On the Distributed Routing and Data Scheduling in Interplanetary Networks

Xiaojian Tian[1] and Zuqing Zhu[1,†]
[1]School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[†]Email: {zqzhu}@ieee.org

***Abstract*—With the advances on human's exploration of the universe, interplanetary networks (IPNs) are attracting more and more research interests. However, the unique characteristics of IPNs make many of the networking technologies on Earth not applicable. In this work, we design a routing and data scheduling algorithm that can make interplanetary data transfers (IP-DTs) more scalable and robust. Specifically, we propose an online approach to schedule and route IP-DTs in the distributed way, by leveraging the Lyapunov optimization. With extensive simulations, we show that our proposed algorithm can optimize the performance of IP-DTs with only the information about local queues on each node in an IPN. The simulation results also verify that our algorithm outperforms the existing ones significantly in terms of the average E2E latency of IP-DTs, and properly adjusts the tradeoff between average E2E latency and delivery ratio.**

***Index Terms*—Interplanetary network (IPN), Delay tolerant network (DTN), Queue scheduling, Lyapunov optimization.**

I. Introduction

Nowadays, we have already witnessed the rapid development of Internet infrastructure on our planet [1–3]. Meanwhile, the increasing activities on deep space (DS) exploration have promoted the research and development (R&D) on interplanetary networks (IPNs). An IPN is responsible for the communications among DS objects [4], and thus it is a critical infrastructure to facilitate DS missions and operates significant differently from the networks on Earth [5–7]. Due to the unique applications, current IPNs usually have relatively simple architectures with sparse topology density. However, the ongoing and future plans of DS missions are bringing more DS objects and services in IPNs, which will not only increase their scales and density but also differentiate the traffic in them [8]. Hence, it is relevant and necessary to accelerate the R&D on networking technologies for IPNs, especially the routing and data scheduling algorithms that will make interplanetary data transfers (IP-DTs) more scalable and robust.

Fig. 1 shows an example of IPN that consists of a ground station and a geostationary satellite, which are on or around Earth, as well as the relay satellites and rovers for other celestial bodies (*i.e.*, Moon and Mars). Here, the relay satellites and rovers are just the DS objects in IPN. The difficulty of realizing scalable and robust IP-DTs mainly comes from two aspects. Firstly, the movement of DS objects and shields of celestial bodies make the links in IPNs time-varying and unstable, and thus we often cannot find available end-to-end routing paths to support IP-DTs. Secondly, the links in IPNs are normally several magnitudes longer than the communication links on Earth, and they are always exposed to complex electro-magnetic interferences in the universe. Therefore, the data transmission on each link will be extremely unreliable.

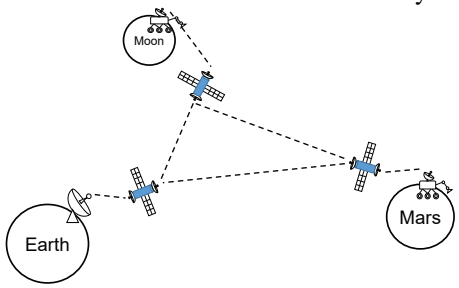


Fig. 1. An example of IPN.

To address these challenges, people have proposed delay tolerant networking (DTN) [9], which adopts the way of store-carry-forward (SCF) to realize IP-DTs. Specifically, DTN defines a series of contiguous data blocks as a "bundle", and each bundle contains sufficient information, with which an application on the destination can make progress [10]. People have standardized several protocols for DTN (*e.g.*, in [10]), and the National Aeronautics and Space Administration (NASA) has conducted in-depth R&D and space tests on them.

As DS objects and celestial bodies usually have fixed orbits and operation periods, the contact plan of each link in an IPN is actually determined, *i.e.*, the topology of each IPN is time-varying but predictable. Hence, the contact graph routing (CGR) [11] was proposed to compute routes for IP-DTs with the predictable topology, and it assumes that pending bundles are sent immediately upon each contact of a link, *i.e.*, the queuing delay of bundles is ignored. However, in a real-world IPN (especially a busy one in the future), many bundles can be buffered at a node, and thus their queuing delay can become long enough to let them miss the planned contacts. To address this issue, the authors of [12] designed the algorithm of CGR-ETO to enhance CGR such that the routing calculation will take the estimated queuing delay of each bundle into account.

Nevertheless, both CGR and CGR-ETO assume that bundles are handled in queues in the first-in-first-out (FIFO) manner, *i.e.*, they do not optimize the scheduling in queues to further improve the performance of IP-DTs. Later on, in [13], Ramamurthy *et al.* proposed to jointly optimize the routing and data scheduling of IP-DTs based on the multi-attribute decision making principle. Although their proposed algorithm (namely, MARS) performs better than those that only address the routing, it only schedules the bundles in a single queue

and does not optimize the scheduling of multiple queues on a same node jointly. Note that, multiple queues can be allocated in a node of IPN, to correspond to different next hops or/and various network services. Moreover, MARS ranks the bundles in a queue to transmit greedily, which is still relatively simple and might not guarantee the best scheduling performance.

In this work, we propose an online approach to schedule and route IP-DTs in the distributed way. The rationale behind designing the distributed algorithm is that the characteristics of IPN make it difficult for nodes to collect information about others, and even if the information can be obtained, it will be out-of-date due to the long latency between nodes. We leverage the Lyapunov optimization [14] to design the distributed online approach, and prove that it can optimize the performance of IP-DTs with only the information about local queues on each node in an IPN. Simulation results confirm that our proposed algorithm can outperform the existing ones in the literature.

The rest of the paper is organized as follows. Section II formulates the problem of routing and data scheduling in an IPN. The algorithm design is presented in Section III, and we discuss the simulations for performance evaluation in Section IV. Finally, Section V summarizes the paper.

II. Problem Formulation

A. Network Model

We model the topology of an IPN as a temporal graph $G_t(V, E_t)$, where V is the set of IPN nodes and E_t denotes the set of temporal links at time t. The nodes can be ground stations or control centers on Earth, satellites, and DS objects (*e.g.*, probes and rovers). We use $e_t = (u, v, t_s, t_e, r, \tau) \in E_t$ to represent a temporal link from u to v ($u, v \in V$), where t_s and t_e denote the start time and stop time of the link's contact, respectively, and r and τ are the data rate and transmission latency of the link, respectively. As the contact plan of each link in the IPN is predictable, the parameters of $\{t_s, t_e, r, \tau\}$ are actually known for e_t when t is provided. Meanwhile, for simplicity, we assume that the values of r and τ do not change during each contact of e_t. This is similar to the assumption used in previous studies [11, 13]. Note that, some of the links can exist consistently in an IPN, *e.g.*, those between ground stations and geostationary satellites. Hence, the values of t_s and t_e of such a permanent link is set as 0 and ∞, respectively.

We define the IP-DT request of a bundle as $B(s, d, \beta, t_a, t_d)$, where s and d are its source and destination ($s, d \in V$), respectively, β represent the data size of the bundle, t_a is the time that the request arrives at s, and t_d is the deadline by when the bundle should be delivered to d. If the bundle cannot reach d by t_d, it will be discarded by the IPN.

B. Distributed Routing and Data Scheduling in IPN

Algorithm 1 shows the operation principle of our proposed approach for distributed routing and data scheduling on each node of an IPN. Specifically, the idea is that we create a few queues on each node $v \in V$ of the IPN $G_t(V, E_t)$ to buffer bundles, and at each time instant t, we leverage the Lyapunov optimization to select bundles from the queues to transmit according to the current network status. Hence, each node works as a discrete-time system that operates on time-slots (TS'), each of which has a fixed duration of Δt[1]. This means that the IP-DT scheme on each node can be changed at the beginning of each TS (*i.e.*, $t = \Delta t, 2\Delta t, \cdots$). For simplicity, we normalize the system time as $t \in \{1, 2, \cdots\}$.

Algorithm 1: Distributed Routing and Data Scheduling

```
1  t = 0, initialize Q_v and {Q_{v,u}};
2  while node v ∈ V is operational do
3      insert all the newly generated/received bundles in Q_v;
4      for each bundle B in Q_v do
5          if the routing path of B is undetermined then
6              calculate a routing path p for B with the CGR
               algorithm, and record p in B.path;
7          end
8          if B.path ≠ ∅ then
9              get the next hop u for B from B.path;
10             insert B in Q_{v,u} and remove B from Q_v;
11         end
12     end
13     for each queue Q_{v,u} do
14         if e_t = (v, u) is in contact then
15             use Algorithm 2 to select bundles in Q_{v,u} to
               transmit and remove them from Q_{v,u};
16         else
17             for each bundle B in Q_{v,u} do
18                 mark routing path of B as undetermined;
19                 insert B in Q_v and remove it from Q_{v,u};
20             end
21         end
22     end
23     t = t + 1, remove expired bundles in Q_v and {Q_{v,u}};
24 end
```

The queues on each node $v \in V$ belong to two categories, *i.e.*, the buffering and outgoing queues. First of all, we allocate a **buffering queue** $\mathcal{Q}_v$ to store all the bundles, which can be either locally generated or received from adjacent nodes for being relayed. Secondly, for each node $\{u : u \neq v,\ u \in V\}$ that can have direct contact with node v, we create an **outgoing queue** $\mathcal{Q}_{v,u}$ to store all the bundles that will use node u as their next hop. In *Algorithm* 1, *Line* 1 initializes the queues on node v, while *Line* 3 explains the enqueue operation of $\mathcal{Q}_v$.

The for-loop that covers *Lines* 4-12 processes the bundles in $\mathcal{Q}_v$ to insert them in $\{\mathcal{Q}_{v,u}\}$ accordingly. Specifically, if the routing path of a bundle B is undetermined, we calculate a routing path p by applying the CGR algorithm [11] on the temporal graph $G_t(V, E_t)$ of the IPN, and store the path as an attribute of B (*i.e.*, $B.path = p$) (*Lines* 5-7). Note that, promoted by NASA, CGR is the most prominent routing algorithm used in IPNs [13]. Based on the temporal graph $G_t(V, E_t)$, CGR first represents the contact plan of each link $e_t = (u, v, t_s, t_e, r, \tau) \in E_t$ as a time-ordered list, each element in which denotes the information of a "contact" (*i.e.*, $\{t_s, t_e, r, \tau\}$). Then, for each bundle $B(s, d, \beta, t_a, t_d)$, CGR considers the contact plans to find the routing path from s to d,

[1]Here, the value of Δt should be selected to guarantee that the largest possible bundle can be transmitted within Δt on the link whose data rate is the lowest in the IPN. This is because the bundle protocol in [10] specifies that for the IP-DTs in an IPN, each bundle is atomic.

which does not have to be continuous in time but can achieve the shortest end-to-end latency. There is a "forfeit time" for each node on the path, which tells the deadline by when B has to be sent to the next hop to be able to reach d by t_d.

For bundle B, if its routing path can be found by CGR, we get the next hop u, and move it from $\mathcal{Q}_v$ to $\mathcal{Q}_{v,u}$ (*Lines* 8-11). As CGR assumes that each bundle will be transmitted immediately when its contact starts, the algorithm ignores the queuing delay after which the bundle's data transfer can actually take place. This, however, can make the bundle miss its forfeit time on certain node of its routing path, especially when the traffic load in the IPN is relatively high. This issue can be addressed with the data scheduling in *Lines* 13-22, where the bundles in each outgoing queue $\mathcal{Q}_{v,u}$ are scheduled.

Specifically, for an outgoing queue $\mathcal{Q}_{v,u}$, we first check whether the corresponding outgoing link is in contact. If yes, we use *Algorithm* 2 to select bundles in $\mathcal{Q}_{v,u}$ to transmit (*Lines* 14-15). Otherwise, we mark the routing paths of the bundles in $\mathcal{Q}_{v,u}$ as undetermined and move them from $\mathcal{Q}_{v,u}$ back to $\mathcal{Q}_v$, to invoke rerouting for them in the next iteration (*Lines* 16-20). Here, *Algorithm* 2 is our proposed distributed data scheduling algorithm, which will be elaborated in the next section. Finally, the system time proceeds for a TS, and we remove all the bundles whose deadlines have already been passed from the queues on node v (*Line* 23).

III. Algorithm Design of Data Scheduling

Comparing with other networks that also require distributed routing and data scheduling, IPN normally has a more sophisticated contact plan. In this section, we explain our algorithm design to realize distributed data scheduling on each node in an IPN. Specifically, on each node $v \in V$, the data scheduling leverages Lyapunov optimization to select bundles in the outgoing queues $\{\mathcal{Q}_{v,u}\}$ to transmit, based on the node's local information. To facilitate the Lyapunov optimization, we preprocess each outgoing queue $\mathcal{Q}_{v,u}$ to categorize the bundles in it based on their previous hop before node v (*i.e.*, node $\hat{v}$) and put them into several virtual queues accordingly.

Each virtual queue can be denoted as $\mathcal{Q}_{v,u}^{\hat{v}}$, which stores the bundles that are in $\mathcal{Q}_{v,u}$ and were sent to node v from node $\hat{v}$. Here, we can have $\hat{v} = v$, which means that the corresponding bundles are locally generated at node v, and we should also ensure $\hat{v} \neq u$ to avoid routing loops. If we define the set of nodes, which can have direct contact with node v, as V_v and assume $|V_v| = K$, the aforementioned preprocessing will distributes the bundles in each outgoing queue $\mathcal{Q}_{v,u}$ to $|(V_v \setminus u) \cup \{v\}| = K$ virtual queues. Hence, we can assign a unique index k to each $\hat{v}$, where we have $k \in [1, K]$, and simplify $\mathcal{Q}_{v,u}^{\hat{v}}$ as $\mathcal{Q}_k$. Then, the data scheduling needs to schedule the data transfers of bundles in $\mathcal{Q}_{v,u}$, according to the status of all of its virtual queues (*i.e.*, $\{\mathcal{Q}_k, \forall k\}$).

A. Lyapunov Optimization Model

Lyapunov optimization [14] is a powerful scheduling technique that can achieve time-average optimization in stochastic systems. To represent the time-varying state of virtual queues $\{\mathcal{Q}_k, \forall k\}$, we define the vector of current queue backlogs as $\mathbf{Q}(t) = [\mathcal{Q}_1(t), \cdots, \mathcal{Q}_K(t)]$, where $\mathcal{Q}_k(t)$ denotes the state of virtual queue $\mathcal{Q}_k$ at TS t. The queue evolution over time is

$$\mathcal{Q}_k(t+1) = \max\left[\mathcal{Q}_k(t) - b_k(t), 0\right] + A_k(t), \quad \forall k, t, \tag{1}$$

where $A_k(t)$ denotes the number of bundles that are put in $\mathcal{Q}_k$ at TS t, and $b_k(t)$ is the number of bundles that are selected for being sent on link $e_t = (v, u)$ at TS t.

Then, we can define a Lyapunov function $L(\mathbf{Q}(t))$ as

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2}\sum_{k=1}^{K}\left[\mathcal{Q}_k(t)\right]^2, \tag{2}$$

which represents a scalar measure of queue congestion, and the conditional Lyapunov drift of TS t can be obtained as

$$\Delta(\mathbf{Q}(t)) \triangleq \mathbb{E}\left\{L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) \mid \mathbf{Q}(t)\right\}, \tag{3}$$

which denotes the expectation of the change in $L(\mathbf{Q}(t))$ due to channel states, queue states and control actions. With Eq. (3), we can push $L(\mathbf{Q}(t))$ to low congestion consistently as,

$$\begin{aligned} L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) &= \frac{1}{2}\sum_{k=1}^{K}\left[\mathcal{Q}_k(t+1)^2 - \mathcal{Q}_k(t)^2\right] \\ &= \frac{1}{2}\sum_{k=1}^{K}\left\{\left\{\max\left[\mathcal{Q}_k(t) - b_k(t), 0\right] + A_k(t)\right\}^2 - \mathcal{Q}_k(t)^2\right\} \\ &\leq \sum_{k=1}^{K}\frac{\left[A_k(t)^2 + b_k(t)^2\right]}{2} + \sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\left[A_k(t) - b_k(t)\right], \end{aligned} \tag{4}$$

because for any $Q \geq 0$, $b \geq 0$, $A \geq 0$, we have the inequality

$$\left\{\max\left[Q - b, 0\right] + A\right\}^2 \leq Q^2 + A^2 + b^2 + 2Q\cdot(A - b). \tag{5}$$

Therefore, the Lyapunov drift $\Delta(\mathbf{Q}(t))$ in Eq. (3) satisfies

$$\begin{aligned} \Delta(\mathbf{Q}(t)) \leq\ & \mathbb{E}\left\{\sum_{k=1}^{K}\frac{\left[A_k(t)^2 + b_k(t)^2\right]}{2} \mid \mathbf{Q}(t)\right\} \\ &+ \sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\lambda_k - \mathbb{E}\left\{\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot b_k(t) \mid \mathbf{Q}(t)\right\}, \end{aligned} \tag{6}$$

where λ_k denotes the time-average value of $\mathbb{E}\left\{A_k(t) \mid \mathbf{Q}(t)\right\}$, which can be got as $\lambda_k = \mathbb{E}\left\{A_k(t) \mid \mathbf{Q}(t)\right\} = \mathbb{E}\left\{A_k(t)\right\}$. This is because $A_k(t)$ denotes the arrivals of new bundles to $\mathcal{Q}_k$, which is independent of the current queue state $\mathbf{Q}(t)$.

As $A_k(t)$ and $b_k(t)$ are finite, we can verify that the first term on the right side of Eq. (6) has a finite upper-bound M

$$\mathbb{E}\left\{\sum_{k=1}^{K}\frac{\left[A_k(t)^2 + b_k(t)^2\right]}{2} \mid \mathbf{Q}(t)\right\} \leq M. \tag{7}$$

Meanwhile, to express the impact of data transfer decisions on the stochastic system, we rewrite $b_k(t)$ as

$$b_k(t) = \hat{b}_k(\alpha(t), \mathbf{S}(t)), \tag{8}$$

where $\alpha(t)$ denotes the data transfer decision for TS t (*i.e.*, from which queue among $\{\mathcal{Q}_k(t)\}$ we choose bundles to transmit at TS t), and $\mathbf{S}(t)$ is the current state vector of temporal link $e_t = (v, u)$ (*i.e.*, $\mathbf{S}(t) = [t_s, t_e, r, \tau]$).

By combining Eqs. (6)-(8), we can obtain

$$\begin{aligned} \Delta(\mathbf{Q}(t)) \leq & M + \sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\lambda_k \\ & - \mathbb{E}\left\{\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\hat{b}_k(\alpha(t), \mathbf{S}(t)) \mid \mathbf{Q}(t)\right\}. \end{aligned} \tag{9}$$

As the right side of Eq. (9) actually denotes the upper-bound of the Lyapunov drift $\Delta(\mathcal{Q}(t))$, we need to minimize it by making the right data transfer decision $\alpha(t)$. Note that, $\alpha(t)$ only affects the last term in the upper-bound, and thus we actually need to maximize the following expression

$$\mathbb{E}\left\{\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\hat{b}_k(\alpha(t),\mathbf{S}(t))\mid\mathbf{Q}(t)\right\}. \tag{10}$$

The conditional expectation in Eq. (10) can be maximized opportunistically by observing $\mathbf{Q}(t)$ and $\mathbf{S}(t)$ and selecting the right data transfer decision $\alpha(t)$ to maximize [14]

$$\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\hat{b}_k(\alpha(t),\mathbf{S}(t)). \tag{11}$$

Assuming that $\alpha^*(t)$ is the optimal data transfer decision at TS t, which can maximize the term in Eq. (11), we can define $b_k^*(t)=\hat{b}_k(\alpha^*(t),\mathbf{S}(t))$. Then, Eq. (9) gets transformed into

$$\begin{aligned}\Delta(\mathbf{Q}(t)) &\le M+\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\lambda_k-\mathbb{E}\left\{\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot b_k^*(t)\mid\mathbf{Q}(t)\right\}\\ &= M-\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\left[\mathbb{E}\left\{b_k^*(t)\mid\mathbf{Q}(t)\right\}-\lambda_k\right].\end{aligned} \tag{12}$$

As $\alpha^*(t)$ is a particular data transfer decision, it is independent of $\mathbf{Q}(t)$. Hence, to ensure the stability of queues, there exists an $\epsilon\ge 0$ that can make $\left[\mathbb{E}\left\{b_k^*(t)\mid\mathbf{Q}(t)\right\}-\lambda_k\right]$ non-negative,

$$\mathbb{E}\left[b_k^*(t)\mid\mathbf{Q}(t)\right]-\lambda_k=\mathbb{E}\left[b_k^*(t)\right]-\lambda_k\ge\epsilon,\ \forall k. \tag{13}$$

Hence, Eq. (12) can be transformed into

$$\Delta(\mathbf{Q}(t))\le M-\sum_{k=1}^{K}\mathcal{Q}_k(t)\cdot\epsilon. \tag{14}$$

By taking the expectation of both sides, we can get

$$\begin{aligned}\mathbb{E}\left[\Delta(\mathbf{Q}(t))\right]&=\mathbb{E}\left\{\mathbb{E}\left[L(\mathbf{Q}(t+1)-L(\mathbf{Q}(t))\mid\mathbf{Q}(t)\right]\right\}\\ &=\mathbb{E}\left[L(\mathbf{Q}(t+1)\right]-\mathbb{E}\left[L(\mathbf{Q}(t)\right]\\ &\le M-\epsilon\cdot\sum_{k=1}^{K}\mathbb{E}\left[\mathcal{Q}_k(t)\right].\end{aligned} \tag{15}$$

Then, by defining the system's operation time as T (*i.e.*, $t\in[0,T-1]$), we can get the telescoping sum of Eq. (15) as

$$\mathbb{E}\left\{L(\mathbf{Q}(T)\right\}-\mathbb{E}\left\{L(\mathbf{Q}(0)\right\}\le M\cdot T-\epsilon\cdot\sum_{t=0}^{T-1}\sum_{k=1}^{K}\mathbb{E}\left\{\mathcal{Q}_k(t)\right\}. \tag{16}$$

Eq. (16) can be further reduced as follows, after dividing both sides by $\epsilon\cdot T$ and utilizing the fact that $L(\mathbf{Q}(T))\ge 0$

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K}\mathbb{E}\left[\mathcal{Q}_k(t)\right]\le\frac{M}{\epsilon}+\frac{\mathbb{E}\left[L(\mathbf{Q}(0)\right]}{\epsilon\cdot T}. \tag{17}$$

Apparently, we have $\mathbb{E}\left\{L(\mathbf{Q}(0)\right\}\ll\infty$. Therefore, when T approaches ∞, we can get

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{k=1}^{K}\mathbb{E}\left[\mathcal{Q}_k(t)\right]\le\frac{M}{\epsilon}, \tag{18}$$

which verifies that all the queues are strongly stable [14] while the time-average length of the queues is bounded.

B. Distributed Data Scheduling Algorithm

In order to improve the delivery ratio of bundles and reduce their average data transfer latency, we need to minimize their average queuing delay on the nodes in an IPN. Meanwhile, according to the Little's Theorem [15], the average queuing delay is proportional to the time-average length of queues. Hence, based on the derivations in the previous subsection, we can obtain an effective algorithm to schedule the data transfers on each node distributedly, such that the time-average length of queues can be minimized. *Algorithm* 2 shows the details of the distributed data scheduling, which schedules the data transfers of bundles in an outgoing queue $\mathcal{Q}_{v,u}$ on node v.

Algorithm 2: Distributed Data Scheduling in IPN

```
   Input: {Q_k(t), ∀k}, e_t = (v,u), S(t) = [t_s, t_e, r, τ]
1  max = 0, α*(t) = 0, Ĉ = r · Δt;
2  while Ĉ > 0 do
3      for k = 1 to K do
4          α(t) = k, b_k(t) = 0, C = Ĉ;
5          sort bundles in Q_k(t) in descending order of their
           deadlines;
6          while Q_k(t) > 0 do
7              pop a bundle B(s, d, β, t_a, t_d) from Q_k(t);
8              if C − β ≥ 0 then
9                  b_k(t) = b_k(t) + 1, C = C − β;
10             else
11                 break;
12             end
13         end
14         x = Q_k(t) · b_k(t);
15         if x > max then
16             max = x;
17             α*(t) = α(t), b*_k(t) = b_k(t), C* = C;
18         end
19     end
20     send bundles according to α*(t) and b*_k(t);
21     if b*_k(t) > 0 then
22         Ĉ = C*;
23     else
24         Ĉ = 0;
25     end
26 end
```

In *Algorithm* 2, *Line* 1 is for initialization, where max and $\alpha^*(t)$ will store the maximal value of $\mathcal{Q}_k(t)\cdot b_k(t)$ and the optimal data transfer decision obtained in each iteration, respectively, and $\hat{C}$ records the available throughput that can be provided by the link $e_t=(v,u)$ within Δt. Then, the while-loop tries to distribute all the throughput of $\hat{C}$ to the bundles in $\{\mathcal{Q}_k(t)\}$ (*Lines* 2-26). Specifically, in each iteration of the while-loop, we first check all the virtual queues and find the one that transferring the bundles in it can maximize $\mathcal{Q}_k(t)\cdot b_k(t)$ (*i.e.*, to maximize the term in Eq. (11)) (*Lines* 3-19), and then send the selected bundles and update the available throughput accordingly (*Lines* 20-25). Note that, in *Lines* 23-24, we will set $\hat{C}$ as 0 if we have $b_k^*(t)=0$. This is because $b_k^*(t)=0$ means that either all the bundles in $\{\mathcal{Q}_k(t)\}$ have been transmitted or the remaining throughput cannot cover any pending bundle in $\{\mathcal{Q}_k(t)\}$. Therefore, the while-loop of data scheduling should be terminated.

C. Time Complexity Analysis

The while-loop of *Lines* 2-26 in *Algorithm* 2 can run $K+1$ times at most, while the for-loop of *Lines* 3-19 will execute for K times. The complexity of the sorting in *Line* 5 can be

assumed to be $O(n^2)$, where n denotes the average length of $\mathcal{Q}_k(t)$. As for the while-loop of *Lines* 6-13, it can run $\eta = \frac{\hat{C}}{\overline{\beta}}$ times, where $\overline{\beta}$ is the average bundle size. Hence, the time complexity of *Algorithm* 2 is $O(K^2 \cdot (n^2 + \eta))$.

IV. Performance Evaluation

A. Simulation Setup

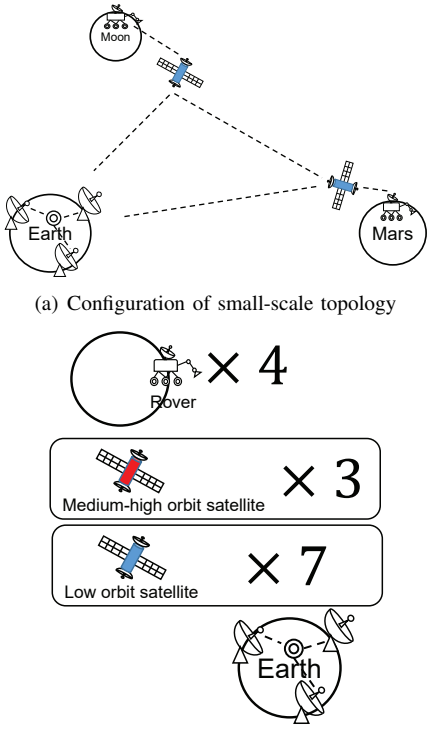
Our simulations consider two IPN topologies. The small-scale topology contains 8 nodes spanning across Earth, Moon, and Mars, while the large-scale one includes 18 nodes. Figs. 2(a) and 2(b) illustrate the configurations of the small-scale and large-scale topologies, respectively. We use the satellite tool kit (STK) [16] to generate the 24-hour contact plan of each topology. The simulation parameters use similar settings as those in [13]. Specifically, the bundle requests are dynamically generated according to the Poisson traffic model. For each request $B(s, d, \beta, t_a, t_d)$, its source s and destination d are randomly selected, its data size β distributes uniformly within $[1, 1024]$ KByte, and the average value of its lifetime $(t_d - t_a)$ is set as $7,200$ seconds. The duration of each TS is set as $\Delta t = 256$ seconds, to ensure that the data transfer of each bundle is atomic. More specifically, the setting of Δt guarantees that the largest bundle of 1 MByte can be transmitted over the link with the lowest data rate (*i.e.*, 32 Kbps) within a TS.

We use the CGR that processes bundles in the FIFO manner (CGR) and the MARS algorithm in [13] as the benchmarks, and evaluate the algorithms' performance on interplanetary data transfers (IP-DTs) in terms of delivery ratio and average end-to-end (E2E) latency of IP-DTs. To ensure sufficient statistical accuracy, the simulations average the results of 5 independent runs to get each data point. Note that, the authors of [13] considered different weight configurations in MARS. Specifically, the weights address the size (w_1), remaining lifetime (w_2), priority (w_3), and time in queue (w_4) of each buffered bundle request. Due to the page limit, we adopt two typical configurations of MARS in our simulations: 1) MARS-1 with $\{w_1 = 0.1,\ w_2 = 0.4,\ w_3 = 0.4,\ w_4 = 0.1\}$, and MARS-2 with $\{w_1 = 0.5,\ w_2 = 0.1,\ w_3 = 0.1,\ w_4 = 0.3\}$.

B. Small-Scale Simulations

As shown in Fig. 2(a), the small-scale IPN topology consists of three ground stations and a ground control center on Earth, as well as two communication systems for Moon and Mars, respectively, each of which includes a rover and a relay satellite. Note that, Fig. 2(a) only shows a snap-shot of the IPN, where the satellites only communicate with one of the ground stations, but each satellite actually can talk with all the ground stations at the right time of contacts, respectively.

Fig. 3(a) shows the results on average E2E latency of IP-DTs, which indicate that our Lyapunov-based algorithm (Lyapunov) achieves significantly shorter average E2E latency than the benchmarks. This verifies the effectiveness of the distributed routing and scheduling in Lyapunov. Meanwhile, it is interesting to observe that the average E2E latencies from the algorithms actually decrease with the traffic load slightly



(a) Configuration of small-scale topology

(b) Configuration of large-scale topology

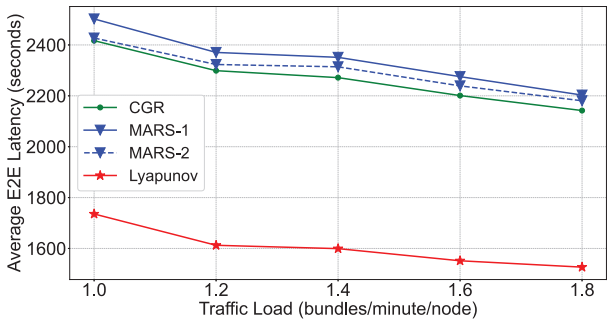
Fig. 2. IPN topologies used in simulations.

in Fig. 3(a). This phenomenon can be explained as follows. The connectivity of the small-scale IPN topology in Fig. 2(a) is actually not very good due to the sparse and time-varying links in it. Hence, when the traffic load increases, the IP-DTs whose latencies have already been relatively long have a larger chance to be dropped by the intermediate nodes on their routing paths, and this helps reduce the average E2E latency slightly.

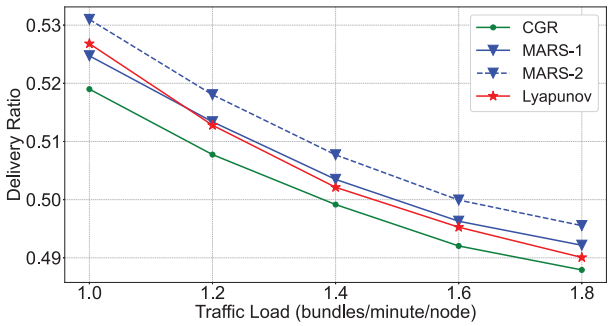
As for the MARS-based algorithms, their average E2E latencies are slightly longer than CGR. We believe that this is because the scheduling in MARS helps more IP-DTs reach their destinations successfully. The results on delivery ratio in Fig. 3(b) confirm this analysis, which shows that the MARS-based algorithms achieves slightly larger delivery ratios than CGR. Meanwhile, by combining the results in Figs. 3(a) and 3(b), we can see that MARS can adjust the tradeoff between average E2E latency and delivery ratio by using different weight configurations. Specifically, the weight configuration of MARS-2 is better because it provides larger delivery ratios and shorter average E2E latency. On the other hand, by comparing MARS-2 and Lyapunov, we find that Lyapunov adjusts the tradeoff between average E2E latency and delivery ratio even better. This is because the delivery ratio of Lyapunov is only slightly less than that of MARS-2 (*i.e.*, the gap is typically less than 1%), while the average E2E latency of Lyapunov is much shorter than that of MARS-2.

C. Large-Scale Simulations

To further evaluate the algorithms in a larger-scale topology with more complex intermittent connections, we simulate them with the IPN topology in Fig. 2(b), which consists of 3 ground stations, 1 ground control center, 4 rovers, 7 low orbit

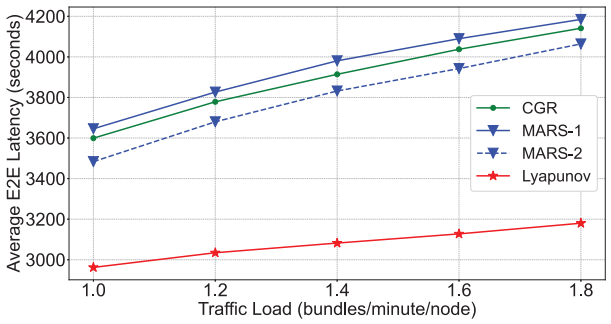


(a) Average E2E latency

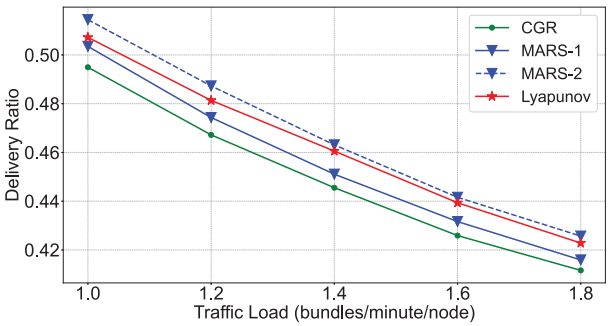


(b) Delivery ratio

Fig. 3. Results of simulations with small-scale topology.



(a) Average E2E delay



(b) Delivery ratio

Fig. 4. Results of simulations with large-scale topology.

satellites, and 3 medium-high orbit satellites on/around Earth, Moon, and Mars. This time, the average E2E latencies in Fig. 4(a) generally increase with the traffic load, because of the much better connectivity of the large-scale IPN topology. We can see that Lyapunov provides the shortest average E2E latency among all the algorithms, and its results are still significantly shorter than those of the benchmarks. Meanwhile, the results in Fig. 4(b) suggest that Lyapunov outperforms MARS-1 and CGR in terms of delivery ratio, and the performance gap of Lyapunov and MARS-2 on delivery ratio is smaller than that in Fig. 3(b). Therefore, Fig. 4 verifies that Lyapunov still adjusts the tradeoff between average E2E latency and delivery ratio the best among all the algorithms, and it actually performs better in a larger scale IPN topology.

V. Conclusion

In this work, we proposed an online approach to schedule and route IP-DTs distributedly in an IPN. We leveraged the Lyapunov optimization to design the distributed online approach, such that it can optimize the performance of IP-DTs with only the information about local queues on each node in an IPN. Our simulation results confirmed that our proposed algorithm outperforms the existing ones significantly in terms of the average E2E latency of IP-DTs, and can properly adjust the tradeoff between average E2E latency and delivery ratio.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (2020YFB1806400), NSFC project 61871357, SPR Program of CAS (XDC02070300), and Fundamental Funds for Central Universities (WK3500000006).

References

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[3] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[4] M. Marchese, "Interplanetary and pervasive communications," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 26, pp. 12–18, Feb. 2011.

[5] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[6] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[7] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[8] A. Alhilal, T. Braud, and P. Hui, "The sky is NOT the limit anymore: Future architecture of the interplanetary Internet," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 34, pp. 22–32, Aug. 2019.

[9] S. Burleigh *et al.*, "Delay-tolerant networking: an approach to interplanetary Internet," *IEEE Commun. Mag.*, vol. 41, pp. 128–136, Jun. 2003.

[10] K. Scott and S. Burleigh, "Bundle protocol specification," *RFC 5050*, Nov. 2007. [Online]. Available: http://tools.ietf.org/html/rfc5050.

[11] S. Burleigh, "Contact graph routing," Jul. 2010. [Online]. Available: https://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-01.

[12] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, and V. Tsaoussidis, "Towards flexibility and accuracy in space DTN communications," in *Proc. of ACM CHANTS 2013*, pp. 43–48, Sept. 2013.

[13] S. El Alaoui and B. Ramamurthy, "MARS: A multi-attribute routing and scheduling algorithm for DTN interplanetary networks," *IEEE/ACM Trans. Netw.*, vol. 28, pp. 2065–2076, Oct. 2020.

[14] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan and Claypool, 2010.

[15] D. Bertsekas, R. Gallager, and P. Humblet, *Data Networks*. New Jersey, USA: Prentice-Hall International, 1992.

[16] Satellite tool kit. [Online]. Available: http://www.agi.com/products/stk/.