Cache Allocation Scheme in Information-Centric Satellite-Terrestrial Integrated Networks

Jie Duan(✉), Xianjing Hu, Hao Liu, and Zhihong Zhang

School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing 400065, China
duanjie@cqupt.edu.cn

Abstract. With the development of satellite communication technologies, satellite-terrestrial integrated network (STIN) is often used for content delivery services as satellites are with wide-area coverage. In terrestrial networks, in-network caching has been proved to be an effective method to improve network performance in terms of throughput and delay, so information-centric networking (ICN) will be deployed in STIN architecture (STI^{2}CN). However, the current cache research scenarios in satellite networks usually ignore cache cooperation among nodes, resulting in lower cache hit rate. Secondly, the cache configuration is mainly implemented in the coverage area, which does not consider satellites mobility and dynamic network topology, so that network transmission efficiency is decreased. Thus, we formulate cache content allocation problem as user average hop count minimization problem, which is solved by matching algorithm to obtain exchange stability between satellites and contents. In addition, a proactive pushing scheme is proposed to solve satellites mobility problem. The cached contents will be pushed to the subsequent satellite covering the area in advance to further improve content retrieval efficiency. Simulation results show that the proposed scheme can significantly reduce user average hop count and improve cache hit rate.

Keywords: Satellite-terrestrial integrated network · Information-centric networking · Cache allocation · Proactive pushing

1 Introduction

As the popularization of mobile communication devices, such as smartphones and tablet PCs, making user requirements for pervasive network access any-

This work was supported by National Natural Science Foundation of China (NO. 61701058), Foundation of key Laboratory in National Defense Science and Technology for Equipment pre-research in the 13-th Five-year Plan (NO. 61422090301), Science and Technology Research Program of Chongqing Municipal Education Commission (NO. KJQN201800633).

H. Gao et al. (Eds.): ChinaCom 2021, LNICST 433, pp. 192–205, 2022.
https://doi.org/10.1007/978-3-030-99200-2_16

time anywhere become more intense. However, the current network infrastructures that provide Internet services are made up of terrestrial devices, which are prone to be damaged in disasters to interrupt communication. In addition, it is inconvenient to deploy network infrastructures in the remote and isolated areas such as deserts, seas and forests. To address the above problems, a new network architecture that integrates satellites into the terrestrial network is proposed—satellite-terrestrial integrated network (STIN) [1–3]. In STIN, satellites can provide services in extreme environments without terrestrial infrastructures, which makes it a strong supplement to traditional terrestrial networks [4,5].

The current Internet traffic is increasing exponentially. From the perspective of content retrieval, users just concern about contents themselves, not the servers or hosts where contents are located. In this case, applying the traditional TCP/IP communication mechanism to STIN will have problems such as low content delivery efficiency, poor mobility and scalability [6]. Information-centric networking (ICN) [7,8] has emerged as one of the promising candidates for the architecture of the future Internet, which has two major characteristics: in-network caching and routing-by-name. These characteristics enable each node to cache passing-by data and reduce redundant transmissions. Therefore, establishing an information-based satellite-terrestrial integrated information-centric network (STI^2CN) can solve a series of problems mentioned above. In-network caching has drawn lots of attention in satellite networks and proved its effectiveness in both improving transmission efficiency and reducing delay [9–15].

The critical characteristics of integrating satellites into ICN architecture were analyzed in [9] and [10], which provide a preliminary direction for exploring caching research in satellite communication systems. Recently, ICN architecture was applied in satellite-assisted emergency communications scenarios to solve mobility and link interruption problems, while effectively reducing end-to-end delay [11]. Literature [12] proposed caching strategy named SatCache in satellite networks to maximize cache hit ratio on the interest profile of users. In fact, with the rapid development of technologies such as on-board processing and storage capabilities, [13] first proposed the caching on-satellite to further improve network performance. However, a two-layer caching model based on genetic algorithm heavily depends on the initialization parameters. In [14], a scheme based on simulated annealing optimization algorithm was proposed in 5G-satellite backhaul networks. Although [13] and [14] studied the cache allocation algorithm for on-board cache, the research scenarios were limited to single-satellite nodes. Thus, a novel caching strategy based on matching game to minimize content access delay was proposed in [15]. But the work mainly focuses on optimal caching content placement at a certain moment, and not considering satellite network topology time-varying characteristics.

All these above have made contributions to solve cache in satellites. However, the research scenarios of existing cache schemes are limited to a single satellite node, non-cooperation among satellites causes cache redundancy. Secondly, the current algorithms have high complexity and slow convergence in a large-scale satellite network. Motivated by the above observations, the paper

considers user preferences and cache content cooperation, and formulates cache content allocation problem as user average hop count minimization problem. Since the formulated optimization problem contains 0–1 variables and it cannot be solved conveniently, we propose a matching-based cache allocation algorithm with fast convergence and low complexity, and finally obtains exchange stability between satellites and contents. In addition, the mobility of LEO satellites incurs users' requests not being responded to in time, which will increase user delay. Therefore, in the case of in-network caching, a proactive pushing scheme is proposed. The cached contents will be pushed to the subsequent satellite covering the area in advance to further improve network transmission efficiency.

The rest of the paper is organized as follows. Section 2 introduces our STI^{2}CN scenarios. In Sect. 3, the cache allocation problem is described and formulated. Then, we propose a matching-based cache allocation algorithm and proactive pushing scheme. Performance analysis are shown in Sect. 4. At last, conclusions are drawn in Sect. 5.

2 Network Scenarios

In this paper, when it is difficult for the conventional terrestrial network to achieve coverage and communication in some remote areas, STI^{2}CN providers Internet access services for users. As shown in Fig. 1, STI^{2}CN mainly comprises two parts: the terrestrial network and satellite network. The terrestrial network is the network currently in use, which is mainly composed of users, ground stations and content servers. It can provide high data rates to users, but the network coverage in rural and remote areas is limited. The satellite network is a strong supplement to terrestrial network, which consists of geostationary earth orbit (GEO) and low earth orbit (LEO) satellites. GEO satellites are relatively stationary with the ground, while exhibiting the highest propagation delay due to the long distance. Thus, GEO satellites are considered as controller to in charge of routing strategy calculation, mobility management and so on. By comparison, LEO satellites are close to the ground and have a short propagation delay, which mainly realize in-network caching and provide wireless access to users for obtaining long-distance global communications. However, due to the high-speed movements of LEO satellites, data transmission in such dynamic environment suffers frequently periodic interruptions, leading to lack of persistent end-to-end paths. Therefore, users and LEO satellites are constantly handover to ensure network connection. Normally, when the LEO satellite moves to another area, users' incomplete data transmissions can be migrated under the guidance of the GEO satellites. The following will briefly illustrate the process of user communication and content pushing with Fig. 1 as an example.

Firstly, users obtain contents as follows: when a user sends a request for a file, the ground station will search for the file in its local cache. If ground station already caches the file, it will respond to the request directly. Otherwise, the request will be sent to LEO satellites. Then, LEO satellites check whether the desired file is currently stored in the cache. If cache hits, LEO satellites will

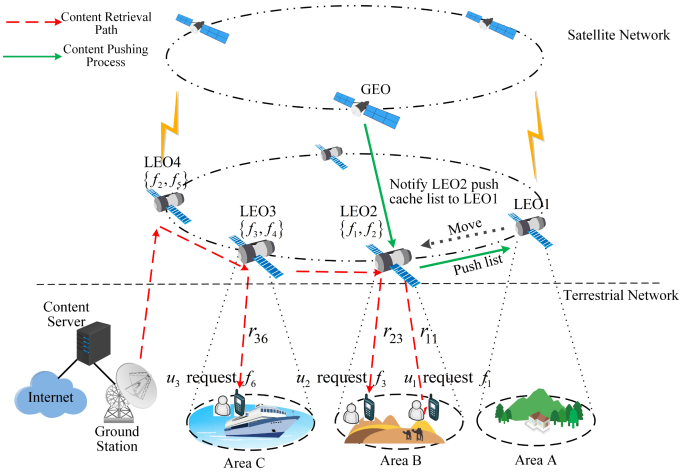


Fig. 1. Network scenario.

return the data to user along the requested path. Otherwise, they pass the file request to the next hop towards the destination of the requested file. Compared to space network, user delay in ground network is negligible, so it will not be discussed in detail. The red dashed arrows in Fig. 1 represent content retrieval path. The first case is that user u_1 acquires the file f_1 from the access satellite LEO2, and content retrieval path is r_{11}. In case of user u_2 requesting file f_3, it will obtain the requested file through inter-satellite links(ISL) from LEO3, content retrieval path is r_{23}. The user u_3 gained file f_6 from content server in the last case, content retrieval path is r_{36}.

In addition, data transmission will often be interrupted due to the mobility of LEO satellite nodes. Generally speaking, the file is waiting to be issued in the caching of the access satellite via ISL. However, ground-to-satellites links (GSL) may be insufficient in the satellite communication, resulting in the link being interrupted before file is not completely delivered, and the subsequent satellite covering the area will transmit file for user. If the file is not contained in the caching of the subsequent satellite, which will request the file from other satellites and cause longer delay. Hence, the cached contents will be pushed to the subsequent satellite in advance. The specific pushing process is shown by the green solid arrows in Fig. 1. GEO detects that LEO2 is about to leave coverage area B, GEO notifies LEO2 to push the cache list to LEO1, and LEO1 sends interest packet to LEO2 to request cached contents.

3 Dynamic Cache Allocation Scheme

3.1 Problem Description

For users, the desired contents can be obtained from content sources, which include source nodes that publish contents (i.e., satellite nodes or content servers

in ground network) and nodes that cache content (i.e., cache nodes). Users experience much less hop count to obtain contents from cache nodes than source nodes. Therefore, it is necessary to reasonably allocate contents to cache nodes, which reduce user hop count and improve network transmission efficiency.

The existing terrestrial network cache mechanisms adopt mostly the design ideas of online caching or centralized computing, which exhibit high computational complexity or high maintenance overhead, these methods are not obviously suitable for STI^{2}CN. When designing cache strategy for satellite networks, the following factors need to be considered. Firstly, the limited cache and processing capacity of satellite nodes requires that the cache strategy cannot be excessive complexity. Secondly, with high inter-satellite latency and data processing rate of cache nodes, the cache strategy needs to consider the cooperation among nodes. Finally, due to the high-speed movement of satellites, satellite's coverage area and request distribution vary greatly. In order to improve cache hit rate, contents cached by nodes should also be changed accordingly. Therefore, it is necessary to propose a cache allocation strategy that meets dynamic network topology and lower overhead.

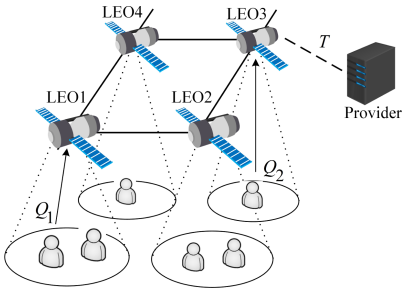


Fig. 2. Example problem description.

As shown in Fig. 2, Q_1 and Q_2 represent the request rate for LEO1 and LEO3, respectively. T is hop count from content provider (i.e., nodes that publish contents) to LEO3. Requests can be forwarded to content provider If caches miss in nodes. It is preferable for users to directly fetch contents from cache nodes rather than content provider which experience longer delay. The cost of satellite nodes cache deployment is node cache capacity, and profit is hop count for users to obtain contents. The smaller profit, the smaller hop count.

We assume that LEO1, LEO2 and LEO3 satellites deploy caches, and the cache capacity of each satellite is C, cache allocation results are shown in Table 1. The single node cache means that only one satellite is deployed cache, and users in other regions can obtain contents from this satellite node. The profit of cache deployment LEO2 = 1 is the best, namely hop count is minimal. Supposing that there are multi-satellite nodes deploying cache, and cache cooperation among satellite nodes is considered. Obviously, the hop count of cache deployment

Table 1. Cache allocation results.

Classification	Cache Deployment	Cost	Profit
Single node cache	LEO1 = 1	C	$2Q_2$
	LEO2 = 1	$\boldsymbol{C}$	$\boldsymbol{Q_1 + Q_2}$
	LEO3 = 1	C	$2Q_1$
Cooperative cache	LEO1 = 1, LEO2 = 1	$2C$	Q_2
	LEO1 = 1, LEO3 = 1	$\boldsymbol{2C}$	**0**
	LEO2 = 1, LEO3 = 1	$2C$	Q_1
Non-cooperative cache	**LEO1 = 1, LEO2 = 1, LEO3 = 1**	$\boldsymbol{3C}$	**0**

LEO1 = 1, LEO3 = 1 is the smallest. Compared with cooperative cache, the last non-cooperative cache leads to larger cache redundancy and waste cache space. In fact, the cache capacity is very limited and it is not practical to deploy large cache space in nodes. Thus, cooperative cache allocation gains better results than the other two cache allocation schemes from network revenue perspective.

3.2 Problem Formulation

Cooperative cache means that the probability of caching the same file among nodes obeys certain rules. By modeling cooperative cache, contents with higher popularity can be cached in nodes closer to users, and contents cached in the network is more diversified to improve transmission performance.

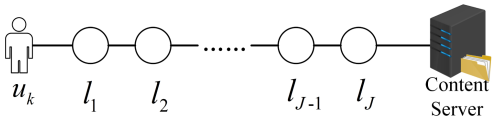


Fig. 3. Path topology for user to obtain contents.

When establishing the cooperative relationship model between nodes, cooperation parameter is proposed to reflect the probability that nodes cache the same file. The path topology for any user to obtain contents is shown in Fig. 3, supposing $L = \{l_j \,|1 \leq j \leq J\}$ indicates that J LEO satellites are connected between user u_k and content server. $F = \{f_m \,|1 \leq m \leq M\}$ means M files published by satellite nodes or content server. The size of file f_m is denoted by b_m in bytes. As described in Sect. 2, when a user sends request, the target file may be obtained at the access satellite l_1. Or the request will be forwarded to other satellites, namely $l_2, \cdots, l_j, \cdots, l_J$. Otherwise, the request will eventually reach content server. To describe the relationship between file and satellite node, Boolean variable $x_m^j \in \{0, 1\}, \forall m, \forall j$ is used to indicate the cache state of file f_m requested by the user at the satellite node l_j, which can be expressed as

$$x_m^j = \begin{cases} 1, \text{ if file } f_m \text{ is cached at node } l_j \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

According to the path for user to obtain contents, it can be known that all the contents must be stored in the content server. Thus, the probability of caching the same file between nodes should have a certain cooperative relationship. For the cache of the file f_m, when the node close to user caches f_m, the probability that the next-level node caches the same file can be expressed as

$$g_m^{j+1} = \frac{\beta}{1 + (-1)^{x_m^j} \cdot g_m^j} = \begin{cases} \frac{\beta}{1-g_m^j}, \ x_m^j = 1 \\ \frac{\beta}{1+g_m^j}, \ x_m^j = 0 \end{cases} \tag{2}$$

Where, g_m^j denotes the probability that the node close to user caches the file f_m, which can be regarded as the popularity of the file f_m. β is a constant between $[0, 1]$, which is seen as the cooperation degree between nodes.

The popularity of many current network traffic has been proved to follow the Zipf distribution. There are M files in the system, contents are sequenced in a descending order according to their popularity, the requested probability p_m for m_{th} popular content is given as

$$p_m = \frac{1/m^\alpha}{C} \tag{3}$$

Where the range of m is $[1, M]$, $C = \sum_{m \in [1,M]} 1/m^\alpha$. α is Zipf parameter, which is denoted by $Zipf\,(a)$. If the total request rate is Q, the request rate of m_{th} most popular content can be expressed as $Q_m = Q \cdot p_m$.

For notational convenience, all satellite nodes before hit node l_j are denoted as $L = \{l_i \,|0 \le i \le J\}$. In this case, hop count hop_m^j when user request to hit node can be denoted as

$$hop_m^j = \prod_{i<j} \left(1 - x_m^i\right) \cdot x_m^j \cdot j \tag{4}$$

Where x_m^i is the cache state of all nodes before the hit node l_j, and j is hop count from user to hit node l_j. Considering that in practice not only the hit node stores the target file, but also other nodes and server may cache the target file. Thus, the total hop count hop_m^J for user to obtain files from satellite nodes or server can be calculated as

$$hop_m^J = \sum_{j<J+1} \prod_{i<j} \left(1 - x_m^i\right) \cdot x_m^j \cdot j \tag{5}$$

In satellite networks, topology is regularly changing and predictable. Assuming that the maximum hop count for user to obtain contents is H, the probability that user will experience hop count is $\frac{1}{H}$. At this time, the hop count expectation hop_m for user to gain files is represented by

$$hop_m = \frac{1}{H} \sum_{J \in [1,H]} hop_m^J \tag{6}$$

The cooperative cache allocation scheme aims to minimize user average hop count *hop*, which is formulated as: for given cache capacity in satellites, content popularity and network topology, which contents should be cached in each satellite so that user average hop count is minimized. Thus, the optimization problem can be expressed as

$$\min hop = \sum_{m \in [1,M]} hop_m \cdot p_m \tag{7}$$

$$\text{s.t. C1: } g_m^{j+1} = \frac{\beta}{1 + (-1)^{x_m^j} \cdot g_m^j} \tag{8}$$

$$\text{C2: } \sum_{m \in [1,M]} x_m^j \cdot b_m \leq C_j, \forall j \tag{9}$$

$$\text{C3: } \sum_{u \in J} Q_u - \sum_{u \in J} Q_{uj} = 0, \text{ if } u = l_i \tag{10}$$

$$\text{C4: } \sum_{u \in J} Q_{ju} - \sum_{u \in J} x_m^u = 0, \text{ if } u = l_j \tag{11}$$

$$\text{C5: } \sum_{u \in J} Q_{ju} - \sum_{u \in J} Q_{uj} = 0, \text{ otherwise} \tag{12}$$

$$\text{C6: } x_m^j \in \{0, 1\}, \forall m, \forall j \tag{13}$$

The decision variable for the optimization objective is files to be cached on every satellite, that is content placement matrix X, and the values of the elements in the matrix X can be obtained by $x_m^j \in \{0, 1\}$. C_j is the cache capacity of each satellite in bytes. C1 ensures cooperative cache relationships between nodes. C2 sets the cache space limit for each satellite node. C3-C5 are flow conservation constraints. C6 limits the storage variables of the satellite nodes to 0 or 1.

3.3 Matching-Based Cache Allocation Algorithm

After modeling the system, it can be concluded that the optimization objective is a problem with high computational complexity because it contains 0–1 variables and it is difficult to find the problem optimal solution in polynomial time. Considering the time-varying characteristics of the satellite network topology and the limited on-board computing power, a low-complexity solution scheme should be explored. The matching theory [16] is a promising approach to perform resource management in satellite networks, which can effectively deal with the allocation problems such as on-board storage resources and cached contents. Thus, a matching-based cache allocation algorithm is proposed. For a node in satellite networks, a certain number of files need to be cached based on storage

space; also, files need to be cached on multiple satellite nodes based on user preferences. Therefore, the problem can be modeled as a many-to-many matching, in which satellites and files are players in matching games. Based on the above descriptions, we give the following definitions.

Definition 1: In the many-to-many matching model, a matching μ is function from the set $L \cup F$ into the set of all subsets of $L \cup F$, $\mu : L \cup F \rightarrow L \cup F$, for every $l_j \in L$ and $f_m \in F$ so that

$$\begin{aligned} &1)\ \mu(f_m) \subseteq L \text{ and } \mu(l_j) \subseteq F \\ &2)\ \sum\nolimits_{m \in [1,M]} |\mu(l_j)| \cdot b_m \leq C_j,\ \forall j \\ &3)\ l_j \in \mu(f_m) \text{ iff } f_m \in \mu(l_j) \end{aligned}$$

In 1), the matching node corresponding to the file belongs to the satellite, while the matching node corresponding to the satellite belongs to the file. 2) is the cache space limit for each satellite node. 3) indicates that file f_m is in the matching set of satellite l_j, if and only if that satellite l_j is also in the matching set of the file f_m.

Next, we need to build a preference list between the file and satellite. Each file has different presences for different satellite nodes, and each satellite node has different preferences for different files similarly. Here these two kinds of preference values are defined. First, based on the optimization objective, we define the preference value of a satellite over a file, which should consider the following principle: the satellite cached files make the user hop count smaller, the preference value of these files is higher. So, the preference value of satellites for different files is set as the ascending order of the utility function as

$$\varepsilon_j(m, \mu) = p_m \cdot hop_m^j + \sum_{m' \neq m} p_{m'} hop_{m'} \tag{14}$$

Similarly, we define the preference value of files over satellite nodes, which can be expressed as

$$\varepsilon_m(j, \mu) = p_m \cdot hop_m^j \tag{15}$$

It can be seen from the above defined preference value that the matching in this paper contains externalities, which the preference value of each satellite node for not only depends on the current satellite cache state, but also is affected by other satellite cache states. In many-to-many matching with externalities, a stability concept cannot be defined straightforwardly because the gain from a matching pair depends on the matching results of other players. Inspired by the definition of exchange stability, swap matching is defined to achieve stability [17].

Definition 2: In the matching μ, a swap matching is defined as

$$\begin{aligned} \mu_{mj}^{m'j'} &= \{\mu \backslash \{(m, \mu(m)), (m', \mu(m'))\}\} \cup \\ &\{(m, \{\{\mu(m) \setminus \{j\}\} \cup \{j'\}\}), (m', \{\{\mu(m') \setminus \{j'\}\} \cup \{j\}\})\} \end{aligned} \tag{16}$$

where $j \in \mu(m), j' \in \mu(m'), j \notin \mu(m')$ and $j' \notin \mu(m)$. In swap matching, two agents in the same set exchange their matches in the opposite set while keeping all other agents' matching state unchanged. It is worth noticing that one of satellite $l_{j'}$ can be a hole which allowing for satellite moving to available vacancies. Based on the operations of swap matching, we will the concept of swap-blocking pairs as follows.

Definition 3: $(l_j, l_{j'})$ is a swap-blocking pairs in the matching μ, such that

$$1)\ \forall i \in \{m, m',\ j, j'\}, \varepsilon_i\left(\mu_{mj}^{m'j'}\right) \leq \varepsilon_i(\mu)$$
$$2)\ \exists i \in \{m, m',\ j, j'\}, \varepsilon_i\left(\mu_{mj}^{m'j'}\right) < \varepsilon_i(\mu)$$

In Definition 3, condition 1) implies that the utilities of all involved satellites and files should not be reduced after the exchange operation between the swap-blocking pairs $(l_j, l_{j'})$. Condition 2) indicates that at least one of the players' utilities is increased after the exchange operation between the swap-blocking pair. The matching μ is two-sided exchange-stable if and only if there are no swap-blocking pairs.

Algorithm 1: Proposed Matching-Based Cache Allocation Strategy.

Initialization
Satellites and files are randomly matched with each other subject to constraints (8)-(13), forming a matching state μ.
Set iteration count $iter = 0$.
Exchange Matching Process
For each satellite l_j, it searches for another satellite $l_{j'}$ or file's available vacancies O to form a swap-blocking pair.
If $(l_j, l_{j'})$ or (l_j, O) forms a swap-blocking pair along with $j \in \mu(m)$, and $j' \in \mu(m')$,
Update the current matching state to $\mu_{mj}^{m'j'}$ and reset $iter = 0$.
Else if there does not exist such a swap-blocking pari,
Keep matching state μ unchanged and update $iter = iter + 1$.
Repeat Step 5 to 9 until $iter > iter_{\max}$.
End Algorithm

The proposed algorithm is shown in Algorithm 1, which includes two stages: Initialization and exchange matching stage. In the initialization stage, complete the random initial matching scheme in the scene, and set iteration count to 0. In the exchange matching stage, it cyclically searches for swap-blocking pairs and performs exchange operations until there are no swap-blocking pairs for certain consecutive times. At this time, the matching formed after multiple exchange operations is a stable matching. In the algorithm, satellites and files try to find swap-blocking pairs according to the preference value, then the number of exchange operations is $O(J \cdot M)$. During each exchange operation, the

complexity caused by the Quick Sort ordering operations is $O(JM\log_2(JM))$. Thus, the complexity of entire caching allocation algorithm can be calculated as $O\left(J^2M^2\log_2(JM)\right)$.

3.4 Proactive Pushing Scheme

As mentioned above, LEO satellites will make cache decision in the satellite's coverage area based on the matching-based cache allocation algorithm proposed in Sect. 3.3. When file arrives at satellite node, it is directly cached if the node cache space is left; otherwise, the name of the arriving file is matched with the file in the cache decision. If the match is successful, the cached file is replaced according to the cache decision; if the match is unsuccessful, no operation is performed on the arriving file.

However, the link between user and access satellite be interrupted due to the mobility of LEO satellites, the subsequent satellite covering the area will provide service for user. In this case, interest packet cannot be responded in time. For the user, the requested file is cached in the previous satellite, and the files cached by the subsequent satellite are not preferred by the user, which will increase hop count for users to obtain contents. Furthermore, there are tremendous differences in user preferences and the number of content requests in different geographic regions, simply adopting local caching policies will lead to frequent cache updates. Therefore, in the case of in-network caching, we propose a proactive pushing scheme to improve network transmission efficiency.

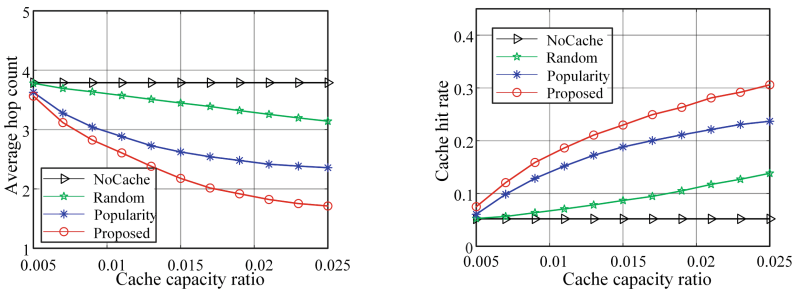
The basic idea of the proactive pushing scheme here is that when an LEO satellite is about to leave the divided area, GEO satellite will notify the LEO to push the cached list to the subsequent satellite covering the area in advance. With the equal division of the area, each LEO satellite can start the proactive cache pushing process one by one under the coordination of GEO satellite. The proactive pushing algorithm can be described as follows.

Step 1: LEO satellites will make cache decision based on the matching-based cache allocation algorithm proposed in Sect. 3.3, while GEO satellites monitor the movement of LEO satellites.
Step 2: According to the movement speed and trajectory of LEO satellite l_j, GEO judges whether satellite l_j is about to leave the coverage area. If so, continue to Step 3. Otherwise, go to Step 1.
Step 3: GEO informs satellite l_j to generate a cache list CL_j with only the file name based on the cached files, and push it to the subsequent satellite l_{j+1}.
Step 4: The subsequent satellite l_{j+1} determines whether CL_j is consistent with CL_{j+1}. If not, the corresponding interest packets will be generated, and sent to satellite l_j to request cache files that are not in CL_{j+1}. Until satellite l_{j+1} no longer sends interest packets, the step ends.

4 Simulation Results

To demonstrate the effectiveness of the proposed cache allocation scheme, MATLAB is used in our simulations. In the simulation environment, the STI^{2}CN scenario should ensure normal satellite communication and the coverage of communication on the whole earth. Therefore, Iridium system is adopted, which consists of 6 orbital planes with 11 satellites residing in each orbit, 86.4° inclined orbit planes with the altitude of 780 km. Each satellite is linked to the forward and rear satellites of adjacent planes. Total number of 1,000 files is supposed Zipf distribution, where $\alpha \in [0.2, 1.2]$ varies from different file types. The sizes of files are same, $b_m = 1, \forall m$. The cache capacity of each LEO satellite is same, which varies from 0.005 to 0.025. The maximum hop count $H = 15$ and simulation parameters $iter_{\max} = 20$ is used.

In addition, three caching schemes are selected as comparison schemes in the simulation process: the first one is NoCache that no caching in satellites and cache capacity is 0, the second is Random that each satellite caches files randomly, the last one is content popularity cache that most popular files are cached in each satellite. During the simulation, the impact on the network performance is observed by changing the size of nodes' cache capacity and the Zipf parameters α. We choose two metrics to evaluate the performance of caching allocation scheme: average hop count and cache hit rate. Average hop count refers to the average hop count that interest packet caches hit or reaches to content source to obtain files. Cache hit rate refers to the percentage of cache hits in the satellite nodes when all the requests arrive at the satellite nodes.

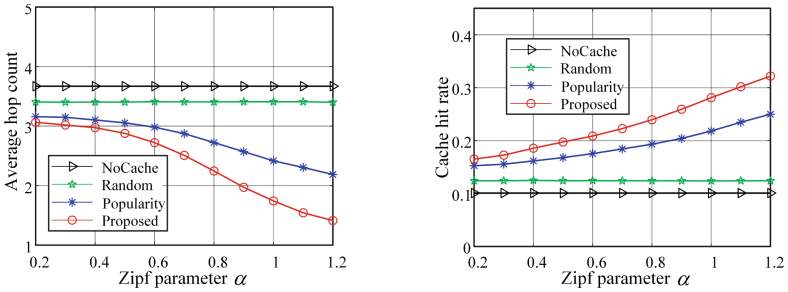


(a) Average hop count (b) Cache hit rate

Fig. 4. The impact of cache capacity ratio.

Figure 4 shows the average hop count and cache hit rate versus different cache capacity ratio, the following simulation results set the Zipf parameter $\alpha = 0.8$. When the cache capacity is small, the increase in cache capacity enables more requests to be responded in neighboring nodes, and the cache distribution of popular files tends to stabilize as the cache capacity increases further. In Fig. 4(a), with the increasing of cache capacity ratio, average hop count among the three schemes of Random, Popularity and Proposed is diminishing. For popularity cache, each satellite tends to cache most popular files based on the preference

of its served users, making it perform better than Random. However, popularity cache considers no cooperation resulting in the cache redundancy between adjacent satellites. As comparison, our scheme further decreases user average hop count by considering cooperation among satellites. Another result in Fig. 4(b) shows that cache hit rate increases as cache capacity ratio increases and eventually plateaus, but our scheme's cache hit rate is always higher than the other two schemes.



(a) Average hop count (b) Cache hit rate

Fig. 5. The impact of Zipf parameter α.

Figure 5 shows the average hop count and cache hit rate versus different α, the following simulation results set cache capacity ratio 0.015. The Zipf parameter α reflects the concentration of file requests. When α is small, users' requests are relatively scattered, and the limited cache space cannot satisfy more requests. When α is large, users' requests are more concentrated on popular contents. Random algorithm is indifferent, while popularity cache and our cache scheme perform better with the increase of α in Fig. 5(a). That is because both schemes make content decisions based on the file popularity. Meanwhile, the cooperative cache allocation in this paper gains most under different popularity models. Similarly, it can be seen from Fig. 5(b) that cache hit rate in Random is basically unchanged, while cache hit rate in other two schemes increases as α increases.

5 Conclusions

In this article, we investigate content cache allocation scheme in information-centric satellite-terrestrial integrated networks. Firstly, the cache allocation problem is formulated as user average hop count minimization by considering content popularity and cache cooperation. Then, a matching algorithm with fast convergence and low complexity is used to solve it, and finally obtains exchange stability between satellites and contents. Additionally, a proactive pushing scheme is proposed to accelerate user content retrieval and further improve network performance. The simulation results show that proposed scheme can effectively reduce user average hop count and increase cache hit rate.

References

1. Liu, J., Shi, Y., Fadlullah, Z.M., Kato, N.: Space-air-ground integrated network: a survey. IEEE Commun. Surv. Tutor. **20**(4), 2714–2741 (2018)
2. Xie, R., Tang, Q.: Satellite-terrestrial integrated edge computing networks: architecture, challenges, and open issues. IEEE Netw. **34**(3), 224–231 (2020)
3. Li, J., Xue, K., Liu, J., Zhang, Y.: An ICN/SDN-based network architecture and efficient content retrieval for future satellite-terrestrial integrated networks. IEEE Netw. **34**(1), 188–195 (2019)
4. Bi, Y., et al.: Software defined space-terrestrial integrated networks: architecture, challenges, and solutions. IEEE Netw. **33**(1), 22–28 (2019)
5. Shi, Y., Cao, Y., Liu, J., Kato, N.: A cross-domain SDN architecture for multi-layered space-terrestrial integrated networks. IEEE Netw. **33**(1), 29–35 (2019)
6. Ji, S., Sheng, M., Zhou, D., Bai, W.: Flexible and distributed mobility management for integrated terrestrial-satellite networks: challenges, architectures, and approaches. IEEE Netw. **35**(4), 73–81 (2021)
7. Din, I.U., Hassan, S., Khan, M.K., Guizani, M.: Caching in information-centric networking: strategies, challenges, and future research directions. IEEE Commun. Surv. Tutor. **20**(2), 1443–1474 (2017)
8. Ngaffo, A.N., El Ayeb, W.: Information-centric networking challenges and opportunities in service discovery: a survey. In: 2020 IEEE Eighth International Conference on Communications and Networking (ComNet), pp. 1–8. IEEE (2020)
9. Liu, Z., Zhu, J., Pan, C., Song, G.: Satellite network architecture design based on SDN and ICN technology. In: 2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC), pp. 124–131. IEEE (2018)
10. Siris, V.A., Ververidis, C.N., Polyzos, G.C.: Information-centric networking (ICN) architectures for integration of satellites into the future internet. In: 2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL), pp. 1–6. IEEE (2012)
11. de Cola, T., Gonzalez, G., et al.: Applicability of ICN-based network architectures to satellite-assisted emergency communications. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2016)
12. D'Oro, S., Galluccio, L., Morabito, G., Palazzo, S.: SatCache: a profile-aware caching strategy for information-centric satellite networks. Trans. Emerg. Telecommun. Technol. **25**(4), 436–444 (2014)
13. Wu, H., Li, J., Lu, H., Hong, P.: A two-layer caching model for content delivery services in satellite-terrestrial networks. In: 2016 IEEE global communications conference (GLOBECOM), pp. 1–6. IEEE (2016)
14. Feng, Y., Wang, W., Liu, S., Cui, G., Zhang, Y.: Research on cooperative caching strategy in 5G-satellite backhaul network. In: Yu, Q. (ed.) SINC 2017. CCIS, vol. 803, pp. 236–248. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7877-4_21
15. Liu, S., Hu, X., Wang, Y., Cui, G., Wang, W.: Distributed caching based on matching game in LEO satellite constellation networks. IEEE Commun. Lett. **22**(2), 300–303 (2017)
16. Gu, Y., Saad, W., Bennis, M., Debbah, M.: Matching theory for future wireless networks: fundamentals and applications. IEEE Commun. Mag. **53**(5), 52–59 (2015)
17. Zhao, J., Liu, Y., Chai, K.K., Chen, Y.: Many-to-many matching with externalities for device-to-device communications. IEEE Wirel. Commun. Lett. **6**(1), 138–141 (2016)