the middle), and (3) storage cost at the replicas. Note that our algorithms are general and can easily be applied to support other performance metrics.

Query cost: Query cost quantifies the cost of serving user requests. We assume that each user will query the closest replica and the total query cost will be proportional to the user number and demand. Specifically, the query cost is

$$\sum_{c}\sum_{t}\sum_{v_{user}\in V_{user}} demand_{v_{user},c,t} \times \min_{v\in S_{c,t}} cost^{t}_{query}(v_{user}, v)$$

where $cost^{t}_{query}(\cdot,\cdot)$ is a cost function between the user vertex and the server vertex. It is derived from the graph G_t, and can be quantified by hop count or latency.

Replication cost: We use replication cost to represent the costs of creating new replicas. It is formulated as

$$\sum_{c}\sum_{t}\sum_{v_{new}\in S_{c,t}} \min_{v_{old}\in S_{c,t-1}} cost^{t}_{replication}(v_{new}, v_{old})$$

where $cost^{t}_{replication}(\cdot,\cdot)$ is a cost function representing the cost of replicating content from an old replica to a new replica. If $v_{new} = v_{old}$, we will have $cost^{t}_{replication}(v_{new}, v_{old}) = 0$. A unique aspect of our formulation is that our network is dynamically changing and the replication cost between two nodes may change depending on their relative position at that time. This is due to the orbital organization of satellites, where replication to adjacent satellites in the same or neighboring orbits is typically more cost-effective than to distant satellites. We encourage content replication to occur as close as possible to reduce replication traffic.

Storage cost: We formulate storage cost as follows:

$$\sum_{c}\sum_{t}\sum_{v\in S_{c,t}} size_c \times cost_{storage}(v)$$

where $cost_{storage}(\cdot)$ is a cost function to quantify the cost of storing one unit size of content at different servers. Storing content on satellites is typically more costly than storing at ground servers [25]. By setting different storage costs for different kinds of servers, we can effectively leverage different kinds of replica servers to reach the optimal.

3.2 Our Replication Algorithms

Our problem is to find $S_{c,t}$ that minimizes the sum of the three costs (i.e., query cost, replication cost, and storage cost) across all time slots, where $S_{c,t}$ is chosen from $V_{replica}$. If only one time slot is considered, we can transform our problem to a well-known NP-hard problem – the uncapacitated facility location problem (UFL) [34], by treating the sum of replication cost and storage cost as the opening cost of a facility, and query cost as the transport cost in UFL. Since only solving one time slot is already NP-hard, our general multi-time-slot problem is also NP-hard. In this section, we will introduce our practical algorithms. We omit the dimension of content c in this part for ease of description because the algorithm can be applied to each content independently.

3.2.1 Multi-time Local Search.

We look for an effective algorithm that works well in practice. Inspired by the local search algorithm for the UFL problem [8], we first propose a multi-time local search (MTLS) method.

The UFL local search algorithm operates as follows: Given an initial facility set F, the algorithm allows for the addition, removal, or replacement of a facility in F if it results in an improvement in the objective. The algorithm continues this process until no further improvements can be made.

Suppose there are T time snapshots. Our task is to find T replica set $S_1, S_2, ..., S_T$. Similarly, we can define the nearby set of each S_t to be: $|S^{nearby}_t \setminus S_t| = 1$ or $|S_t \setminus S^{nearby}_t| = 1$. For the sake of efficiency, we do not test S^{nearby}_t one by one. Instead, we solve the following sub-problem: Given $\{S_1, S_2, ..., S_T\}$, what is the best solution among all possiblities of $\{S^{nearby}_1, S^{nearby}_2, ..., S^{nearby}_T\}$. Recall that we always have $V_{origin} \subset S_t$ and we need to choose new replicas from $V_{replica}$. Let $N = |V_{replica}|$. For each S_t, the total number of S^{nearby}_t will be $O(N^2)$, where adding a new replica yields $O(N)$ sets, deleting a replica yields $O(N)$ sets, and replacing a replica yields $O(N^2)$ sets. Using an exhaustive search method to acquire the best solution will take $O(N^{2T})$.

We use dynamic programming (DP) to avoid the exhaustive search and enhance efficiency, which takes $O(TN^4)$. To introduce our DP, we first introduce some symbols as follows. We define $f(t, S_t^{nearby})$ to represent the minimum total cost for time $1, 2...t$ and we are required to use replica set S_t^{nearby} at time t. $\hat{S}_{t-1}^{nearby}$ is defined as the replica set used at $t-1$. $RC(t, \hat{S}_{t-1}^{nearby}, S_t^{nearby})$ is defined as the replication cost from $\hat{S}_{t-1}^{nearby}$ to S_t^{nearby} following our definition in Section 3.1, while we use $QC(t, S_t^{nearby})$ and $SC(t, S_t^{nearby})$ to represent the query cost and storage cost at time t when we choose S_t^{nearby} as the replica set.

We are required to use S_t^{nearby} for $f(t, S_t^{nearby})$, thus the query cost and storage cost at time t are fixed, represented as $QC(t, S_t^{nearby})$ and $SC(t, S_t^{nearby})$, respectively. Besides, to minimize $f(t, S_t^{nearby})$, we should minimize sum of replication costs at time t ($RC(t, \hat{S}_{t-1}^{nearby}), S_t^{nearby})$ and total costs before time at time $t-1$ ($f(t-1, \hat{S}_{t-1}^{nearby})$). We enumerate all possible $\hat{S}_{t-1}^{nearby}$ and assign the smallest $RC(t, \hat{S}_{t-1}^{nearby}, S_t^{nearby}) + f(t-1, \hat{S}_{t-1}^{nearby}) + QC(t, S_t^{nearby}) + SC(t, S_t^{nearby})$ to $f(t, S_t^{nearby})$. Details are shown in Algorithm 1.

Our DP can be proved by mathematical induction. There is no user demand at $t = 0$ thus $f(0, V_{origin}) = 0$ is the best for $t = 0$. Now assuming that $f(t-1, \hat{S}_{t-1}^{nearby})$ is the smallest cost for time $1, 2, .., t-1$, $f(t, S_t^{nearby})$ has to be constructed from $t-1$ and we have enumerated all possibilities of replicas we can use in time $t-1$. Otherwise, there is a contradiction if we check the replica we use at $t-1$.

The time complexity $O(TN^4)$ comes from the following calculation. The number of total states in DP is TN^2, where N^2 is the number of nearby set (S_t^{nearby}). To get every state, we need to enumerate N^2 nearby sets (S_{t-1}^{nearby}) at time $t-1$. Thus the DP process will take $O(TN^4)$.

We find that the replacing operation yields $O(N^2)$ nearby sets, which is the bottleneck of computation and results in the term N^4 in the time complexity expression $O(TN^4)$. If we require that the replacing operation can only replace the replica v with its k nearest neighbors in the graph G_t, the replacing operation will only yield $O(kN)$ candidates, hence the complexity of DP will be reduced to $O(Tk^2N^2)$. It makes sense because replicating to nearby satellites incurs lower replication costs. In our evaluation, we set $k = 4$. The overall time complexity of MTLS is $O(MTk^2N^2)$, where M is the maximum number of iterations. We call this algorithm multi-time local search (MTLS) in this paper.

3.2.2 Multi-time Orbit-based Local Search. The MTLS algorithm is slow in practice due to the large value of N in modern LEO satellite constellations. To overcome this issue, we propose utilizing the orbit information of satellites.

Satellites are typically organized into multiple orbits, for instance, StarLink phase I has 72 orbits with 22 satellites each [18]. Given the fixed trajectory of orbits, we propose a hierarchical method for selecting satellite replicas. First, we select an orbit for each time snapshot ($o_1, o_2, ..., o_t$), then we pick the replica from the corresponding orbit o_t.

Algorithm 1 Multi-time Local Search (MTLS)

```
Input: $T, V_{origin}, V_{replica}, QC, RC, SC, M$
1:  $S_t = V_{origin}, t = 1, 2, .., T$
2:  for $m = 1$ to $M$ do                    ▷ iteratively update $S_t$
3:      $f(0, V_{origin}) = 0$
4:      for $t = 1$ to $T$ do
5:          for each $S_t^{nearby}$ at time $t$ do
6:              for each $\hat{S}_{t-1}^{nearby}$ at time $t-1$ do
7:                  $\hat{c} = QC(t, S_t^{nearby}) + SC(t, S_t^{nearby}) + RC(t, \hat{S}_{t-1}^{nearby}, S_t^{nearby}) + f(t-1, \hat{S}_{t-1}^{nearby})$
8:                  $f(t, S_t^{nearby}) = min\{\hat{c}, f(t, S_t^{nearby})\}$
9:              end for
10:         end for
11:     end for
12:     Update $S_t, t = 1, 2, .., T$ according to $f$
13: end for
Output: $S_t, t = 1, 2, .., T$
```

DP can also be applied to select orbits. For each time slot t and each orbit o, we first calculate the best satellite replica with the lowest query cost, denoted as $v_{o,t}$. Then, we determine the best orbit we can choose in every time slot. Here, choosing orbit o in time slot t means deploying the replica at $v_{o,t}$. In other words, we quantify the quality of the orbit based on the best replica in the orbit. The orbit selection DP utilizes $g(t, o)$ to represent the minimum cost for time $1, 2, ..., t$ and orbit o must be chosen at time t. We enumerate the chosen orbit in $t-1$ to minimize $g(t, o)$.

Given the orbit selection $o_1, o_2, ..., o_t$, and the current replica set $\{S_1, S_2, ..., S_T\}$, we require that only satellite from orbit o_t can be added to S_t when constructing the nearby set. We do not allow deletion and replacement operation to reduce computation cost. We arrive at the multi-time orbit-based local search (MTOLS) algorithm as shown in Algorithm 2.

Each iteration in MTOLS consists of one orbit selection DP and one replica selection DP. Suppose P is the orbit number, and Q is the satellite number in each orbit. The orbit selection DP takes $O(TP^2)$, and the replica selection DP takes $O(TQ^2)$. Therefore, the total time complexity of MTOLS is $O(MT(P^2 + Q^2))$, where M is the maximum iteration number. In contrast, the time complexity of MTLS is $O(MTk^2N^2)$, where $N = PQ$. MTOLS is faster than MTLS because $P^2 + Q^2 \ll P^2Q^2 = N^2 < k^2N^2$. The inequality $P^2 + Q^2 < P^2Q^2$ holds when $P > 2$ and $Q > 2$. As an example, we have $P = 72$ and $Q = 22$ for Starlink phase I satellites. MTOLS will be $\frac{22^2 \cdot 72^2}{22^2 + 72^2} \cdot k^2 \approx 442 \cdot k^2$ times faster than MTLS theoretically when applied to Starlink phase I.

Algorithm 2 Multi-time Orbit-based Local Search (MTOLS)

```
Input: $T, V_{origin}, V_{replica}, QC, RC, SC, M$
1: $S_t = V_{origin}, t = 1, 2, .., T$
2: for $m = 1$ to $M$ do                                  ▷ iteratively update $S_t$
3:     $g(0, \emptyset) = 0$                                  ▷ orbit selection DP
4:     for $t = 1$ to $T$ do
5:         for each $o_t$ at time $t$ do
6:             for each $\hat{o}_{t-1}$ at time $t - 1$ do
7:                 $\hat{c} = QC(t, o_t) + SC(t, o_t) + RC(t, \hat{o}_{t-1}, o_t) + g(t-1, \hat{o}_{t-1})$
8:                 $g(t, o_t) = min\{\hat{c}, g(t, o_t)\}$
9:             end for
10:        end for
11:    end for
12:    Get $o_t, t = 1, 2, .., T$ according to $g$
13:    Update $S_t$ similarly with Algorithm 1 but only adding one replica from $o_t$ is allowed to generate $S_t^{nearby}$
14: end for
Output: $S_t, t = 1, 2, .., T$
```

4 PRACTICAL CONSIDERATIONS

To realize a content replication system for a satellite network, there are several practical issues we need to consider: (1) predicting network topology, (2) predicting user demand, (3) supporting both web pages and video download, and (4) redirecting the users' requests to an appropriate replica.

For (1), to predict network topology, we leverage the public satellites' trajectory information.

For (2), we group users to regions and use Exponential Weighted Moving Average (EWMA) to predict the demand for each region. We examine the effectiveness of our methods based on both the predicted and ground truth demand.

For (3), we observe there is a major difference between downloading video versus downloading web pages. To support users with different network connectivity, videos are split into small encoded chunks (e.g., a few seconds each) at multiple bit rates. The client selects the desired bit rate according to its current network condition using an adaptive bitrate (ABR) algorithm [47]. A number of ABR algorithms have been proposed in the literature. Among them, Model Predictive Control (MPC) [55] has been shown to be effective and widely used. MPC optimizes video Quality of Experience, which is defined as $QoE_k = a \cdot R_k - b \cdot t_{rebuf,k} - c \cdot |R_k - R_{k-1}|$, where k is the chunk index, R_k is the selected bitrate, and $t_{rebuf,k}$ is the rebuffer time. We set $a = c = 1$ and $b = 4.3$ for our experiments and compare video QoE and replication cost for different replication strategies.

For (4), to redirect a user's request, we can borrow the DNS request redirection from the traditional CDN. It maps the client's location to an appropriate replica server. A simple mapping strategy is based on proximity. However, this may cause performance degradation during a flash crowd (i.e., the replica closest to a hotspot easily gets overloaded). One can alleviate this issue by redirecting to different replica servers. For example, a round robin strategy redirects the client requests randomly to one of the n closest replicas [53]. Weighted round robin strategy is similar to round robin except that it weights the probability of selecting a replica based on its proximity to the client [7]. To apply DNS redirection to the LEO network, whose topology changes very frequently, we need to refresh the DNS entries frequently. The frequent access can be alleviated by computing the content placement beforehand according to the satellite trajectories. Users can get content location periodically from the DNS server.

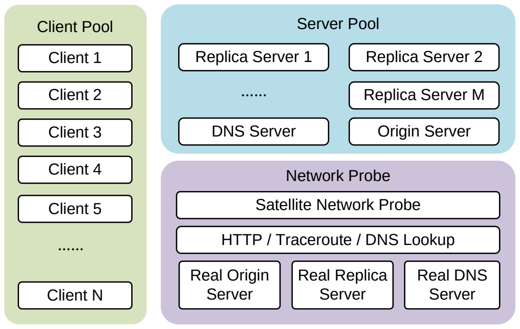

Figure 3: Prototype System Implementation

5 SYSTEM IMPLEMENTATION

We evaluate our system through a prototype implementation. As we do not have control access to the satellite network (e.g., replicating content to real satellites), we emulate a satellite network using network traces and let user requests go through network links according to the traces. Figure 3 shows our system. It consists of 3 parts: the client pool, the server pool, and the network probe. The client pool and the server pool are process pools. User clients, replica servers, the DNS server, and the origin server run as separate processes.

To conduct a realistic evaluation, the link performance between the user pool and the server pool follows the network traces. We collect the network traces in the Starlink network. We set up an origin server on the Internet, and measure its bandwidth and latency from the clients. Similarly, we measure the latency and bandwidth to a replica server by sending requests to an Akamai server [35]. We also use traceroute to measure the latency between a satellite and a gateway. We derive the latency between two satellites based on their distance and speed of light. Meanwhile, we also measure the latency to a DNS server by sending DNS queries.

We construct links in our testbed according to the network traces obtained from the above measurement. As discussed in Section 4, we pre-compute the placement of content. All clients have to query the DNS server every 10 minutes to get the latest location information of the content.

6 EVALUATION

We conduct comprehensive experiments to evaluate the performance of our methods in comparison to baselines and uncover key features of satellite networks. Experiments include trace-driven simulations and prototype implementation.

6.1 Experiment Setup

Satellite Constellations: Satellite constellations can be roughly categorized into 3 types by the orbit height: LEO, MEO, and GEO satellites. In this paper, we want to examine all of these satellite constellations. We select StarLink phase I for LEO, O3b for MEO, and ViaSat for GEO. StarLink phase I consists of 1,584 satellites in 72 orbits at 550km height [18]. O3b, owned by SES S.A., is a MEO constellation of 20 satellites operating at 8,062km [19]. ViaSat consists of 4 geostationary satellites [6]. All three constellations provide Internet access and have real-world applications. We enable inter-satellite connections for Starlink, with each satellite connected to 4 neighboring satellites in its orbit and adjacent orbits [12]. O3b and ViaSat do not have inter-satellite links.
Gateways: As illustrated in Figure 1, the user client connects to satellites, which then connect to gateways, also known as ground stations. The gateways dispatch client requests to the Internet. 166 gateways for Starlink were collected from a website [1] and used in our experiment.
Trace Datasets: We conduct trace-driven simulation using three real traces: MAWI [51], Wikipedia [46], and CAIDA [3]. MAWI and CAIDA are packet traces collected in monitored links in Japan and the US, respectively, and are filtered to only include web requests by checking the protocol and port used in packets. The Wikipedia dataset is collected from a west-coast machine in the US which serves multi-media content for Wikipedia. This dataset does not contain geographical information of visitors, thus we randomly assign the requests to each US state based on population distribution and exclude Alaska and Hawaii, as they are out of the service region of Starlink or ViaSat. We only keep the top 10 most visited content and 1-hour trace for all of these three datasets. In addition, we also use more datasets, which will be introduced in specific experiments.
Sampled Real Latency: We consider 3 types of metrics: hop count, ideal latency and sampled real latency. Hop count is straightforward: satellite-to-user, satellite-to-gateway, and every inter-satellite link are all counted as 1 hop each. Ideal latency is calculated using physical distance and the speed of transmission. Additionally, real latency measurements were conducted in a Starlink network. We acquire the latency of the connection between the user and a gateway by using traceroute. We measure these latency numbers for 1 day. In our experiments, we can randomly sample the latency from the measured one links between the satellites and the ground.
Replication Cost Ratio: Query cost can be measured by latency or hop count. Besides, proper settings for replication cost and storage cost are also necessary. The transfer cost between nodes v_1 and v_2 is set to be α times of the query cost between these nodes, where $\alpha > 1$. The intuition here is that replicating between nearby satellites is cheaper and more convenient than between remote satellites. To avoid frequent replica position changes, $\alpha > 1$ is used, with $\alpha = 50$ as the default value in experiments. A value of $\alpha = 1$ would result in replicating the origin server to a nearby satellite for each user, yielding the same cost as direct querying from the origin server. A larger α requires a careful selection of replica positions to serve a large number of users for a long time.
Storage Cost Ratio: We define the smallest query cost across all time stamps to be c_{qmin}. It will be 1 or several milliseconds if hop count or latency is used as metrics, respectively. The storage cost is set to $\beta \cdot c_{qmin}$ for gateways and $\gamma \cdot c_{qmin}$ for satellites, where $\gamma > \beta$. We refer to β and γ as storage ratios. The difference between γ and β reflects the cost difference between terrestrial and satellite storage. Usually satellite storage is much more expensive than terrestrial storage. In our evaluation, $\beta = 1$ and $\gamma = 10$ are used based on the data from [25], which shows that transferring 1 GB of data costs 0.1$ for terrestrial CDN operators and 1$ for satellite network providers.

6.2 Algorithm Evaluation

6.2.1 Baseline methods. In this section, we introduce the baseline methods. In general, our baselines contain not only heuristic algorithms for UFL [8, 20], but also algorithms designed for a satellite network [25, 37].
Algorithms for UFL: Our problem can be reduced to a UFL problem if only one time slot is considered. Therefore, we can employ an algorithm of UFL and apply it at each time slot independently and ignore the relationship across multiple timeslots, which may lead to large replication traffic.

We evaluate the following three popular UFL algorithms: First, a naive greedy algorithm tries to minimize the total cost every time by adding a new facility until the total cost cannot be reduced. Second, a smarter greedy algorithm proposed in [20] uses an average cost instead of the total cost as the greedy metrics, and assigns users to facilities while considering the average opening cost and transport cost to unassigned users and the reduction in transport cost for

[1]The website is starlink.sx

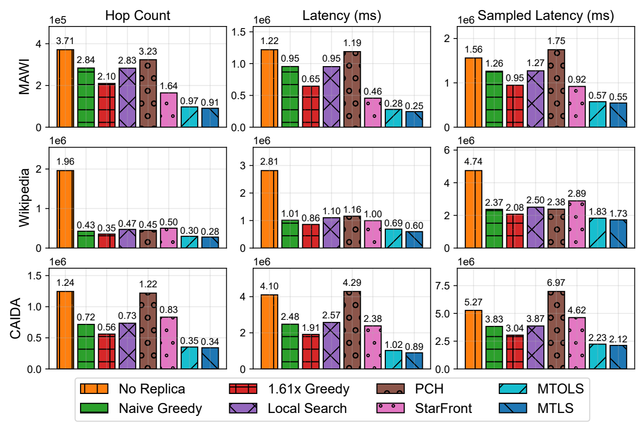

Figure 4: Evaluation of different methods based on 3 real traces (MAWI, Wikipedia, CAIDA) is presented in the figure. The total cost of various metrics is displayed in columns.

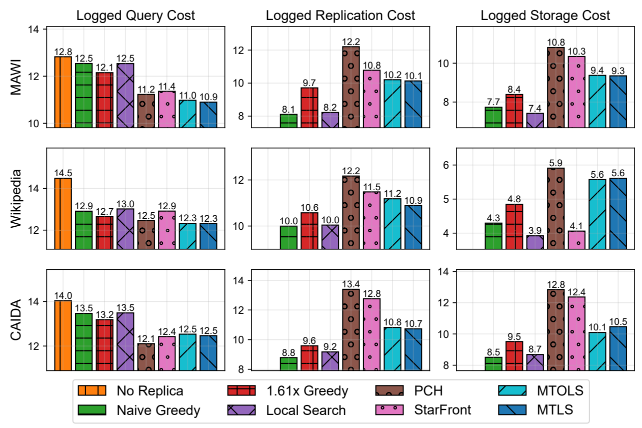

Figure 5: This figure shows the breakdown of the total cost: Query cost, replication cost and storage cost. The metric is hop count.

assigned users after a new facility is established. This algorithm has been shown to be a 1.61-approximation of the UFL problem and will be referred to as the "1.61x greedy" in our experiments. Third, a local search algorithm [8] of UFL has an approximation factor of 3 for the UFL problem, as proved in [17]. Some other algorithms based on linear programming rounding reach lower approximation factors [10, 29]. We do not include these methods because we have thousands of satellites and running linear programming on these many facilities will take too long.

StarFront: [25] proposes StarFront to set up replicas on satellites and cloud servers to minimize the latency. The algorithm needs a latency threshold for all users and it chooses replicas for every user to satisfy the latency threshold. If multiple replicas satisfy the threshold, the one with the smallest sum of replication cost and storage cost will be chosen. Our metrics include not only latency but also hop count. Thus we extend this algorithm to support hop count. We set an appropriate threshold by minimizing the total cost.

Periodic Cache Handoff (PCH): [37] proposes a heuristic to manage satellite cache. It will place the cache near the end users first. Then the cache will be replicated to the next satellite on the same orbit periodically. There is also inter-orbit cache replication with a lower frequency. We refer to this method as periodic cache handoff (PCH).

6.2.2 *Results of Different Metrics.*

Our results, shown in Figure 4, demonstrate the effectiveness of our proposed methods, MTLS and MTOLS. The experiments include LEO satellites and gateways in the network structure.

Among the evaluated algorithms, our methods consistently achieved the lowest total cost across 3 trace datasets and 3 metric types. MTLS outperformed all other algorithms with a reduction in total cost ranging from 16.91% to 53.26% compared to the strongest baseline. MTOLS also showed favorable performance compared to all baselines.

The total cost of each method is decomposed into query cost, replication cost, and storage cost, as shown in Figure 5. It reveals that UFL algorithms typically have lower replication and storage costs, but higher query costs. This is due to the UFL algorithms' focus on demand within each time slot without considering future demand. They are designed to optimize for static networks and cannot handle dynamic network and dynamic user demands. In contrast, PCH has low query costs but high replication and storage costs. As noted in [37], the intra-orbit replication of PCH takes place every 4.3 minutes for all replicas, which will incur high replication cost. StarFront has low query costs but high replication and storage costs because it does not allow for replica changes during periods. Our methods balance all costs effectively. MTLS has lower replication costs than MTOLS. The reason may be that MTOLS does not have replacement operations but MTLS has it when generating the nearby set.

6.2.3 *Oracle v.s. Prediction of Demand.*

In Section 6.2.2, the experiments used oracle information of demand to select replicas, but this information is not always available in real-world scenarios. We conduct additional experiments without oracle information, but using averaged historical demand data over the past 5 minutes to predict future demand. This prediction is used as input for all algorithms except PCH, which is rule-based and does not require predictions. Figure 6 shows that the gap between PCH and other methods narrows, but our methods continue to outperform the other algorithms. This demonstrates their robustness.

6.2.4 *Computation time.*

We show the computation time on the MAWI dataset in Table 1. All methods are evaluated on

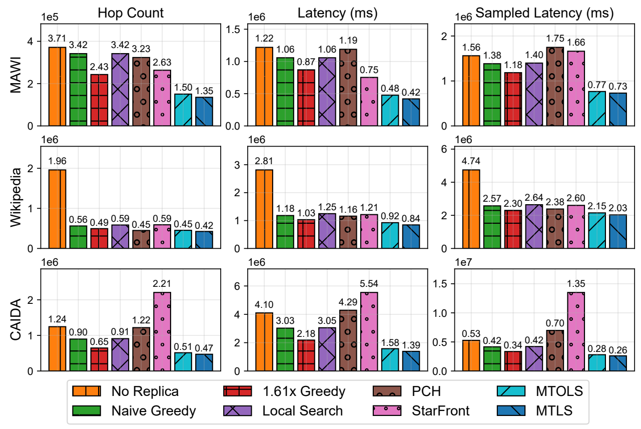

Figure 6: Evaluation of methods without oracle information of demand. Predicted demand based on average historical values is used for each method.

Table 1: Computation time of different methods on the MAWI dataset.

Method	Computation Time (sec)
No Replica	0.0
Naive Greedy	20.9
1.61x Greedy	30.1
Local Search	33.9
PCH	0.3
StarFront	13.8
MTOLS	495.3
MTLS	98,576.3

the same machine with Intel Xeon CPU E5-2690 v3 CPU and 220 GB memory size. Although the CPU has 24 vcores, we limit the program to use only up to 1 vcore for comparison. We find that MTOLS can have 200x speedup over MTLS by incorporating orbit information. It can be further accelerated using parallel execution. We can optimize the placement of different content in parallel and make different processes to handle different subsets in the DP phase in parallel.

6.2.5 Combining Different Satellite Constellations. We compare the performance of three types of satellite networks: LEO (Starlink), MEO (O3b), and GEO (ViaSat). Figure 7 visualizes the coverage map for these three types of satellites. For a fair comparison, we set a unified minimum elevation angle (10°) for all satellites. We also remove all gateways from the network structure. The experiment is conducted using a simulated dataset of requests from a 5x10 grid region in the US over 4 hours, with a time slot gap of 5 minutes.

The results indicate that the choice of satellites depends on storage cost factors and the chosen metric: either hop count or latency. When storage cost factors for LEO, MEO,

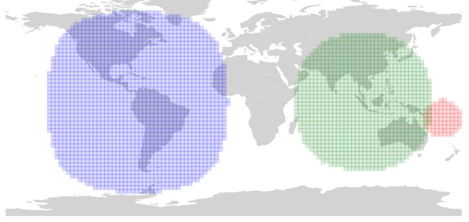

Figure 7: The coverage map of one Starlink (LEO), O3b (MEO), and ViaSat (GEO) satellite, colored in red, green, and blue, respectively. We set the minimum elevation angle to 10 degree for all satellites for fair comparison.

Table 2: The results of integrating MEO and GEO in the network, with MTLS determining satellite selection based on different storage cost factors. The results include query cost, replication cost, and storage cost in the last three columns, with the time ratio of MEO and GEO usage shown in the "MEO" and "GEO" columns.

γ_{meo}	γ_{geo}	MEO	GEO	Qry.	Rep.	Sto.
0	10	100.0%	0.0%	18,060	300	0
0.5	10	100.0%	0.0%	18,060	300	18
1	10	38.9%	61.1%	18,010	150	234
2	10	19.4%	80.6%	18,000	100	304
3	10	0.0%	100.0%	18,000	50	360
5	10	0.0%	100.0%	18,000	50	360
10	10	0.0%	100.0%	18,000	50	360

Table 3: The results of integrating LEO and MEO in the network, with MTLS determining satellite selection. We set different storage cost factor γ for LEO and MEO: $\gamma_{leo} = 1$, and $\gamma_{meo} = 25$. The results include query cost, replication cost, and storage cost in the last three columns, with the time ratio of LEO and MEO usage shown in the "LEO" and "MEO" columns.

Region	LEO	MEO	Qry.	Rep.	Sto.
1x2	100.0%	0.0%	24,000	1,200	48
2x4	39.6%	60.4%	24,000	600	744
3x6	2.1%	97.9%	24,000	300	1,176
4x8	2.1%	97.9%	24,016	300	1,176
5x10	2.1%	97.9%	24,080	450	1,176

and GEO are identical, the preferred network is GEO for hop count metric because of its large coverage range, and LEO for latency metric due to its close distance to the ground.

Evaluation using the hop count metric with different storage cost factors for MEO and GEO shows that a reduction in MEO's storage cost factor leads to more frequent use of MEO, as shown in Table 2. The coverage areas of MEO and GEO

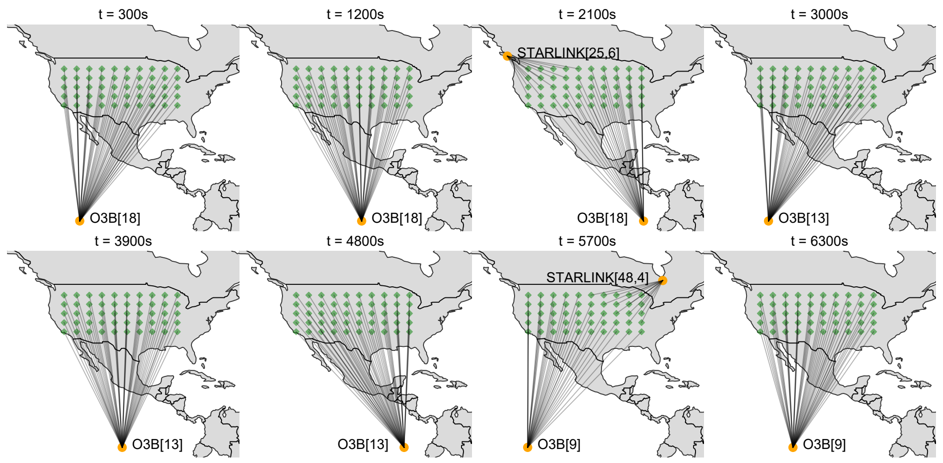

Figure 8: We put LEO and MEO satellites in the same network, and let MTLS decides which kind of satellites to use. We set different storage cost factor γ for LEO and MEO: $\gamma_{leo} = 1$, and $\gamma_{meo} = 50$. In this scenario, LEO satellites and MEO satellites can be cooperatively deployed as replicas to cover all of the demand.

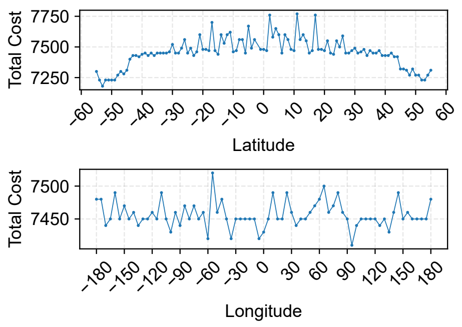

Figure 9: One client is set at different locations and MTLS is used to calculate total cost for serving it. We place the client at longitude = 0° with different latitudes, and also latitude = 40° with different longitudes.

satellites are similar, however, MEO satellites move slowly and require content replication to the next satellite when moving away from the US. The likelihood of using MEO increases when the reduction in storage cost compensates for the increase in replication cost.

MEO has a larger coverage area than LEO. We examine the impact of the geographical distribution of demand on satellite selection by limiting demand to 1x2, 2x4, 3x6, 4x8, and 5x10 out of the 5x10 grid region. Storage cost factors of LEO and MEO are set at 1 and 25, respectively. Table 3 shows that LEO is preferred for serving small regions of demand, while MEO is preferred for serving larger regions. Meanwhile, if the storage cost of LEO is low enough, both LEO and MEO can be cooperatively deployed as shown in Figure 8.

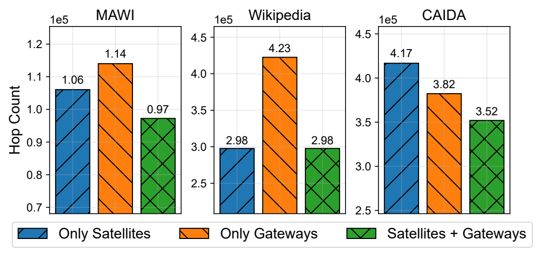

Figure 10: Results of limiting MTLS to choose replicas only from gateways or only from satellites. Hop count is used as a metric.

6.2.6 *Effects of Geographical Locations.*

We study the total cost for serving a single client placed at varying latitudes and longitudes using MTLS with only Starlink satellites and no gateways. Figure 9 shows that higher latitudes have less total cost than lower latitudes, which is likely due to the increased visibility of satellites [32]. However, increased total cost at very high latitude (larger than 50° or less than −50°) is likely due to proximity to the service border of Starlink satellites. No obvious trends are observed for longitudes.

6.2.7 *Effects of gateways.*

To examine the impact of gateways, we maintain the same network structure but force the algorithm to deploy replicas only on gateways or only on satellites. Figure 10 shows that using both gateways and satellites is preferred for the MAWI and CAIDA datasets. On the Wikipedia dataset, utilizing only satellites is sufficient. The reason is that the requests of the Wikipedia dataset are concentrated in the US and LEO satellites alone are sufficient to cover its demands. One limitation of a satellite replica is its limited coverage area. This disadvantage is mitigated

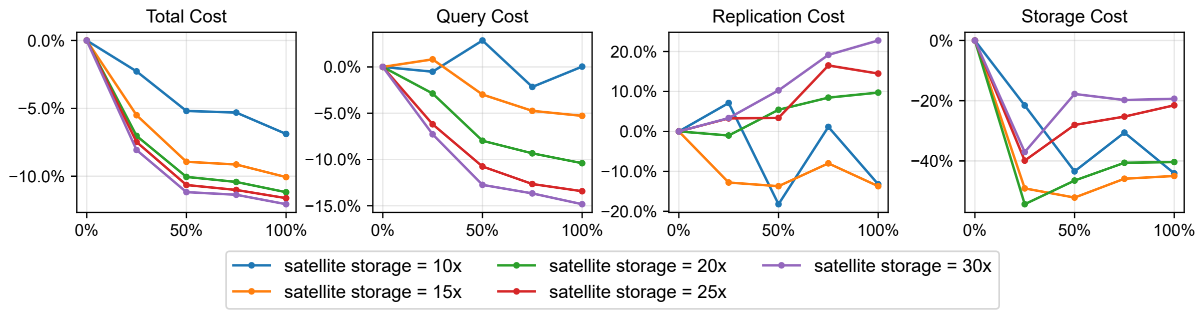

Figure 11: Results of adding new gateways near clients. In the experiment, we place new gateways near 0%, 25%, 50%, 75%, and 100% of the clients. Satellite storage is set to be 10, 15, 20, 25, and 30 times of gatway storage. CAIDA is used as the assessed dataset and hop count is employed as the metric.

in the case of concentrated requests, e.g., on the Wikipedia dataset. The tradeoff between gateways and satellites is that satellites can provide less latency, less hop count but larger storage cost and moving coverage area, while gateways can provide smaller storage cost, static coverage area but more latency and hop count.

We further evaluate the impact of adding gateways to a satellite network using the CAIDA dataset. We use hop count as the metric. Figure 11 demonstrates that adding gateways reduces the overall cost especially when the cost of satellite storage is much higher than gateway storage. With moderate satellite storage costs (e.g., 10× or 15×), satellites are used as intermediate transfer points: The replica is first transmitted to a satellite, then the satellite travels to a specific region, and finally the replica is unloaded from the satellite to a gateway in that region, thereby reducing the replication cost. Our automatically generated solution is similar to the proposal in [14]. But different from [14], our algorithm is more general and can adapt to different scenarios. The above strategy is less preferred when the satellite storage becomes more expensive (e.g., 20x, 25x, 30x). In this case, our algorithm prioritizes deploying replicas on more gateways, which leads to increased replication cost but decreased query cost.

6.2.8 Video Delivery. We test the performance of different methods for video delivery. We create video demand by randomly generating video chunk downloading requests and assigning them to different states of the US according to population distribution. We exclude Alaska and Hawaii.

The download time of each chunk is calculated as the sum of propagation delay and transmission delay, where propagation delay is determined by distance and transmission delay is determined by chunk size and throughput. We assume 20 Gbps for terrestrial links and 10 Gbps for satellite links.

Besides redirecting the user to the closest server, we use round robin and weighted round robin for redirection. We take $n = 3$ for these 2 strategies. For weighted round robin, the 3 closest replicas will get $\frac{4}{7}$, $\frac{2}{7}$, and $\frac{1}{7}$ traffic, respectively.

Table 4 summarizes our results. MTOLS and MTLS usually have better QoE than the baselines because our methods can generate relatively well-distributed replicas, which results in a more balanced traffic distribution in the network. Although the local search has lowest traffic but it results from its low QoE. The traffic of our methods is lower than all baselines except local search when "closest" or weighted round robin is used as the traffic strategy. For most algorithms, a weighted round robin is better than "closest" and round robin.

6.3 System Evaluation

We run our network probe depicted in Figure 3 within a satellite network provided by Starlink RV in Texas. The network trace shows that the bandwidth to the origin server is 4.90±1.12 Mbps, while the bandwidth to the closest Akamai server is 11.76±3.11 Mbps. The latency between our client and the closest Starlink ground station is 129.79±92.2 ms. We apply the trace to the connections between the clients and servers in our prototype system. Our origin server is provided by a cloud server and is located in Asia. It has 16GB memory and 2 CPU vcores. Our client machine is in Texas with 64GB memory and 24 CPU vcores.

6.3.1 Web Browsing Results. In order to optimize the web browsing experience, real-world CDNs are utilized to host static web content, such as images, CSS files, HTML documents, and fonts on replicas to minimize the page load time experienced by end users. Our study assesses the proposed system's performance for web browsing by letting clients download a static item of size 16KB from the servers in our prototype system. We generate simulated requests from the top 50 populated cities on Earth. The replica placement is determined by applying various algorithms to the dataset. The resulting replicas are deployed in the server pool, and

Table 4: The results of the simulated video delivery. The Mean QoE denotes the average quality of experience at each time slot, and the Traffic GB indicates the cumulative transmitted traffic in all links. If one piece of content is transmitted through multiple links, the traffic will be counted for multiple times.

Method	**Replica Num**	**Closest**		**Round Robin**		**Weighted Round Robin**	
		Mean QoE	Traffic GB	Mean QoE	Traffic GB	Mean QoE	Traffic GB
No Replica	1.0	1.70	963,159	1.70	963,159	1.70	963,159
Naive Greedy	6.5	5.40	308,808	4.19	405,530	5.12	404,049
1.61x Greedy	11.4	6.25	320,389	5.40	425,899	6.28	412,831
Local Search	5.3	4.82	**293,368**	3.35	**376,532**	4.29	**387,299**
PCH	18.5	6.41	304,255	6.33	416,780	6.89	397,025
StarFront	26.0	3.18	318,911	4.75	476,172	4.51	447,326
MTOLS	16.9	**6.63**	295,152	**6.59**	416,093	**7.12**	389,084
MTLS	18.3	**6.63**	295,313	6.54	418,526	7.08	389,732

clients in the user pool are instructed to request from the servers. The download time is recorded and is presented in Table 5. The cumulative distribution function (CDF) plot of download time is also shown in Figure 12. Our methods have the lowest total cost, and MTLS has the smallest median download time. In addition, MTOLS exhibits a similar download time to PCH, but with a more efficient deployment of 25.0 replicas compared to PCH's 34.9 replicas.

6.3.2 Video Streaming Results. Furthermore, we also evaluate the performance of our system for video streaming. We adopt similar settings to Section 6.2.8 but make the following two modifications to the settings: (1) we limit the capacity of each server to 96 Mbps, instead of applying a capacity limit on connections, and (2) we randomize the start time for each user's video stream. This allows us to reduce the total number of concurrent requests and make the experiments practicable on our machines. Table 6 summarizes our results. Our methods – MTOLS and MTLS out-perform the other algorithms in terms of total cost and QoE. While MTOLS and MTLS have similar query costs (2055 and 2025), they differ in terms of replication cost (343 and 250) and QoE (9.00 and 9.15). Our hypothesis is that optimizing the replication cost in the network can lead to a more balanced distribution of traffic, thereby improving the overall QoE.

6.4 Summary

In summary, in Section 6.2.2, 6.2.3, and 6.2.4, we demonstrate that MTOLS and MTLS proposed in this paper can generate better replication with lower total cost than all the baselines. Our methods are robust when future demand information is unknown and predicted demand is used. MTLS has a long computation time but MTOLS is much faster. These two methods can be further accelerated through parallel execution. In Section 6.2.5, 6.2.6, 6.2.7, and 6.2.8, we also reveal key insights for optimizing in different satellite network settings, such as the benefits of combining LEO, MEO, and GEO

Table 5: Performance comparison of various methods for web browsing in the prototype system. The download time of a file from the server is measured. Median download time is recorded and represented as "Time (ms)". The average number of deployed replicas in each time slot is represented as "Num". StarFront[MTLS] and StarFront[MTOLS] indicate that we align the number of replicas set by StarFront to be the same with MTLS and MTOLS.

Method	**Total Cost**	**Time (ms)**	**Num**
No Replica	29,241,304.1	2,093.3	1.0
Naive Greedy	211,829.0	179.8	8.3
1.61x Greedy	185,017.8	154.4	10.9
Local Search	262,370.6	188.1	4.4
PCH	188,536.3	111.9	34.9
StarFront	192,022.7	129.5	43.0
StarFront [MTOLS]	177,399.1	140.7	26.0
StarFront [MTLS]	186,532.7	133.5	38.0
MTOLS	102,664.6	112.4	25.0
MTLS	**87,665.2**	**99.7**	37.5

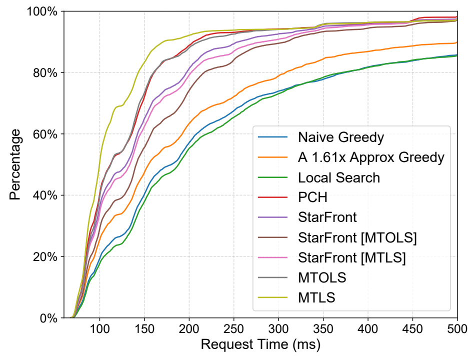

Figure 12: CDF plot of download time of different methods in real implemented system.

Table 6: Performance comparison of various methods for video streaming in the prototype system. The average number of deployed replicas in each time slot was represented as "Num". StarFront[Aligned] means we align the number of replicas set by StarFront to be the same with MTLS and MTOLS.

Method	Total Cost	Mean QoE	Num
No Replica	30628.5	1.56	1.0
Naive Greedy	2612.1	8.00	6.3
1.61x Greedy	2431.3	8.74	8.2
Local Search	3058.2	7.23	4.8
PCH	2714.8	8.93	17.6
StarFront	3634.6	8.70	29.0
StarFront [Aligned]	4386.2	6.84	11.0
MTOLS	2403.5	9.00	10.4
MTLS	**2281.0**	**9.15**	10.8

satellites, the impact of user latitude, the importance of colocating replicas on both gateways and satellites, and the advantages of using weighted round robin strategy in the video delivery scenario. In Section 6.3, we further demonstrate the effectiveness of our methods using a prototype implementation for both web browsing and video streaming using real traces collected from the Starlink network.

7 LIMITATIONS

In this section, we describe a few limitations of our work. First, our synthetic trace could be further improved to produce more realistic traces, such as modeling the spatial and temporal correlation between user demands and satellite throughput fluctuation. Second, we can further improve the diversity of our satellite and Web traces for a more comprehensive evaluation. Last but not least, there are several practical considerations that must be addressed before our methods can be deployed in the real world. In order to realize our proposed system in the real world, new satellites with storage and computing device must be available. Moreover, we need to optimize DNS servers to reduce DNS query overhead.

8 RELATED WORK

The existing work can be broadly classified into (i) CDN optimization, and (ii) satellite networks.

Replica Placement for Traditional CDN: Optimizing CDN server placement has attracted significant research. They aim to optimize the placement of the server replicas or individual content under certain constraints (e.g., bandwidth, latency, the number of replicas). They formulate the problems as theoretical models, including facility location [42, 49, 54], K-median [27, 28, 39, 50], K-center [21, 22], K-cache [23], etc. In general, the placement problems are NP-hard. Various algorithms have been proposed. They differ in terms of performance metrics, types of constraints, target networks, and effectiveness. Refer to [44] for a detailed survey. These algorithms all target for static networks and can not be applied to LEO or MEO satellite networks. [25] and [37] consider replica server or cache placement in LEO network but our methods can outperform them a lot as shown in the evaluation section. Moreover, our work explicitly leverages the orbit information to enhance both scalability and effectiveness and supports not only web pages but also video content, not only LEO but also MEO, LEO and their combinations.

Satellite Network: Recently there is a surge of interest in LEO satellite networks as the launch and hardware cost of LEO satellites rapidly decreases. A number of works are devoted to simulating, measuring and understanding LEO networks, such as NOAA, Starlink, and OneWeb. For example, [31] develops a simulator for Starlink and shows that it can provide lower latency than a terrestrial optical fiber network over 3000 km. [24] develops StarPerf to simulate mega-constellation and uses it to drive the design of constellation and relay selection. [30] analyzes the impact of inter-satellite links and reports that these links can significantly improve the performance of LEO networks.

[52] proposes adding low-cost ground stations that are only download capable to a LEO network, and develops a rate adaptation algorithm to support links without ACK feedback. [45] proposes to develop a technique that allows multiple nearby low-cost ground stations to collaboratively recover the LEO satellite signals. It reports 8 dB SNR increase over a large commercial receiver. [26] improves the delay and throughput in LEO networks by leveraging links between many satellites and ground stations. [1] proposes SPACERTC to select nodes for relaying real-time traffic and allocates the flows to reduce latency. [14] proposes a hybrid backhaul solution using a combination of terrestrial and satellite networks to reduce the placing time of popular content in 5G edge nodes. [25] is the closest related work to ours and also studies replica placement. As shown in our evaluation, our approach significantly outperforms [25].

9 CONCLUSION

In this paper, we develop novel replica servers for satellite networks. We leverage the deterministic satellite movement trajectories along with the user demands to move content close to the users when needed. Using both synthetic and real measurements from StarLink, we evaluate our approach across diverse network scenarios. Our results show our placement yields 16.91% to 53.26% reduction in the total cost while maintaining low query latency and high video QoE. Using prototype implementation and experiments, we further show the feasibility and effectiveness of our approach.

REFERENCES

[1] 2022. SpaceRTC: Unleashing the Low-latency Potential of Mega-constellations for Real-Time Communications. In *Proc. of IEEE INFOCOM.*

[2] 2022. SpaceX's first-Gen Starlink Fleet halfway complete after back-to-back launches. https://t.ly/Qn_G. (2022).

[3] 2023. CAIDA catalog. (2023). https://catalog.caida.org/

[4] 2023. FCC approves Amazon's satellite broadband plan over SpaceX's objections. https://arstechnica.com/tech-policy/2023/02/fcc-approves-amazons-satellite-broadband-plan-over-spacexs-objections/. (2023).

[5] 2023. OneWeb of 600 internet satellites in space. https://www.dw.com/en/oneweb-of-600-internet-satellites-in-space/a-47690785. (2023).

[6] 2023. Viasat (American company). https://en.wikipedia.org/wiki/Viasat_(American_company). (2023).

[7] Vijay Kumar Adhikari, Yang Guo, Fang Hao, Matteo Varvello, Volker Hilt, Moritz Steiner, and Zhi-Li Zhang. 2012. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *2012 Proceedings IEEE INFOCOM.* IEEE, 1620–1628.

[8] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. 2001. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing.* 21–29.

[9] Thomas Barnett, Shruti Jain, Usha Andra, and Taru Khurana. 2018. Cisco visual networking index (vni) complete forecast update, 2017–2022. *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation* (2018), 1–30.

[10] Jaroslaw Byrka. 2007. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007. Proceedings.* Springer, 29–43.

[11] cdfperf 2023. CDN performance. (2023). www.cdnperf.com.

[12] Aizaz U Chaudhry and Halim Yanikomeroglu. 2021. Laser intersatellite links in a starlink constellation: A classification and analysis. *IEEE Vehicular Technology Magazine* 16, 2 (2021), 48–56.

[13] Robert A Divine and Robert Alexander Divine. 1993. *The sputnik challenge.* Oxford University Press.

[14] Alexis A Dowhuszko, Juan Fraire, Musbah Shaat, and Ana Pérez-Neira. 2020. LEO satellite constellations to offload optical terrestrial networks in placement of popular content in 5G edge nodes. In *2020 22nd International Conference on Transparent Optical Networks (ICTON).* IEEE, 1–6.

[15] Barry G Evans, Paul T Thompson, Giovanni E Corazza, Alessandro Vanelli-Coralli, and Enzo Alberto Candreva. 2011. 1945–2010: 65 years of satellite history from early visions to latest missions. *Proc. IEEE* 99, 11 (2011), 1840–1857.

[16] Mohinder S Grewal, Lawrence R Weill, and Angus P Andrews. 2007. *Global positioning systems, inertial navigation, and integration.* John Wiley & Sons.

[17] Anupam Gupta and Kanat Tangwongsan. 2008. Simpler analyses of local search algorithms for facility location. *arXiv preprint arXiv:0809.2554* (2008).

[18] Mark Handley. 2019. Using ground relays for low-latency wide-area routing in megaconstellations. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks.* 125–132.

[19] Jinhui Huang and Jiang Cao. 2020. Recent development of commercial satellite communications systems. In *Artificial Intelligence in China: Proceedings of the International Conference on Artificial Intelligence in China.* Springer, 531–536.

[20] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. 2003. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM (JACM)* 50, 6 (2003), 795–824.

[21] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. 2000. On the placement of internet instrumentation. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064),* Vol. 1. IEEE, 295–304.

[22] Yichao Jin, Yonggang Wen, and Kyle Guan. 2016. Toward cost-efficient content placement in media cloud: Modeling and analysis. *IEEE Transactions on Multimedia* 18, 5 (2016), 807–819.

[23] Paddy Krishnan, Danny Raz, and Yuval Shavitt. 2000. The cache location problem. *IEEE/ACM transactions on networking* 8, 5 (2000), 568–582.

[24] Zeqi Lai, Hewu Li, and Jihao Li. 2020. StarPerf: Characterizing Network Performance for Emerging Mega-Constellations. In *Proc. of ICNP.*

[25] 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 *Proc. of ICNP.* IEEE, 1–12.

[26] Z. Lai, Q. Wu, H. Li, M. Lv, and J. Wu. 2021. OrbitCast: Exploiting Mega-Constellations for Low-Latency Earth Observation. In *Proc. of IEEE ICNP.*

[27] Nikolaos Laoutaris, Georgios Smaragdakis, Konstantinos Oikonomou, Ioannis Stavrakakis, and Azer Bestavros. 2007. Distributed placement of service facilities in large-scale networks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications.* IEEE, 2144–2152.

[28] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros. 2007. Distributed Placement of Service Facilities in Large-Scale Networks. In *Proc. of INFOCOM.*

[29] Shi Li. 2013. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation* 222 (2013), 45–58.

[30] Y. Li, H. Li, L. Liu, W. Liu, J. Liu, J. Wu, Q. Wu, J. Liu, and Z. Lai. 2021. Internet in Space for Terrestrial Users via Cyber-Physical Convergence. In *Proc. of ACM HotNets.*

[31] booktitle = Mark Handley". [n. d.]. Delay is Not an Option: Low Latency Routing in Space.

[32] Jonathan C McDowell. 2020. The low earth orbit satellite population and impacts of the SpaceX Starlink constellation. *The Astrophysical Journal Letters* 892, 2 (2020), L36.

[33] François Michel, Martino Trevisan, Danilo Giordano, and Olivier Bonaventure. 2022. A first look at starlink performance. In *Proceedings of the 22nd ACM Internet Measurement Conference.* 130–136.

[34] Pitu B Mirchandani and Richard L Francis. 1990. *Discrete location theory.*

[35] 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.

[36] Soohyun Park and Joongheon Kim. 2021. Trends in LEO satellite handover algorithms. In *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN).* IEEE, 422–425.

[37] Tobias Pfandzelter and David Bermbach. 2021. Edge (of the Earth) Replication: Optimizing Content Delivery in Large LEO Satellite Communication Networks. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid).* IEEE, 565–575.

[38] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2018. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156* (2018).

[39] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. 2001. On the Placement of Web Server Replicas. In *Proc. of INFOCOM*.

[40] Zhicheng Qu, Gengxin Zhang, Haotong Cao, and Jidong Xie. 2017. LEO satellite constellation for Internet of Things. *IEEE access* 5 (2017), 18391–18401.

[41] P Krishna Rao, Susan J Holmes, Ralph K Anderson, Jay S Winston, and Paul E Lehr. 1990. Weather satellites: Systems, data, and environmental applications. (1990).

[42] Georgios Rodolakis, Stavroula Siachalou, and Leonidas Georgiadis. 2006. Replicated server placement with QoS constraints. *IEEE Transactions on Parallel and Distributed Systems* 17, 10 (2006), 1151–1162.

[43] Jagruti Sahoo, Mohammad A Salahuddin, Roch Glitho, Halima Elbiaze, and Wessam Ajib. 2016. A survey on replica server placement algorithms for content delivery networks. *IEEE Communications Surveys & Tutorials* 19, 2 (2016), 1002–1026.

[44] Jagruti Sahoo, Mohammad A. Salahuddin, Roch Glitho, Halima Elbiaze, and Wessam Ajib. 2016. A Survey on Replica Server Placement Algorithms for Content Delivery Networks. *IEEE Communications Surveys and Tutorials* (2016).

[45] V. Singh, A. Prabhakara, D. Zhang, O. Yagan, and S. Kumar. 2021. A community-driven approach to democratize access to satellite ground stations. In *Proc. of ACM MobiCom*.

[46] Zhenyu Song, Daniel S Berger, Kai Li, and Wyatt Lloyd. 2020. Learning relaxed belady for content distribution network caching. In *17th USENIX Symposium on Networked Systems Design and Implementation*.

[47] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions On Networking* 28, 4 (2020), 1698–1711.

[48] Yongtao Su, Yaoqi Liu, Yiqing Zhou, Jinhong Yuan, Huan Cao, and Jinglin Shi. 2019. Broadband LEO satellite communications: Architectures and key technologies. *IEEE Wireless Communications* 26, 2 (2019), 55–61.

[49] Jihoon Sung, Minseok Kim, Kyongchun Lim, and June-Koo Kevin Rhee. 2013. Efficient cache placement strategy for wireless content delivery networks. In *2013 International Conference on ICT Convergence (ICTC)*. IEEE, 238–239.

[50] Michal Szymaniak, Guillaume Pierre, and Maarten Van Steen. 2006. Latency-driven replica placement. *IPSJ Digital Courier* 2 (2006), 561–572.

[51] the WIDE Project. [n. d.]. MAWI Working Group Traffic Archive. ([n. d.]). https://mawi.wide.ad.jp/mawi/

[52] Deepak Vasisht, Jayanth Shenoy, and Ranveer Chandra. [n. d.]. L2D2: Low Latency Distributed Downlink for Low Earth Orbit Satellites.

[53] Limin Wang, Vivek Pai, and Larry Peterson. 2002. The effectiveness of request redirection on CDN robustness. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 345–360.

[54] Hao Yin, Xu Zhang, Tongyu Zhan, Ying Zhang, Geyong Min, and Dapeng Oliver Wu. 2013. NetClust: A framework for scalable and pareto-optimal media server placement. *IEEE Transactions on Multimedia* 15, 8 (2013), 2114–2124.

[55] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 325–338.