A Dynamic Combined Flow Algorithm for the Two-Commodity Max-Flow Problem Over Delay-Tolerant Networks

Tao Zhang, Hongyan Li, *Member, IEEE*, Jiandong Li, *Senior Member, IEEE*, Shun Zhang, *Member, IEEE*, and Haiying Shen, *Senior Member, IEEE*

***Abstract*— The multi-commodity flow problem plays an important role in network optimization, routing, and service scheduling. With the network partitioning and the intermittent connectivity, the commodity flows in delay tolerant networks (DTNs) are time-dependent, which is very different from that over the static networks. As an NP-hard problem, existing works can only obtain sub-optimal results on maximizing the multi-commodity flow of dynamic networks. To overcome these bottlenecks, in this paper, we propose a graph-based algorithm to solve the maximum two-commodity flow problem over the DTNs. Through analyzing the relationship between the two commodities, we propose a maximum two-commodity flow theorem to simplify the coupling two-commodity flow problem as the two single-commodity flow ones. Then, with the help of the storage time aggregated graph (STAG) (a DTN model with less memory), we construct a pair of flow graphs to describe the reduced two single-commodity flows (addition flow and subtraction flow), and design the corresponding flow calculation methods. Moreover, we design a STAG-based dynamic combined flow algorithm to maximize the two-commodity flow. Finally, the computational complexity of the proposed algorithm is analyzed, and its efficacy has also been demonstrated through an illustrative example and numerical simulations.**

***Index Terms*— Delay tolerant networks, satellite networks, network capacity, storage time aggregated graph, two-commodity maximum flow.**

Manuscript received January 15, 2018; revised July 20, 2018; accepted September 18, 2018. Date of publication October 5, 2018; date of current version December 10, 2018. The work of T. Zhang, H. Li, J. Li, and S. Zhang was supported in part by the National Natural Science Foundation of China under Grant 91638202, Grant 61871456, Grant 61401326, and Grant 61571351, in part by the National Key Research and Development Program of China under Grant 2016YFB0501004, in part by the National S&T Major Project under Grant 2015ZX03002006, in part by the 111 Project under Grant B08038, and in part by the Natural Science Basic Research Plan in Shaanxi Province of China under Grant 2016JQ6054. The work of H. Shen was supported in part by the U.S. NSF Grants OAC-1724845, ACI-1719397, and CNS-1733596, and in part by the Microsoft Research Faculty Fellowship 8300751. The associate editor coordinating the review of this paper and approving it for publication was M. Li. *(Corresponding author: Hongyan Li.)*

T. Zhang, H. Li, J. Li, and S. Zhang are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: taozhangfsz@gmail.com; hyli@mail.xidian.edu.cn; jdli@mail.xidian.edu.cn; zhangshunsdu@xidian.edu.cn).

H. Shen is with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903 USA (e-mail: hs6ms@virginia.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2018.2872551

I. Introduction

SINCE the maximum flow can effectively depict the maximal transmission capacity of a network, it has been widely applied in network optimization, routing and service scheduling [1]–[4]. Especially, for the goods transportation [2], the information transmission [3] and the vehicle deployment [4], the goods, the messages, or the vehicles are to be transmitted from several places to some others, and the multiple commodities (various flow demands) between different sources and destinations appear. In such scenarios, the maximal network capacity can be achieved through solving one multi-commodity max-flow problem. Besides, with the help of the multi-commodity flow theory, both the end-to-end paths and the accurately resource allocation schemes can be calculated, which can support the flow assignment of each commodity. Therefore, the multi-commodity max-flow problem plays an important role in current information, logistics and transportation networks.

Recently, delay tolerant networks (DTNs) have attracted a lot of attentions, owing to their widely applications in social networks [5], sensor networks [6], satellite networks [7], and ad hoc networks [8]. In order to satisfy the transmission requirements of different services, it is necessary to solve the multi-commodity max-flow problem over DTNs. However, in a DTN, there are almost no consistent end-to-end paths from the sources to the destinations, which is caused by the node mobility, the wireless propagation effects, the sparse node density, and the other adverse factors [9]. As such, the commodity flows in this kind of networks are time-dependent, which are very different from that over the static networks. Actually, DTNs are typical dynamic networks [10], which can also be treated as the temporal networks [11]. Therefore, the multi-commodity max-flow problem in DTNs would confront critical challenges. First, because of the time-varying characteristics, it is difficult to construct one representation scheme to accurately describe both the network model and the node buffers. Furthermore, there are complex resource competitions and coupling relationships among multiple commodities, which means that the multi-commodity max-flow problem can not be simply treated as a combination of several single commodity flow problems.

In fact, many excellent works [12]–[22] have been done about the maximum flow problem in the both static and

dynamic networks. The Ford-Fulkerson augmenting path algorithm [12] has been treated as the most classic graph theory based method for solving the maximum flow problem of the static networks. Furthermore, with the help of time expended graphs (TEG), it can be extended for the dynamic networks [13]. However, the method focused on the single commodity maximum flow problem, and could not solve the coupling relationships of multiple commodities. Hu [14] designed an efficient graph-based algorithm to find the maximum two-commodity flow over an undirected static network. Nevertheless, this algorithm did not consider the time-varying network characteristics, and could not be applied for the dynamic multi-commodity maximum flow problem, especially in DTNs. Hall *et al.* [18] showed that the multi-commodity flow over time problem is NP-hard. Furthermore, for the dynamic multi-commodity maximum flow problem, most of existing graph-based works [18]–[20] could only obtain a fully polynomial time approximation scheme, but not the optimal methods.

In our previous work [22], we modeled the DTN as a storage time aggregated graph (STAG), which needed less memory to depict the time variant topology and link capacity than TEG did. Correspondingly, a bidirectional storage transfer series was carefully designed to describe the node buffers. Moreover, a low-complex STAG-based algorithm was proposed to obtain the maximum network flow. However, only one source and one destination were considered. Therefore, in this paper, we will focus on the maximum two-commodity flow problem in a DTN network, and will propose a dynamic combined flow algorithm to obtain the maximum two commodities flow. The contributions of this paper are summarized as follows:

- To entirely depict the data storage process of each node and the time variant characteristics of the network, the STAG will be utilized to model a DTN. Besides, the modified bidirectional storage transfer series and corresponding transfer rule are designed to facilitate the calculation of the STAG-based commodity flows.
- After analysing the relationship of the two commodities, we will introduce a maximum two-commodity flow theorem. Specifically, we will construct a pair of duality flows (addition flow and subtraction flow) and the time-related flow constraint to transform the two-commodity coupling relationship into the independent ones. Hence, the two-commodity flow problem will be simplified as the two single-commodity flow ones.
- With the modified bidirectional storage transfer series and corresponding transfer rule, we will separately maximize the duality flows in both the addition flow graph STAG^{+} and the substraction flow graph STAG^{-}. Then, in the undirected STAG, the *dynamic combined flow* (DCF) algorithm will be proposed to maximize the two-commodity flow of a DTN.
- An illustrative example is given for understanding the proposed algorithm. The complexity analysis about this proposed algorithm will be presented. Finally, the algorithm's efficacy will be demonstrated through numerical simulations.

The remainder of this paper is organized as follows. The related works about the topic of this paper is summarized in Section II. In Section III, we present the system model STAG. Section IV provides the maximum two-commodity flow theorem. In Section V, we propose and describe the STAG-based maximum two-commodity flow algorithm in detail. Then, we give the analysis of the algorithm in Section VI. Simulations are discussed In Section VII. Finally, we conclude this work and discuss the future works in Section VIII.

II. Related Work

A. Graph Model for Modeling DTNs

The graph theory can effectively model the network through a set of nodes and edges. Generally, the achieved graph can be classified into the directed graph and the undirected one. Recently, many time-varying graph-based works have been done over the DTNs [21]–[29]. The authors in [23] constructed the snapshots graph (or temporal graph), where each snapshot corresponded to the network topology at a given time interval. However, there was no correlation between snapshots, and massive snapshots would be generated in a long time period. To overcome these bottlenecks, in [21], [24], the time expanded graph (TEG) was utilized to replicate the network at each interval, where any two time intervals of the same node were connected via one link. Similarly, some TEG-based graphs have also been designed, such as the contact graph [25], the event-driven graph [26], the space-time graph [27], the time-evolving graph [29], etc. But, both the TEG and TEG-based graphs required the large memory resources, and were high-complex. On the contrary, the authors in [28] proposed the time aggregated graph (TAG) to formulate the time variant characteristics of network as one specific timing series. Because of no network replication operation, the TAG needed less memory and possessed the low-complexity. However, neither the correlation between time intervals nor nodes storage were considered in TAG. Therefore, in our previous work [22], we modified the TAG as storage time aggregated graphs (STAG), and designed the bidirectional storage transfer series to thoroughly depict the store-carry-and-forward mechanism [30] of a DTN.

B. Maximum Flow in Static Networks

For nearly half a century, the maximum flow problem has drawn many researchers' attention. With respect to the static networks, the earliest work about the maximum flow problem was done by Ford and Fulkerson [12], who proposed a graph-based labeling (or augmenting path) algorithm from the max-flow min-cut theory. On the basis of this interesting work, some more elaborate algorithms, such as the shortest augmenting path algorithm [16], the capacity scaling algorithm [17], etc., were proposed. However, these methods focused on the single-commodity max-flow problem. Hu [14] proposed an algorithm to find the maximum two-commodity flow in an undirected graph, where the algorithm complexity was bounded by the size of capacities. In order to decrease the algorithm complexity, the authors in [15] gave an improved

Hu's two-commodity flow algorithm. However, it has been proved that finding maximum two or more commodities flow in a directed graph is as difficult as linear programming, and the multiple (more than two) commodities flow problems in undirected graph are NP-complete problems [15]. Recently, the authors in [20] designed a planarity-exploiting algorithm in a near-liner time to solve the multiple-source multiple-sink maximum flow. But, it can be only applied for a directed planar graph. Usually, the multi-commodity (with multiple sources and destinations) problem in graph is reduced to the single-source single-destination case, by introducing an virtual source and sink. However, this method cannot obtain optimal results because of the interaction of different commodities.

C. Maximum Flow in Dynamic Networks

In terms of dynamic networks, Ford and Fulkerson [13] firstly reduced the flow-over-time problems into static ones via TEGs, and proposed an efficient algorithm for the maximum single-commodity flow within a given time horizon. Besides, the authors in [21] analyzed the impact of nodes storage, and implemented a TEG-based joint routing and storage control (usage) optimization for the case of a single commodity transmission over dynamic networks. In order to overcome the bottleneck of the TEG's large memory resources, in our previous work [22], we modeled the DTN as STAG, and proposed a low-complex STAG-based maximum flow algorithm to cope with the single commodity maximum flow problem. However, for the case of multiple commodities, Hall *et al.* [18] showed that the problem of the multi-commodity flow over time is NP-hard, even in the series-parallel graphs or to the case of only two commodities. Subsequently, the authors in [19] proposed a fully polynomial time approximation scheme to maximize the multi-commodity flows over time without intermediate storage. And the authors in [31] formulated the time-depended multi-commodity network via linear programming method, and designed a heuristic algorithm to minimize the joint costs of both the routing commodities and the activating arcs throughout the horizon. Although, many works have focused on the dynamic multi-commodity flow problem, there is still no optimal graph-based method for solving it.

III. System Model

In this section, we resort to an undirected time-varying STAG to model the predicted DTNs, where some STAG-based definitions are presented for facilitating the computation of maximum flow.

A. STAG for Modeling Predicted DTNs

In this paper, we exploit an undirected STAG to describe the time-varying characteristics of the predicted DTNs, including the topology structure, the buffer sizes of nodes, and the capacity of edges. Although the networks are changing over time, these features are well predicted. For example, in satellite networks, since the motion trajectory and motion period of each satellite can be known in advance, the communication opportunities between satellites are predictable [25].

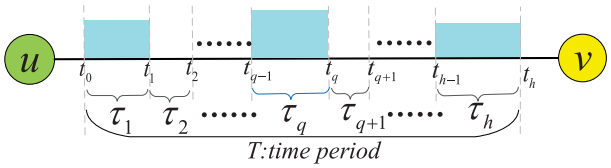


Fig. 1. Predicted connectivity of an edge during time period T.

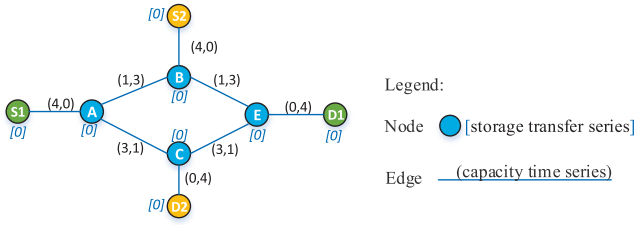


Fig. 2. Example of the storage time aggregated graph.

In particular, a time intervals system is adopted. As shown in Fig. 1, for a given time period $T = [t_0, t_h)$, we can divide it into h small time intervals $\{\tau_1, \ldots, \tau_q, \ldots, \tau_h\}$, where $\tau_q = [t_{q-1}, t_q), 1 \le q \le h$. Obviously, the edge between u and v is intermittent connectivity, and has been separated into h segments by the partitioned time intervals. Therefore, the predicted DTNs could be characterized by an undirected graph as STAG = $\{(\mathcal{V},\ \mathcal{E}, T, C_{u,v}(T), N_v(T)) | u \in \mathcal{V},\ v \in \mathcal{V},\ [u,v] \in \mathcal{E}\}$, where

- $\mathcal{V}$ is the set of nodes;
- $\mathcal{E}$ is the set of edges. Note that $[u, v]$ denotes an undirected edge between node u and v, while (u, v) represents a directed edge from node u to v;
- T represents the given time period;
- $C_{u,v}(T) = (c_{u,v}(\tau_1), \ldots, c_{u,v}(\tau_q), \ldots, c_{u,v}(\tau_h))$ is a capacity time series for the edge $[u, v]$, where $c_{u,v}(\tau_q) = \int_{\tau_q} w_{u,v}(t)dt$ is the total capacity during $\tau_q = [t_{q-1}, t_q)$, while $w_{u,v}(t)$ is the capacity of the edge $[u, v]$ at a time instant $t \in \tau_q$;
- $N_v(T) = [n_v(\tau_1, \tau_2), \ldots, n_v(\tau_{q-1}, \tau_q), \ldots, n_v(\tau_{h-1}, \tau_h)]$ is a bidirectional storage transfer series of node v, and depicts the amount of data stored and carried in a node during the time period T. Notice that $n_v(\tau_{q-1}, \tau_q)$ is the amount of the transferred data between the adjacent time intervals τ_{q-1} and τ_q.

For clearness, we present a STAG with two pairs of source-destination nodes $\{\mathbb{S}_1, \mathbb{D}_1\}$ and $\{\mathbb{S}_2, \mathbb{D}_2\}$ in Fig. 2. It can be seen from Fig. 2 that each node has a bidirectional storage transfer series, and the capacity of each undirected edge is denoted as a capacity time series.

B. Basic Definitions in STAG

Unlike the constant flow in static networks, the feasible flow over the DTNs is a time-dependent function, which can be explained as follows. For the DTNs, the links represent intermittent connectivity, and the feasible flow of one specific path is constrained by the time-dependent connectivity of its both the front and the back links, and the buffer sizes of nodes.

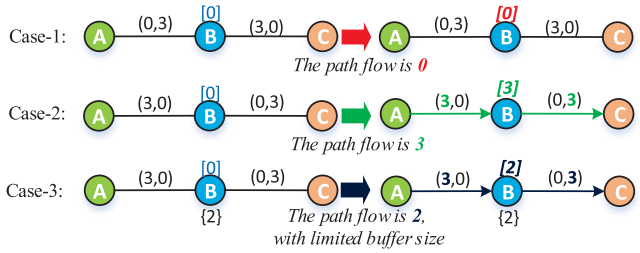


Fig. 3. The feasible flow of a path in different cases.

For example, we present three different connection cases for the specific path $(\mathbb{A}-\mathbb{B}-\mathbb{C})$ in Fig. 3. It should be noticed that only in case-3 the node $\mathbb{B}$ has a limited buffer $\{2\}$. It can been checked that the number of the feasible flows for this path over the case-1, the case-2, and the case-3 are 0, 3, and 2, respectively.

Before proceeding, we present some STAG-based definitions about the path, feasible flow, node's storage, etc.

1) Path in STAG: For the given source node $\mathbb{S}$ and destination node $\mathbb{D}$, the path from $\mathbb{S}$ to $\mathbb{D}$ is defined as the forward path l^+, while that from $\mathbb{D}$ to $\mathbb{S}$ is called as the reverse path l^-.

2) Dynamic Feasible Flow: In the STAG, we wish to send a flow from $\mathbb{S}$ to $\mathbb{D}$ within the capacity of any edge along the corresponding forward path. Therefore, we can denote the numbers of the dynamic feasible flows at different time intervals within the period T as a time series $f(T)=(f(\tau_1),\ldots,f(\tau_q),\ldots,f(\tau_h))$, which should satisfy the following constraint as:

- The capacity constraint ($\forall u\in\mathcal{V},\forall v\in\mathcal{V}$):

$$0\le f_{u,v}(\tau_q)\le c_{u,v}(\tau_q)\quad \forall 1\le q\le h,\ \forall[u,v]\in\mathcal{E}. \tag{1}$$

- The flow conservation ($\forall[u,v]\in\mathcal{E}$):

$$\sum_{v\in\mathcal{V}}\sum_{q=1}^{h} f_{u,v}(\tau_q)-\sum_{v\in\mathcal{V}}\sum_{q=1}^{h} f_{v,u}(\tau_q) = \begin{cases} \sum_{q=1}^{h} f(\tau_q), & u=\mathbb{S},\\ 0, & u\ne\mathbb{S},\mathbb{D},\\ -\sum_{q=1}^{h} f(\tau_q), & u=\mathbb{D}. \end{cases} \tag{2}$$

- The storage-transfer constraint:

For the node $v\in\mathcal{V}$, the elements of the bidirectional storage transfer series $N_v(T)$ are closely related to all the feasible flows into and out from the node v. Explicitly, the amount of the transferred data $n_v(\tau_1,\tau_2)$ at the time interval τ_1 can be listed as

$$\sum_{u\in\mathcal{V}} f_{u,v}(\tau_1)-\sum_{u\in\mathcal{V}} f_{v,u}(\tau_1)=n_v(\tau_1,\tau_2)\ge 0, \tag{3}$$

but the cached data in the previous time intervals should be taken into consideration to derive $n_v(\tau_q,\tau_q+1)$ for the time intervals $\tau_q, 1<q<h$, as

$$n_v(\tau_{q-1},\tau_q)+\sum_{u\in\mathcal{V}} f_{u,v}(\tau_q)-\sum_{u\in\mathcal{V}} f_{v,u}(\tau_q)=n_v(\tau_q,\tau_q+1). \tag{4}$$

Therefore, the storage-transfer constraint can be written as

$$\sum_{i=1}^{q}\sum_{u\in\mathcal{V}} f_{u,v}(\tau_i)-\sum_{i=1}^{q}\sum_{u\in\mathcal{V}} f_{v,u}(\tau_i) = \begin{cases} 0, & q=h,\\ n_v(\tau_q,\tau_{q+1})\ge 0, & 1\le q<h. \end{cases} \tag{5}$$

3) Residual STAG (rSTAG): the rSTAG consists of the new added directed edges with nonzero capacities, and the nodes with the bidirectional storage transfer series. Specifically, for the given STAG with feasible flow $f(T)$ and a pair of vertices $u\in\mathcal{V},\ v\in\mathcal{V}$, the residual capacity time series $rc_{u,v}(T)=(rc_{u,v}(\tau_1),\ldots,rc_{u,v}(\tau_q),\ldots,rc_{u,v}(\tau_h))$ of the edge (u,v) can be denoted as (6), where $\forall 1\le q\le h$,

$$rc_{u,v}(\tau_q)=c_{u,v}(\tau_q)-f_{u,v}(\tau_q),\quad \text{if } (u,v)\in\mathcal{E}. \tag{6}$$

Correspondingly, we can define the residual capacity time series $rc_{v,u}(T)=(rc_{v,u}(\tau_1),\ldots,rc_{v,u}(\tau_q),\ldots,rc_{v,u}(\tau_h))$ for the edge (v,u) as

$$rc_{v,u}(\tau_q)=\begin{cases} rc_{v,u}(\tau_q)+f_{u,v}(\tau_q), & \text{if } (v,u)\in\mathcal{E},\\ f_{u,v}(\tau_q), & \text{if } (v,u)\notin\mathcal{E}. \end{cases} \tag{7}$$

IV. Maximum Two-Commodity Flow Theorem

In this section, the maximum two-commodity flow problem with dynamic flow conservation, capacity and storage-transfer constraint are introduced. Meanwhile, we construct a pair of duality flows to reduce the two-commodity flow problem into two single-commodity flow ones.

A. Maximum Two-Commodity Flow Problem

Let us consider an undirected STAG $=\{(\mathcal{V},\ \mathcal{E},T,C_{u,v}(T),N_v(T))|u\in\mathcal{V},\ v\in\mathcal{V},\ [u,v]\in\mathcal{E}\}$ with n nodes and m edges. It is assumed that there are no parallel edges and self-loops in this graph. In this STAG, the given two pairs of the source-destination nodes $\{\mathbb{S}_1,\mathbb{D}_1\}$ and $\{\mathbb{S}_2,\mathbb{D}_2\}$ will have different commodities, whose corresponding flows from u to v along the edge $[u,v]$ can be separately written as

$$f^1_{u,v}(T)=\sum_{q=1}^{h} f^1_{u,v}(\tau_q),\quad f^2_{u,v}(T)=\sum_{q=1}^{h} f^2_{u,v}(\tau_q). \tag{8}$$

Then, the maximum two-commodity flow problem for $\{\mathbb{S}_1,\mathbb{D}_1\}$ and $\{\mathbb{S}_2,\mathbb{D}_2\}$ can be formulated as

$$\max\{F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)+F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)\}, \tag{9}$$

where $F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)$ (or $F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)$) represents the total flow from $\mathbb{S}_1$ to $\mathbb{D}_1$ (or from $\mathbb{S}_2$ to $\mathbb{D}_2$), and can be explicitly written as

$$\begin{cases} F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)=\sum_{v\in\mathcal{V}} f^1_{\mathbb{S}_1,v}(T)-\sum_{v\in\mathcal{V}} f^1_{v,\mathbb{S}_1}(T),\\ F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)=\sum_{v\in\mathcal{V}} f^2_{\mathbb{S}_2,v}(T)-\sum_{v\in\mathcal{V}} f^2_{v,\mathbb{S}_2}(T). \end{cases} \tag{10}$$

As claimed in the above section, the commodity flows related to the path from $\mathbb{S}_1$ to $\mathbb{D}_1$ and $\mathbb{S}_2$ to $\mathbb{D}_2$ should satisfy the dynamic conservation, the capacity and the storage-transfer constraint as follows:

1) Flow conservation ($\forall u, v \in \mathcal{V}$):

$$\sum_{v\in\mathcal{V}} f^1_{u,v}(T) - \sum_{v\in\mathcal{V}} f^1_{v,u}(T) = \begin{cases} F^1_{\mathbb{S}_1,\mathbb{D}_1}(T), & u = \mathbb{S}_1, \\ 0, & u \neq \mathbb{S}_1, \mathbb{D}_1, \\ -F^1_{\mathbb{S}_1,\mathbb{D}_1}(T), & u = \mathbb{D}_1. \end{cases} \tag{11}$$

$$\sum_{v\in\mathcal{V}} f^2_{u,v}(T) - \sum_{v\in\mathcal{V}} f^2_{v,u}(T) = \begin{cases} F^2_{\mathbb{S}_2,\mathbb{D}_2}(T), & u = \mathbb{S}_2, \\ 0, & u \neq \mathbb{S}_2, \mathbb{D}_2, \\ -F^2_{\mathbb{S}_2,\mathbb{D}_2}(T), & u = \mathbb{D}_2. \end{cases} \tag{12}$$

2) Capacity constraint ($\forall [u, v] \in \mathcal{E}$):

$$|f^1_{u,v}(\tau_q)| + |f^2_{u,v}(\tau_q)| \leq c_{u,v}(\tau_q), \quad q = 1, \ldots, h. \tag{13}$$

3) Storage-transfer constraint ($\forall v \in \mathcal{V},\ k = 1, 2$):

$$\sum_{i=1}^{q}\sum_{u\in\mathcal{V}} f^k_{u,v}(\tau_i) - \sum_{i=1}^{q}\sum_{u\in\mathcal{V}} f^k_{v,u}(\tau_i) = \begin{cases} 0, & q = h, \\ n^k_v(\tau_q, \tau_{q+1}) \geq 0, & 1 \leq q < h. \end{cases} \tag{14}$$

B. Transformation of Maximum Two-Commodity Flow Problem

1) Construction of Both the Addition and Subtraction Flows: It can be seen from (13) that two commodities are not independent of each other, and this coupling relationship complicates the two-commodity flow problem. However, after some simple mathematical operations, we can rewrite it as

$$\begin{aligned} c_{u,v}(\tau_q) &\geq |f^1_{u,v}(\tau_q)| + |f^2_{u,v}(\tau_q)| \\ &= \max\{|f^1_{u,v}(\tau_q) + f^2_{u,v}(\tau_q)|, |f^1_{u,v}(\tau_q) - f^2_{u,v}(\tau_q)|\}, \end{aligned} \tag{15}$$

$$\Longleftrightarrow \begin{cases} |f^1_{u,v}(\tau_q) + f^2_{u,v}(\tau_q)| \leq c_{u,v}(\tau_q), \\ |f^1_{u,v}(\tau_q) - f^2_{u,v}(\tau_q)| \leq c_{u,v}(\tau_q). \end{cases} \forall [u, v] \in \mathcal{E}. \tag{16}$$

which means that the addition flow $(f^1_{u,v}(\tau_q) + f^2_{u,v}(\tau_q))$ and the subtraction flow $(f^1_{u,v}(\tau_q) - f^2_{u,v}(\tau_q))$ can be separately constrained, and the coupling relationship of two commodity can be reduced into independent one. For notational simplicity, we introduce a pair of duality flows $\delta^+_{u,v}(\tau_q)$ and $\delta^-_{u,v}(\tau_q)$ to denote the addition and subtraction flows as,

$$\begin{cases} \delta^+_{u,v}(\tau_q) = f^1_{u,v}(\tau_q) + f^2_{u,v}(\tau_q), \\ \delta^-_{u,v}(\tau_q) = f^1_{u,v}(\tau_q) - f^2_{u,v}(\tau_q), \end{cases} \forall [u, v] \in \mathcal{E}. \tag{17}$$

Thus, it can be easily checked that,

$$\begin{cases} f^1_{u,v}(\tau_q) = \dfrac{1}{2}\{\delta^+_{u,v}(\tau_q) + \delta^-_{u,v}(\tau_q)\}, \\ f^2_{u,v}(\tau_q) = \dfrac{1}{2}\{\delta^+_{u,v}(\tau_q) - \delta^-_{u,v}(\tau_q)\}, \end{cases} \forall [u, v] \in \mathcal{E}. \tag{18}$$

2) Equivalent Constraint: Substituting (17) into (11), (12), and (13), and taking some reorganizations, we can derive the equivalent flow conservation and flow constraint on the terms $\delta^+_{u,v}(\tau_q)$ and $\delta^-_{u,v}(\tau_q)$ as

$$\begin{aligned} &\sum_{v\in V}\sum_{q=1}^{h} \delta^+_{u,v}(\tau_q) - \sum_{v\in V}\sum_{q=1}^{h} \delta^+_{v,u}(\tau_q) \\ &\quad = \begin{cases} F^1_{\mathbb{S}_1,\mathbb{D}_1}(T), & u = \mathbb{S}_1, \\ F^2_{\mathbb{S}_2,\mathbb{D}_2}(T), & u = \mathbb{S}_2, \\ 0, & u \neq \mathbb{S}_1, \mathbb{D}_1, \mathbb{S}_2, \mathbb{D}_2 \\ -F^1_{\mathbb{S}_1,\mathbb{D}_1}(T), & u = \mathbb{D}_1, \\ -F^2_{\mathbb{S}_2,\mathbb{D}_2}(T), & u = \mathbb{D}_2. \end{cases} \end{aligned} \tag{19}$$

$$\begin{aligned} &\sum_{v\in V}\sum_{q=1}^{h} \delta^-_{u,v}(\tau_q) - \sum_{v\in V}\sum_{q=1}^{h} \delta^-_{v,u}(\tau_q) \\ &\quad = \begin{cases} F^1_{\mathbb{S}_1,\mathbb{D}_1}(T), & u = \mathbb{S}_1, \\ -F^2_{\mathbb{S}_2,\mathbb{D}_2}(T), & u = \mathbb{S}_2, \\ 0, & u \neq \mathbb{S}_1, \mathbb{D}_1, \mathbb{S}_2, \mathbb{D}_2 \\ -F^1_{\mathbb{S}_1,\mathbb{D}_1}(T), & u = \mathbb{D}_1, \\ F^2_{\mathbb{S}_2,\mathbb{D}_2}(T), & u = \mathbb{D}_2. \end{cases} \end{aligned} \tag{20}$$

$$|\delta^+_{u,v}(\tau_q)| \leq c_{u,v}(\tau_q), \quad q = 1, \ldots, h. \tag{21}$$

$$|\delta^-_{u,v}(\tau_q)| \leq c_{u,v}(\tau_q), \quad q = 1, \ldots, h. \tag{22}$$

Similarly, with (17) and (14), we can reexpress the storage-transfer constraint as

$$\begin{aligned} &\sum_{i=1}^{q}\sum_{u\in\mathcal{V}} \delta^+_{u,v}(\tau_i) - \sum_{i=1}^{q}\sum_{u\in\mathcal{V}} \delta^+_{v,u}(\tau_i) \\ &\quad = \begin{cases} 0, & q = h, \\ n^1_v(\tau_q, \tau_{q+1}) + n^2_v(\tau_q, \tau_{q+1}), & 1 \leq q < h. \end{cases} \\ &\quad = \begin{cases} 0, & q = h, \\ n^+_v(\tau_q, \tau_{q+1}) \geq 0, & 1 \leq q < h. \end{cases} \end{aligned} \tag{23}$$

$$\begin{aligned} &\sum_{i=1}^{q}\sum_{u\in\mathcal{V}} \delta^-_{u,v}(\tau_i) - \sum_{i=1}^{q}\sum_{u\in\mathcal{V}} \delta^-_{v,u}(\tau_i) \\ &\quad = \begin{cases} 0, & q = h, \\ n^1_v(\tau_q, \tau_{q+1}) - n^2_v(\tau_q, \tau_{q+1}), & 1 \leq q < h. \end{cases} \\ &\quad = \begin{cases} 0, & q = h, \\ n^-_v(\tau_q, \tau_{q+1}), & 1 \leq q < h. \end{cases} \end{aligned} \tag{24}$$

It can be found that the equations (19)-(24) are the constraint of two independent single-commodity flow problem. We can easily checked that the solution that satisfies (10)-(14) must also make the equations (19)-(24) hold. However, the solutions for the constraint (19)-(24) may be not the ones for (10)-(14), which can be explained as follows. The fact $n^-_v(\tau_q, \tau_{q+1})$ in (24) could be any nonzero value can not ensure both $n^1_v(\tau_q, \tau_{q+1})$ and $n^2_v(\tau_q, \tau_{q+1})$ are positive in (14). Nevertheless, with the storage-transfer constraint for both the addition and subtraction flows as follows,

$$\begin{cases} n^+_v(\tau_q, \tau_{q+1}) = n^1_v(\tau_q, \tau_{q+1}) + n^2_v(\tau_q, \tau_{q+1}), \\ n^-_v(\tau_q, \tau_{q+1}) = n^1_v(\tau_q, \tau_{q+1}) - n^2_v(\tau_q, \tau_{q+1}), \end{cases} \tag{25}$$

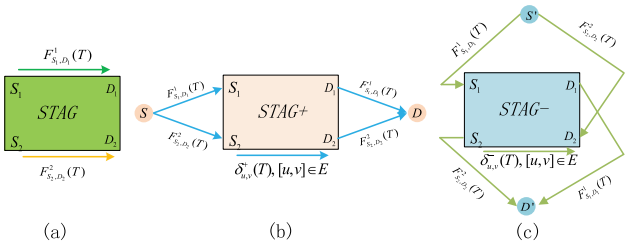


Fig. 4. Two-commodity maximum flow problem, and the corresponding addition and subtraction maximum flow problem.

we can derive that

$$\begin{cases} 2n_v^1(\tau_q,\tau_{q+1}) = n_v^+(\tau_q,\tau_{q+1}) + n_v^-(\tau_q,\tau_{q+1}) \geq 0, \\ 2n_v^2(\tau_q,\tau_{q+1}) = n_v^+(\tau_q,\tau_{q+1}) - n_v^-(\tau_q,\tau_{q+1}) \geq 0. \end{cases} \tag{26}$$

Thus, the terms $n_v^+(\tau_q,\tau_{q+1})$ and $n_v^-(\tau_q,\tau_{q+1})$ must satisfy an added node's constraint as

$$n_v^+(\tau_q,\tau_{q+1}) \geq |n_v^-(\tau_q,\tau_{q+1})|. \tag{27}$$

Hence, the solutions for the constraints (19)-(24) and (27) must be the ones for (10)-(14).

C. The Addition and Subtraction Maximum Flow Problem

Obviously, the two-commodity max-flow problem over an undirected DTN possesses multiple constraint. Moreover, there is no directional constraint of the feasible flow on each edge, and the coupling relationship of two commodity would complicate the two-commodity max-flow problem. The key idea behind the maximum two-commodity flow theorem is to utilize some mathematical operations to transform the capacity constraint of each edge, and to introduce the corresponding addition flow and subtraction flow to reduce the coupling relationship into independent one. Therefore, with the help of the constructed addition/subtraction flows, we can reduce one two-commodity maximum flow problem in Fig.4.(a) into one addition maximum flow problem in Fig.4.(b) and one subtraction maximum flow problem in Fig.4.(c).

In particular, we introduce a pair of virtual source-destination nodes $\{\mathbb{S},\mathbb{D}\}$ to construct the addition flow graph STAG$^+$ in Fig.4.(b), where the related edges $[\mathbb{S},\mathbb{S}_1]$, $[\mathbb{S},\mathbb{S}_2]$, $[\mathbb{D}_1,\mathbb{D}]$, and $[\mathbb{S}_2,\mathbb{D}]$ are added. From (19), the feasible flow of these added edges are $F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)$ or $F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)$. Similarly, in Fig.4.(c), the subtraction flow graph STAG$^-$ is designed through adding a pair of virtual source-destination nodes $\{\mathbb{S}',\mathbb{D}'\}$, where the edges and the feasible flow are marked with the capacity series defined in (20). Notice that the source and destination nodes in STAG$^+$ are different from that in STAG$^-$.

V. Dynamic Combined Flow Algorithm

In this section, based on the maximum two-commodity flow theorem, a STAG-based dynamic combined flow algorithm is proposed and described in detail to solve the two-commodity max-flow problem in DTNs.

A. Dynamic Combined Flow Algorithm in STAG

Even though the two-commodity max-flow problem can be reduced into two single-commodity flow problem, it is still complex for us to utilize the linear programming method to solve the two single-commodity flows problem with multiple constraints. In order to overcome this bottleneck, we will resort to the time-varying graph theory. On the basis of the maximum two-commodity flow theorem, we present a STAG-based dynamic combined flow algorithm, named as DCF algorithm, to obtain the maximum two-commodity flow in Algorithm 1.

As presented in Algorithm.1, firstly through introducing two pairs of the virtual source-destination nodes $\{\mathbb{S},\mathbb{D}\}$ and $\{\mathbb{S}',\mathbb{D}'\}$, the original STAG could be transformed into STAG$^+$ and STAG$^-$, whose details have been shown in Section IV-C. And then, with help of the bidirectional storage transfer series, in Section V-B, we will seek as many augmenting paths as possible to obtain the maximum network flow in both STAG$^+$ and STAG$^-$, where the link correlation between different time intervals is fully utilized here. Furthermore, in Section V-C, we will present an adjustment strategy to assure that the bidirectional storage transfer series in both STAG$^+$ and STAG$^-$ satisfy the node's constraint derived in (27) without changing $\delta^+_{u,v}(\tau_q)$ and $\delta^-_{u,v}(\tau_q)$ $(1 \leq q \leq h)$. Finally, we can combine the maximum flow of two single-commodity with (18) and (26), and derive the two-commodity max-flow ($F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)$ and $F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)$) as

$$\begin{cases} F^1_{\mathbb{S}_1,\mathbb{D}_1}(T) = \sum\limits_{q=1}^{h}\sum\limits_{v\in\mathcal{V}} f^1_{\mathbb{S}_1,v}(\tau_q) - \sum\limits_{q=1}^{h}\sum\limits_{v\in\mathcal{V}} f^1_{v,\mathbb{S}_1}(\tau_q), \\ F^2_{\mathbb{S}_2,\mathbb{D}_2}(T) = \sum\limits_{q=1}^{h}\sum\limits_{v\in\mathcal{V}} f^2_{\mathbb{S}_2,v}(\tau_q) - \sum\limits_{q=1}^{h}\sum\limits_{v\in\mathcal{V}} f^2_{v,\mathbb{S}_2}(\tau_q). \end{cases} \tag{28}$$

B. Computation of the Maximum Flow in STAG$^+$ and STAG$^-$

The problem of the single commodity maximum flow is to send as much flow as possible from the source to the destination node under the feasible flow conservation, and the capacity and storage-transfer constraints, which can be explicitly formulated as

$$f^M(T) = \max \sum_{l_i} f^{l_i}(T) = \max \sum_{l_i}\sum_{q=1}^{h} f^{l_i}(\tau_q), \tag{29}$$

where l_i is the ith augmenting path in the residual network rSTAG, $f^M(T)$ denotes the maximum flow of STAG, and $f^{l_i}(T) = (f^{l_i}(\tau_1),\ldots,f^{l_i}(\tau_q),\ldots,f^{l_i}(\tau_h))$ is the maximum feasible flow of l_i.

In fact, in our previous work [22], the *Max flow-STAG algorithm* has been proposed to solve the problem (29) through the following three steps: seeking an augmenting path from source to destination node, computing its maximum feasible flow, and getting its residual network. However, we can not directly apply the results in [22] here, which can be explained as follows. Some reverse path l^- may be taken as the augmenting path in STAG$^-$; during obtaining the maximum feasible of path l^-, the storage transfer series of all nodes along this

Algorithm 1 DCF Algorithm

1: **Input:** STAG = $\{(\mathcal{V},\ \mathcal{E}, T, C_{u,v}(T), N_v(T))|u,v \in \mathcal{V}, [u,v] \in E\}$, the source nodes $\mathbb{S}_1$ and $\mathbb{S}_2$ $(\mathbb{S}_1, \mathbb{S}_2 \in \mathcal{V})$, the destination nodes $\mathbb{D}_1$ and $\mathbb{D}_2$ $(\mathbb{D}_1, \mathbb{D}_2 \in \mathcal{V})$.
2: **Construct** the addition flow graph STAG$^+$ and the subtraction flow graph STAG$^-$.
3: **Compute** the maximum network flow $f^M_{\text{STAG}^+}(T)$ and $f^M_{\text{STAG}^-}(T)$, and obtain the feasible flow of each edge $\delta^+_{u,v}(\tau_q)$ and $\delta^-_{u,v}(\tau_q)$, $1 \le q \le h$, in STAG$^+$ and STAG$^-$, respectively.
4: **Adjust** the storage transfer series to meet the constraint $n^+_v(\tau_q, \tau_{q+1}) \ge |n^-_v(\tau_q, \tau_{q+1})|$, $v \in \mathcal{V}$ without changing $\delta^+_{u,v}(\tau_q)$ and $\delta^-_{u,v}(\tau_q)$, $\forall [u,v] \in \mathcal{E}$.
5: **Obtain** the maximum network flow of each commodity,
$$\begin{cases} f^1_{u,v}(\tau_q) = \frac{1}{2}\{\delta^+_{u,v}(\tau_q) + \delta^-_{u,v}(\tau_q)\}, \\ f^2_{u,v}(\tau_q) = \frac{1}{2}\{\delta^+_{u,v}(\tau_q) - \delta^-_{u,v}(\tau_q)\}, \end{cases}$$
and
$$\begin{cases} n^1_v(\tau_q, \tau_{q+1}) = \frac{1}{2}\{n^+_v(\tau_q, \tau_{q+1}) + n^-_v(\tau_q, \tau_{q+1})\}, \\ n^2_v(\tau_q, \tau_{q+1}) = \frac{1}{2}\{n^+_v(\tau_q, \tau_{q+1}) - n^-_v(\tau_q, \tau_{q+1})\}. \end{cases}$$
Then derive
$$\begin{cases} F^1_{\mathbb{S}_1,\mathbb{D}_1}(T) = \sum_{q=1}^{h} \sum_{v\in\mathcal{V}} f^1_{\mathbb{S}_1,v}(\tau_q) - \sum_{q=1}^{h} \sum_{v\in\mathcal{V}} f^1_{v,\mathbb{S}_1}(\tau_q), \\ F^2_{\mathbb{S}_2,\mathbb{D}_2}(T) = \sum_{q=1}^{h} \sum_{v\in\mathcal{V}} f^2_{\mathbb{S}_2,v}(\tau_q) - \sum_{q=1}^{h} \sum_{v\in\mathcal{V}} f^2_{v,\mathbb{S}_2}(\tau_q). \end{cases}$$
6: **Output:** the maximum flow of two commodities $F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)$ and $F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)$.

path may be a negative value, which is quite different from the case in STAG. Hence, we will modify the existing algorithms, especially for the STAG$^-$, in the following paragraphs. Furthermore, for clarity and completeness, we present the existing *Max flow-STAG algorithm* in Alg. 2.

Case 1: The Maximum Flow in STAG$^+$

It can be checked that both the source $\{\mathbb{S}_1, \mathbb{S}_2\}$ and the destination $\{\mathbb{D}_1, \mathbb{D}_2\}$ in STAG$^+$ are same with the ones in STAG. Thus, in STAG$^+$, all the selected augmenting paths would be the forward path l^+. Therefore, the *Max flow-STAG algorithm* in Algorithm 2 can be directly applied for STAG$^+$. Note that, the augment paths are orderly constructed from $\mathbb{S}_1$ to $\mathbb{D}_1$, and then from $\mathbb{S}_2$ to $\mathbb{D}_2$.

Case 2: The Maximum Flow in STAG$^-$

As shown in Fig. 4, the source and destination nodes in STAG$^-$ can be listed as $\{\mathbb{S}_1, \mathbb{D}_2\}$ and $\{\mathbb{D}_1, \mathbb{S}_2\}$, which are different from the ones in STAG$^+$. Hence, besides those forward paths l^+ from $\mathbb{S}_1$ to $\mathbb{D}_1$, there exists some other reverse paths l^- from $\mathbb{D}_2$ to $\mathbb{S}_2$ in STAG$^-$. Since the connection and the capacity for each edge are time-dependent in DTNs, the maximum flow of path for a multi-hops path $l^+(\mathbb{S} \to \mathbb{A} \to \mathbb{D})$ may be different from that of the reverse path $l^-(\mathbb{D} \to \mathbb{A} \to \mathbb{S})$ achieved from *Max flow-STAG algorithm*, even though the same edge has a same capacity time series. However, the real situation of the commodity in the network is always transmitted from the source to destination, and the reverse path l^- should have the same feasible flow as that in the forward path l^+, when the paths have same nodes and edge's capacity time series. Therefore, in order to obtain the maximum flow of STAG$^-$, we should modify the computation

Algorithm 2 Max Flow-STAG Algorithm

1: **Input:** STAG = $\{(\mathcal{V},\ \mathcal{E}, T, C_{u,v}(T), N_v(T))|u,v \in \mathcal{V}, [u,v] \in \mathcal{E}\}$, the source node $\mathbb{S}$ and the destination node $\mathbb{D}$ $(\mathbb{S}, \mathbb{D} \in \mathcal{V})$.
2: **Initialize** set both the maximum flow $f^M(T)$ and the storage transfer series $N_v(T)$ as 0, and the initial residual network rSTAG is STAG;
3: **Repeat** the following steps:
   1) **Adopt** the depth first search (DFS) method to acquire an augmenting path l_i from $\mathbb{S}$ to $\mathbb{D}$ in the residual network, where the storage strategy is considered;
   2) **Augment** flow $f^M(T)$ along l_i :
      - Compute the maximum flow $f^{l_i}(T)$ of path l_i and add it to $f^M(T)$ with (29);
      - Update the residual network and the storage transfer series of all nodes along the path l_i;
   3) **Return** $f^M(T)$.
4: **Until** there are no augmenting paths from $\mathbb{S}$ to $\mathbb{D}$ in the residual network.
5: **Output:** the single commodity maximum flow $f^M(T)$ from $\mathbb{S}$ to $\mathbb{D}$ for a given time period T.

rules of both the bidirectional storage transfer series and the reverse path l^- maximum flow to improve the existing STAG-based maximum flow algorithm.

1) Generation of the Bidirectional Storage Transfer Series: Let us consider one specific time period T, and assume that this period has been divided into the intervals $\tau_1, \tau_2, \ldots, \tau_h$. Then, along the forward path l^+, we can generate the storage transfer series of the node v, i.e., $N^+_v(T) = [n^+_v(\tau_1, \tau_2), \ldots, n^+_v(\tau_{q-1}, \tau_q), \ldots, n^+_v(\tau_{h-1}, \tau_h)]$, from the first interval to the last, as follow:

$$n^+_v(\tau_q, \tau_{q+1}) = \begin{cases} f^{l^+_{u,v}}(\tau_q) - f^{l^+_{v,w}}(\tau_q), \\ q = 1, \\ n^+_v(\tau_{q-1}, \tau_q) + f^{l^+_{u,v}}(\tau_{q-1}) - f^{l^+_{v,w}}(\tau_{q-1}), \\ q = 2, \ldots, h-1. \end{cases} \tag{30}$$

where the $n^+_v(\tau_q, \tau_{q+1})$ represents the amount of the transferred data from τ_q to τ_{q+1}; $f^{l^+_{u,v}}(\tau_q)$ and $f^{l^+_{v,w}}(\tau_q)$ separately denote the feasible flow into and out of node v along the path l^+, during the time interval τ_q.

However, with respect to the reverse path l^- in STAG$^-$, in order to get a equivalent maximum flow of the path, the storage transfer series should only be derived from the last interval but not the first interval as

$$'n^-_v(\tau_q, \tau_{q-1}) = \begin{cases} f^{l^-_{u,v}}(\tau_q) - f^{l^-_{v,w}}(\tau_q), \\ q = h \ge 2, \\ 'n^-_v(\tau_{q+1}, \tau_q) + f^{l^-_{u,v}}(\tau_q) - f^{l^-_{v,w}}(\tau_q), \\ q = h-1, \ldots, 2. \end{cases} \tag{31}$$

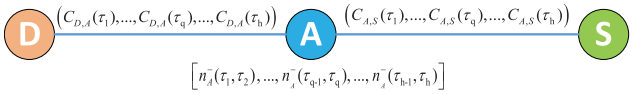


Fig. 5. The augmenting path l^- with two links.

where $f^{l^-_{u,v}}(\tau_q)$ $(f^{l^-_{v,w}}(\tau_q))$ is the feasible flow into (out of) the node v along the path l^- during the time interval τ_q, and $'n_v^-(\tau_q,\tau_{q-1})$ represents the amount of the transferred data from τ_q to τ_{q-1}. Moreover, from the definitions of $'n_v^-(\tau_q,\tau_{q-1})$ and $n_v^-(\tau_{q-1},\tau_q)$, it can be obtained

$$n_v^-(\tau_{q-1},\tau_q) = -'n_v^-(\tau_q,\tau_{q-1}), \quad q = 2,3,\ldots,h, \tag{32}$$

which means the storage transfer series of all nodes in the path l^- could be the negative value. Then, we will present a new method for compute the maximum flow of the path l^-.

2) Maximum Flow Computation of the Path l^-: After detecting the augmenting path l^-, we can compute its maximum flow $f^{l^-}(T)$ and the corresponding residual capacity of each edge. Then, the storage transfer series of all the nodes along the path l^- can be updated via the equation (31). The big principle behind the whole computation process is the capacity constraints of edges and the storage transfer series of nodes, which are utilized to derive the temporal feasible flow $tf^{l^-_{u,v}}(T)$ and feasible flow $f^{l^-_{u,v}}(T)$ of each edge $[u,v]$, $[u,v] \in l^-$, respectively.

For simplicity, we only examine the augmenting path with two edges, as shown in Fig. 5. Nevertheless, the computation method can be extended for the path with more edges. As marked in Fig. 5, the augment path l^- has three nodes $\mathbb{D}$, $\mathbb{A}$, and $\mathbb{S}$, where the given time T is divided into h time intervals, and the capacity time series for the edges $[\mathbb{D},\mathbb{A}]$ and $[\mathbb{A},\mathbb{S}]$ are $C_{\mathbb{D},\mathbb{A}}(T)$ and $C_{\mathbb{A},\mathbb{S}}(T)$, respectively. Moreover, the initial bidirectional storage transfer series for the node $\mathbb{A}$ is set as $N_{\mathbb{A}}^-(T)$.

To acquire the maximum flow of this path l^-, we compute the temporal feasible flow from its last edge to the first one. Note that this process would not change any original data. Hence, we define some duplicates $tC_{\mathbb{D},\mathbb{A}}(T) = C_{\mathbb{D},\mathbb{A}}(T)$, $tC_{\mathbb{A},\mathbb{S}}(T) = C_{\mathbb{A},\mathbb{S}}(T)$, and $tN_{\mathbb{A}}^-(T) = N_{\mathbb{A}}^-(T)$. With the capacity constraint (1), we can get the temporary feasible flow of edge $[\mathbb{A},\mathbb{S}]$ as $tf^{l^-_{\mathbb{A},\mathbb{S}}}(T) = tC_{\mathbb{A},\mathbb{S}}(T)$. However, the calculation of the temporary feasible flow for the edge $[\mathbb{D},\mathbb{A}]$ is more complex, because it is determined by $tC_{\mathbb{D},\mathbb{A}}(T)$, $tf^{l^-_{\mathbb{A},\mathbb{S}}}(T)$ and $tN_{\mathbb{A}}^-(T)$. Therefore, we need to traverse the entire time interval reversely. Specifically, for each interval $\tau_q, 1 \le q \le h$, the temporal feasible flow $tf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q)$ of edge $[\mathbb{D},\mathbb{A}]$ is

$$tf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q) = tsf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q) + trf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q), \tag{33}$$

where the temporal forward flow $tsf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q)$ represents the amount of the permitted data transferring from interval τ_q to its latter τ_k $(q < k \le h)$, and is defined as the sum of all temporal forward data td_{τ_q,τ_k}, i.e.

$$tsf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q) = \sum_{k=q+1}^{h} td_{\tau_q,\tau_k}, \tag{34}$$

$$td_{\tau_q,\tau_k} = \min\{tC_{\mathbb{D},\mathbb{A}}(\tau_q), \beta_{\tau_q,\tau_k}, tf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_k)\}, \tag{35}$$

$$\beta_{\tau_q,\tau_k} = \min\{|n_{\mathbb{A}}^-(\tau_q,\tau_{q+1})|,\cdots,|n_{\mathbb{A}}^-(\tau_{k+1},\tau_k)|\}. \tag{36}$$

Correspondingly, the temporal reciprocal flow $trf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q)$ in the equation (33) is the amount of the permitted data transferring from interval τ_q to its former τ_p $(1 \le p \le q)$ in node $\mathbb{A}$, and can be written as

$$trf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q) = \min\{tC_{\mathbb{D},\mathbb{A}}(\tau_q) - tsf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q), \delta\}, \tag{37}$$

$$\delta = \sum_{p=1}^{q} tf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_p) - \sum_{i=1}^{q-1} trf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_i), \tag{38}$$

$$tsf^{l^-_e}(\tau_j) = trf^{l^-_e}(\tau_j) = 0, \quad j \notin \{1,\ldots,h\}, e \in l^-. \tag{39}$$

Then, we can acquire the maximum flow of path l^- as $f^{l^-}(T) = tf^{l^-_{\mathbb{D},\mathbb{A}}}(T)$.

After obtaining $f^{l^-}(T)$, we need to get the residual capacity of each edge, which is tightly related to its feasible flow. Different from the processes of acquiring the temporal feasible flow, we will compute the feasible flow from the first edge to the last one along the path l^- and update the related network data. Firstly, the maximum flow $f^{l^-}(T)$ is set as the feasible flow of the first edge $[\mathbb{D},\mathbb{A}]$, $f^{l^-_{\mathbb{D},\mathbb{A}}}(T) = f^{l^-}(T)$, and its residual capacity is updated with (6) and (7). Then, we can obtain the feasible flow $f^{l^-_{\mathbb{A},\mathbb{S}}}(T)$, which is determined by $f^{l^-_{\mathbb{D},\mathbb{A}}}(T)$, $C_{\mathbb{A},\mathbb{S}}(T)$, and $N_{\mathbb{A}}^-(T)$. Similar to the computation of $tf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_q)$, we also traverse the entire time interval orderly to compute $f^{l^-_{\mathbb{A},\mathbb{S}}}(T)$. Notice that we will start from the first time interval τ_h while the former ones are begin at τ_1.

For each interval τ_q, $h \ge q \ge 1$, the feasible flow $f^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q)$ of the edge $[\mathbb{A},\mathbb{S}]$ is

$$f^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q) = sf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q) + rf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q), \tag{40}$$

where the forward flow $sf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q)$ represents the amount of the data transferred from the former interval τ_p to τ_q $(1 \le p < q)$, and can be defined by the sum of all forward data d_{τ_p,τ_q} as

$$sf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q) = \sum_{p=1}^{q} d_{\tau_p,\tau_q}, \tag{41}$$

$$d_{\tau_p,\tau_q} = \min\{f^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_p), \beta_{\tau_p,\tau_q}, C_{\mathbb{A},\mathbb{S}}(\tau_q)\}, \tag{42}$$

$$\beta_{\tau_p,\tau_q} = \min\{|n_{\mathbb{A}}^-(\tau_p,\tau_{p+1})|,\cdots,|n_{\mathbb{A}}^-(\tau_{q-1},\tau_q)|\}. \tag{43}$$

Similarly, the reciprocal flow $rf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q)$ in (40) is the amount of the data transferred from the latter interval τ_k to τ_q $(q \le k \le h)$ in node $\mathbb{A}$, and can be written as

$$rf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q) = \min\{C_{\mathbb{A},\mathbb{S}}(\tau_q) - sf^{l^-_{\mathbb{A},\mathbb{S}}}(\tau_q), \sigma\}. \tag{44}$$

$$\sigma = \sum_{k=p}^{h} f^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_k) - \sum_{i=p+1}^{h} sf^{l^-_{\mathbb{D},\mathbb{A}}}(\tau_i), \tag{45}$$

$$bf^{l^-_e}(\tau_j) = sf^{l^-_e}(\tau_j) = 0, \quad j \notin \{1,\ldots,h\}, e \in l^-. \tag{46}$$

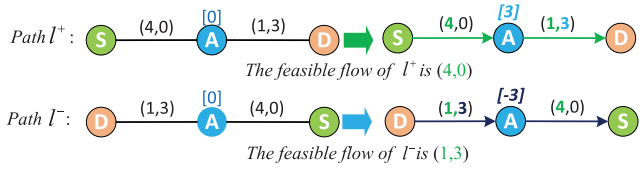


Fig. 6. Feasible flow of the path l^+ and l^-.

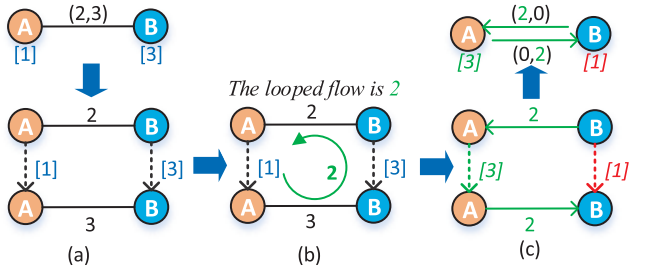


Fig. 7. Adjust bidirectional storage transfer series with the method of flow circle around single edge.

Finally, we can acquire the feasible flow $f^{l^-_{\mathbb{A},\mathbb{S}}}(T)$, and then derive the residual capacity of the edge $[\mathbb{A},\mathbb{S}]$ with (6) and (7). Moreover, the bidirectional storage transfer series $N^-_{\mathbb{A}}(T)$ can be updated with (31).

With respect to calculating the path maximum feasible flow, the largest difference between the path l^- and l^+ is the generation and utilization of the bidirectional storage time series. As shown in Fig.(6), there are two paths l^+ ($\mathbb{S} \rightarrow \mathbb{A} \rightarrow \mathbb{D}$) and $l^-(\mathbb{D} \rightarrow \mathbb{A} \rightarrow \mathbb{S})$. The maximum feasible flow of the path l^+ is (4,0), where based on the storage of node $\mathbb{A}$, 3 units data came from the source node $\mathbb{S}$ at time interval τ_1 can be transferred to the destination node $\mathbb{D}$ at τ_2, which is marked as $N^+_{\mathbb{A}}(T) = [3]$. For the path l^-, through introducing a negative storage time series, $N^-_{\mathbb{A}}(T) = [-3]$, we can obtain the maximum feasible flow as (1,3). Indeed, the path l^- and l^+ have a dual relation. Hence, the two paths have the same maximum flow.

C. Adjustment of the Bidirectional Storage Transfer Series

After calculating the maximum network flow $f^M_{\text{STAG}^+}(T)$ and $f^M_{\text{STAG}^-}(T)$, we can obtain the final feasible flows $\delta^+_{u,v}(\tau_q)$ and $\delta^-_{u,v}(\tau_q)$ for each edge in STAG$^+$ and STAG$^-$ as

$$\begin{cases} \delta^+_{u,v}(\tau_q) = \sum\limits_{l_i} f^{l_i(u,v)}(\tau_q), (u,v) \in l_i \in \text{STAG}^+, \\ \delta^-_{u,v}(\tau_q) = \sum\limits_{l'_i} f^{l'_i(u,v)}(\tau_q), (u,v) \in l'_i \in \text{STAG}^-. \end{cases} \tag{47}$$

Due to the different augmenting paths in STAG$^+$ and STAG$^-$, one specific node may have two distinct storage transfer series. Thus, it can not guarantee that all nodes satisfy the node's constraint in (27). To deal with this problem, we first introduce a *flow circle around single edge* (FCE) method to modify the bidirectional storage transfer series, but not to change the value of the obtained maximum network flow. Then, with FCE, we propose a adjustment strategy for the nodes in both STAG$^+$ and STAG$^-$ to satisfy the node's constraint.

As defined in section III, the element $n_v(\tau_{q-1},\tau_q)$ of $N_v(T)$ describes the amount of the transferred flow between the adjacent time intervals τ_{q-1} and τ_q, $1 < q \leq h$. Therefore, if we increase (or decrease) the flow into node v at τ_{q-1} or decrease (or increase) the flow out of node v at τ_q, the corresponding value of $n_v(\tau_{q-1},\tau_q)$ would also increase (or decrease). However, if the changed flow belong to different edges, it may affect the value of network flow. Hence, in order to avoid changing the network flow, the partial flow operation "FCE" is introduced, which can effectively adjust the value of the bidirectional storage transfer series. As shown in Fig.7.(a), the undirected edge $[\mathbb{A},\mathbb{B}]$ has a capacity time series $C_{\mathbb{A},\mathbb{B}}(2) = (2,3)$, and $n_{\mathbb{A}}(\tau_1,\tau_2) = [1]$ and $n_{\mathbb{B}}(\tau_1,\tau_2) = [3]$. Then, the time-aggregated edge $[\mathbb{A},\mathbb{B}]$ could be expanded along the time intervals as a partial graph. Through setting a counterclockwise (or clockwise) looped flow $\theta = min\{C_{\mathbb{A},\mathbb{B}}(\tau_1), C_{\mathbb{A},\mathbb{B}}(\tau_2)\} = min\{2,3\} = 2$ in Fig.7.(b), an adjusted edge flow and storage transfer series could be obtained in Fig.7.(c). It can be concluded that the FCE method is based on a single edge. This method can increase the value of $n_v(\tau_{q-1},\tau_q)$ by adding the flow into node v at τ_{q-1} and out of v at τ_q, and decrease its value on a contrary process, which can be seen from the operations of the nodes $\mathbb{A}$ and $\mathbb{B}$ in Fig.7.(c).

With FCE, we can realize the adjustment of the bidirectional storage transfer series in both STAG$^+$ and STAG$^-$. Furthermore, in order to satisfy the node's constraint, we propose a combined adjustment strategy, which contains the following steps:

- *Step 1:* Compare the bidirectional storage transfer series of all the nodes both in STAG$^+$ and STAG$^-$, find out those nodes that can not meet the node's constraint, and record them as the adjusting nodes.
- *Step 2:* For the adjusting nodes, on the basis of the FCE, increase the bidirectional storage transfer series $n^+_v(\tau_{q-1},\tau_q)$ in STAG$^+$, but decrease the $n^-_v(\tau_{q-1},\tau_q)$ in STAG$^-$.
- *Step 3:* Go back to the first step until all nodes satisfy the node's constraints.

VI. Algorithm Analysis and Example

In this section, we analyze the complexity of the proposed DCF algorithm and give an illustrative example to illustrate its application for the two-commodity max-flow problem in the DTN.

A. Analysis of Algorithm Complexity

Theorem 1: For given STAG with n nodes and m edges, the computational complexity of the DCF algorithm is $O((m+nh)|f^M(T)|)$, where T denotes the given time period, h is the number of the partitioned time instants in T, and $f^M(T)$ is the network maximum flow.

Proof: As presented in Algorithm 1, there are three main steps in DCF algorithm, which can be listed as follows. Step 1, construct and compute the maximum addition/subtraction flow in both STAG$^+$ or STAG$^-$; step 2, adjust the node's storage transfer series; step 3, combine the addition and subtraction

flow to obtain the two-commodity max-flow. For step 1, the computation complexity of each iteration mainly lies in seeking an augmenting path, computing the maximum flow for the searched augment path, and deriving the dynamic residual network, which can be concluded from Algorithm 2. With the depth first search method, the complexity for the augmenting path search can be written as $O(m+n)$ [32]. Furthermore, it can be checked that the complexity for the other two main operations can be separately denoted as $O((m+nh)$. Hence, the complexity to deal with the single augmenting path can be written as $O(m+n)+2*O((m+nh))=O((m+nh))$. If the flow value increases by at least one unit in each iteration, the number of the iterations in step 1 will be less than the maximum flow $f^M(T)$. Correspondingly, the maximum computational complexity of step 1 will be $O((m+nh)|f^M(T)|)$. With respect to step 2, the worst case is that all the nodes in STAG$^+$ or STAG$^-$ do not hold the node's constraint, and the maximum complexity is $O(nh)$. Finally, the complexity for step 3 reaches $O(mh)$. Hence, the total complexity of DCF algorithm can be listed as $2*O((m+nh)|f^M(T)|)+2*O(nh)+2*O(mh)\approx O((m+nh)|f^M(T)|)$. □

Indeed, the proposed DCF algorithm with the complexity $O((m+nh)|f^M(T)|)$ is pseudo-polynomial time algorithm, which is similar to the augmenting path algorithm proposed by Ford and Fulkerson [12]. Nevertheless, we can improve the augmenting path seek strategies to modify the DCF algorithm into the polynomial time one. Explicitly, we can resort to the methods for the static networks to reduce algorithm complexity. For example, Edmonds set the shortest path as the augmenting path [16], and Dinic constructed a "layered network", where multiple augmenting paths could be found for each depth first search [33]. Both methods can effectively made the Ford-Fulkerson method strongly polynomial [34]. Due to space limitation, the details of improving DCF algorithm complexity is omitted in this paper.

B. Application of DCF Algorithm

Suppose there are two different commodity flows in a predicted DTN with distinct sources and destinations. As shown in Fig.8(a), we can model the DTN as STAG, where the sources are $\mathbb{S}_1$, $\mathbb{S}_2$, the destinations are $\mathbb{D}_1$, $\mathbb{D}_2$. Furthermore, the given time horizon T is partitioned into 2 small time intervals, and the initial feasible flow is set as zero.

With the maximum two-commodity flow theorem, we can add two pairs of virtual source-destination nodes $\{\mathbb{S},\mathbb{D}\}$ and $\{\mathbb{S}',\mathbb{D}'\}$ into STAG. Correspondingly, the addition flow graph STAG$^+$ and the subtraction flow graph STAG$^-$ can be constructed in Fig.8(b1) and (b2), respectively. Then, with the algorithm.2 and the modified methods in subsection V-B, we can derive the maximum flow in both STAG$^+$ and STAG$^-$ as $f^M_{\text{STAG}^+}(2)=\{8,0\}$ in Fig.8(c1), and $f^M_{\text{STAG}^-}(2)=\{4,4\}$ in Fig.8(c2), respectively. Moreover, after comparing the bidirectional storage transfer series for all the nodes in both STAG$^+$ and STAG$^-$, all the adjusting nodes that could not meet node's constraint can be found as node $\mathbb{A}$ and $\mathbb{E}$. Next, with the adjustment strategy in subsection V-C, we can obtain the adjusted bidirectional storage transfer series for STAG$^+$ in Fig.8(d1), and that for STAG$^-$ in Fig.8(d2). Notice that the maximum flow in Fig.8(d1) and Fig.8(d2) are same with that in Fig.8(c1) and Fig.8(c2), which is unexpected and can be explained by the core ideas of our proposed adjustment strategy. For clear presentation, we take the node $\mathbb{A}$ for an example to introduce the adjusting process. Through utilizing the flow loop around the edge $[\mathbb{A},\mathbb{B}]$ in STAG$^+$, the value of $n^+_{\mathbb{A}}(\tau_1,\tau_2)$ can be increased from [1] to [2]. Similarly, with the edge $[\mathbb{A},\mathbb{C}]$ in STAG$^-$, the value of $n^-_{\mathbb{A}}(\tau_1,\tau_2)$ can be decreased from [3] to [2].

Finally, with (18) and (26), the maximum flow in STAG$^+$ and STAG$^-$ can be combined. Then, we can obtain the maximum two-commodity flow in Fig.8(e), where the flow of commodity-1 and commodity-2 are $F^1_{\mathbb{S}_1,\mathbb{D}_1}(2)=\{4,0\}$ and $F^2_{\mathbb{S}_2,\mathbb{D}_2}(2)=\{4,0\}$, respectively. Besides, as shown in Fig.8(e), from the results of the two-commodity max-flow in STAG, we can get not only a routing scheme for the two commodities with maximum flow, but also the occupancy of edge capacity and node storage, which is beneficial to fully utilize the scarcity network resources.

VII. Performance Evaluation

In this section, numerical results and analysis are presented to demonstrate the performance of our proposed algorithm.

A. Simulation Setup

We conduct our simulations over a low orbital satellite network, which is a typically predicted DTN scenario. In this satellite network, there are 6 orbital planes, and 1-4 satellites are fixed on each orbital plane. All satellites are arbitrarily selected from the Iridium constellation [35], and are fixed at a approximated height of 780 km and inclination of $86.4°$. Moreover, we utilize the Satellite Tool Kit (STK) simulator to generate the contact topology. As an analytical tool, STK can perform the complex analysis of land, sea, air and space assets in an integrated solution. Besides, we assume that there is no limitation on the node storage size. The time interval is set as 1 minute, and the capacity of each contact is randomly chosen from 20, 50, and 100 Mbps.

The given time period T and the size of the network topology are treated as two independent factors in our system model. Here, we will conduct two different simulation cases: in the first scenario, there are 6 orbital planes, 2 satellites are fixed on each plane, and T varies from 0 to 220 minutes; in the second case, the network contains 6 orbital planes, and from 0 to 4 satellites are placed on each plane, and the time period T is fixed as 90 minutes which nearly is the motion cycle of Iridium constellation. The performance of our proposed algorithm is evaluated in terms of the following metrics:

- *Network Maximum Flow* : the maximum flow of the two commodities from two distinct sources to their corresponding destinations, i.e., $\max\{F^1_{\mathbb{S}_1,\mathbb{D}_1}(T)+F^2_{\mathbb{S}_2,\mathbb{D}_2}(T)\}$.
- *Link Utilization Ratio* γ: the ratio of the total flow of all links to the whole network capacity, i.e., $\gamma=\sum_{u,v\in\mathcal{V}}(|f^1_{u,v}(T)|+(|f^2_{u,v}(T)|)/\sum_{u,v\in\mathcal{V}}C_{u,v}(T)$.

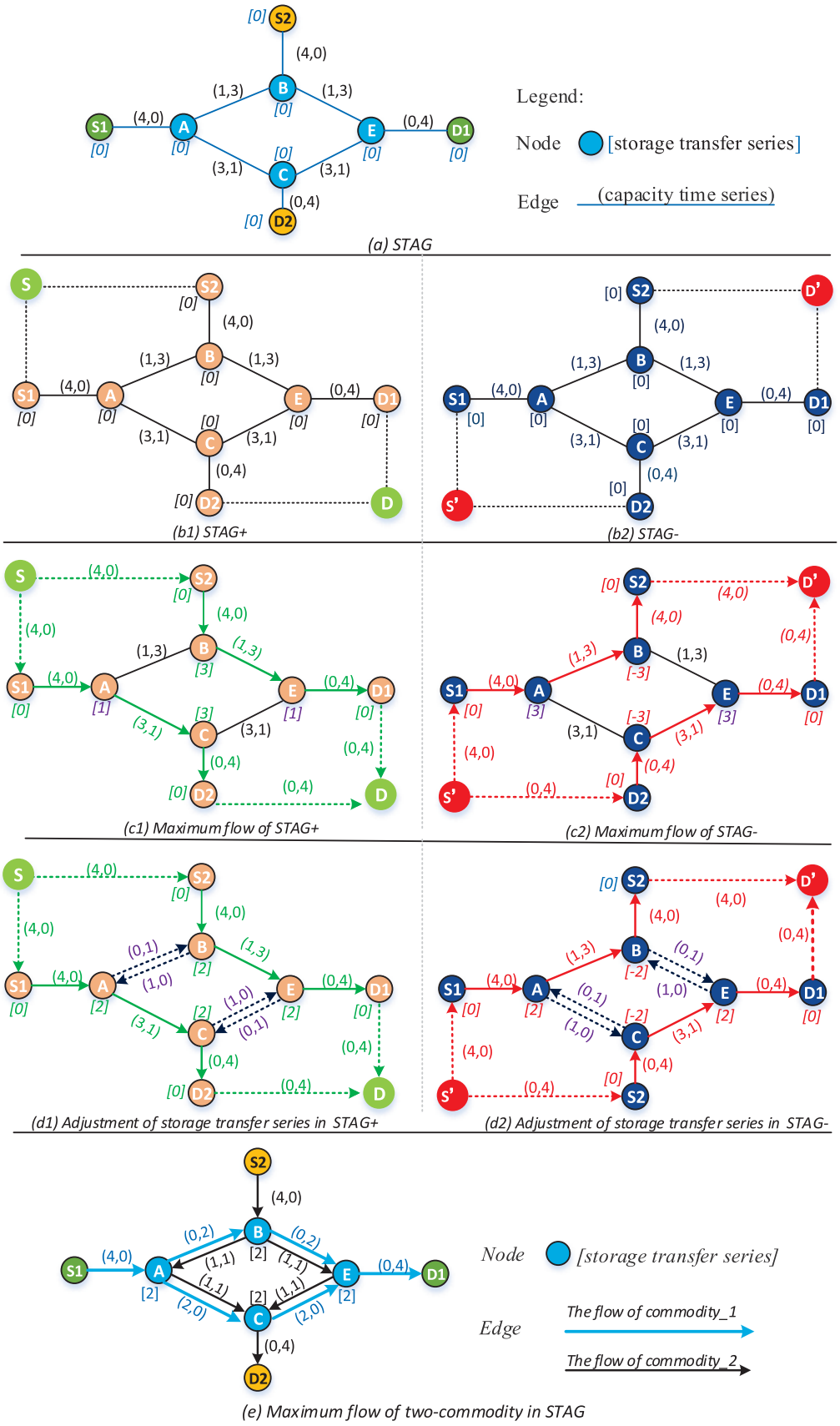


Fig. 8. Example of the dynamic combined algorithm for Two-commodity max-flow problem in a DTN.

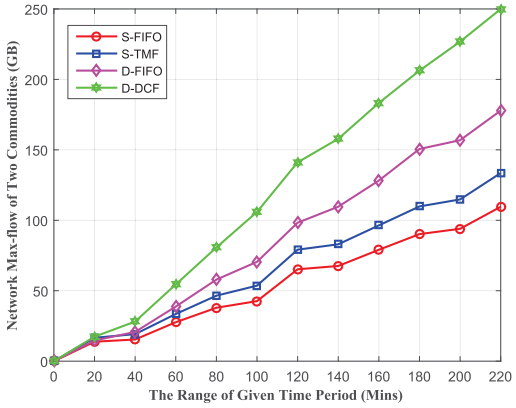


Fig. 9. The curves of the maximum flow versus the given time period T for 12 satellites.

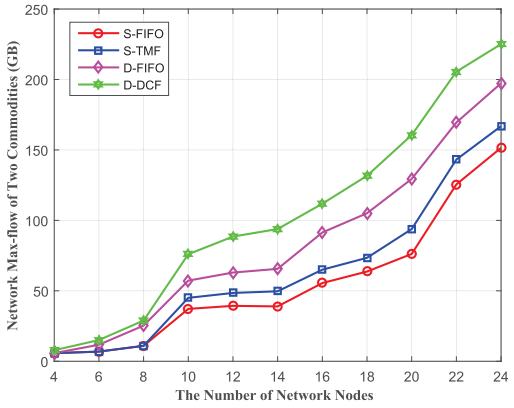


Fig. 10. The curves of the maximum flow versus the number of the network nodes with given time period $T = 90$ minutes.

- *Maximum Node Cache* α: the maximum stored data of all nodes, i.e., the largest element in all nodes storage transfer series $\alpha = \max\{n_v(\tau_{q-1}, \tau_q)|1 < q \leq h, \forall v \in \mathcal{V}\}$.
- *Average Node Cache* β: the average stored data of all nodes $\beta = \sum_{v \in \mathcal{V}} \max\{n_v(\tau_{q-1}, \tau_q)|1 < q \leq h\}/N$, where N is the size of $\mathcal{V}$.

In our simulations, we arbitrarily set the sources and destinations, and each metric is averaged among 10 simulation runs. All simulations are executed on a Hewlett-Packard Z640 Tower workstation (Intel Xeon E5-2620 v3 2.40GHz, 64GB RAM, O.S. Windows 7 Professional 64bits).

B. Simulation Results and Discuss

To evaluate the proposed DCF algorithm (marked as D-DCF), we compare it with three existing methods, which are listed as: the two-commodity maximum flow algorithm for the static network (S-TMF) in [14], the method (S-FIFO) based on both the commodity priority strategy and the static Ford-Fulkerson algorithm in [12], and the method (D-FIFO) based on both the STAG-based single commodity maximum flow algorithm in [22] and the commodity priority strategy. Specifically, for the S-FIFO and the D-FIFO, in order to obtain the two-commodity maximum flow, we apply the corresponding single commodity maximum flow algorithm for the higher-priority commodity and then for the another one.

Fig. 9 and Fig. 10 present the curves of the network maximum flow with respect to different given time periods and the number of the nodes in the network, respectively. It can be seen that with the increase of the given time period or the network size, the amounts of the network maximum flow obtained from all methods gradually rise, which is attributed to the fact that the more contacts can be generated with the increasing of the network size. Then, it can be checked that both the D-DCF and the D-FIFO have a larger network flow than that from S-TMF and S-FIFO, which is not unexpected and can be explained as follows. The process of the data storing-carrying are considered at each node of STAG, and the connectivity of the adjacent contacts would be enhanced. Moreover, the curve of D-DCF is higher than that of any

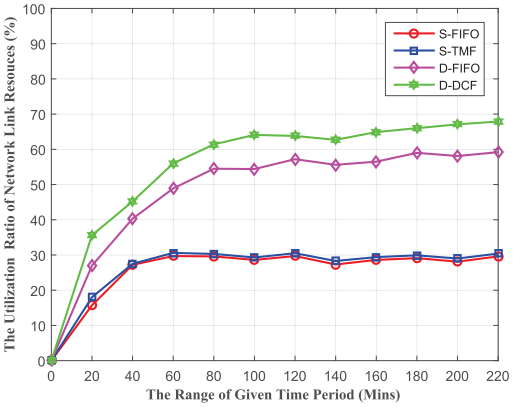


Fig. 11. The curves of the link utilization ratio versus the given time period T for 12 satellites.

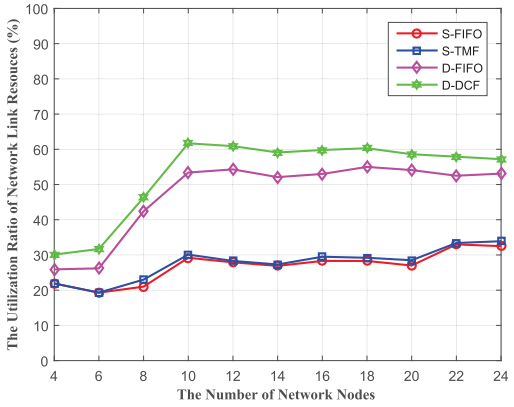


Fig. 12. The curves of the link utilization ratio versus the number of the network nodes with given time period $T = 90$ minutes.

another methods, which means our proposed algorithm can achieve the best performance in terms of the network flow.

With respect to the link utilization ratio γ, it can be seen from Fig. 11 that the curves of all methods increase quickly before 90 minutes and then tend to be stead. This phenomena indicates that, for the satellite network with fixed number of satellites, the link utilization ratio becomes one constant value if the given time period is much bigger than the satellite

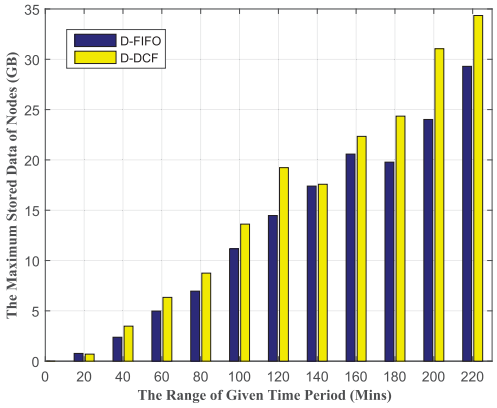


Fig. 13. The maximum sizes of the stored data at nodes for 12 satellites with different time periods.

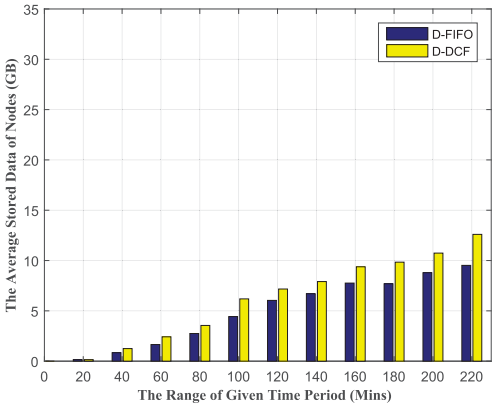


Fig. 15. The average sizes of the stored data for 12 satellites with variant time periods.

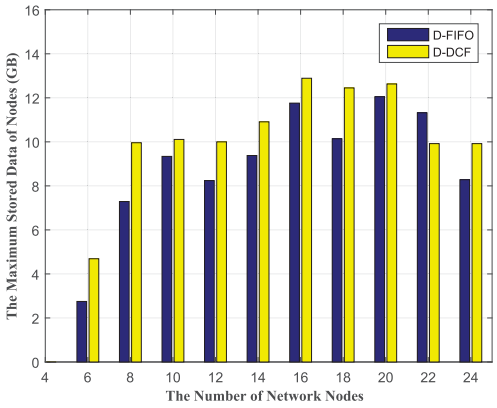


Fig. 14. The maximum sizes of the stored data during 90 minutes with different network sizes.

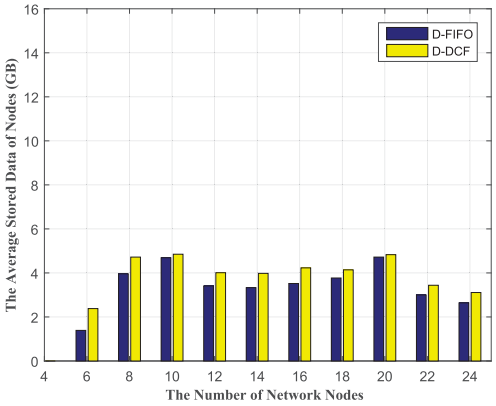


Fig. 16. The average sizes of the stored data during 90 minutes with different network sizes.

motion cycle time. Meanwhile, our proposed algorithms can also achieve higher link utilization ratio than S-FIFO, S-TMF, and D-FIFO methods. Similar results are shown in Fig.12. However, the value of γ has a greatly fluctuation in the region of the small node numbers, which is caused by the rapidly topology change in the networks of the small sizes. Moreover, both the D-DCF and D-FIFO methods can achieve higher γ than S-TMF and S-FIFO, which means the contacts of network can be fully exploited with the help of the node storage.

In order to illustrate the maximum storage occupancy of our proposed D-DCF algorithm, we compare it with the the D-FIFO method. Fig. 13 and Fig. 14 separately show the impact of the given time period and the network nodes' number on the maximum node cache. Fig. 13 shows that the maximum amounts of the stored data at nodes are closely related to the given time periods, which can be explained by the following fact. The links are intermittent connectivity over the constructed satellite network, and a larger time period would cause more data to be stored and carried at the nodes. Hence, the nodes will consume more storage source. However, it can be found from Fig. 14 that the maximum sizes of the stored data at nodes do not change drastically and keep in one relatively stable level with the increase of the nodes' numbers. This observation can be accounted by the fact that more links can be generated with the increasing nodes' numbers, and the nodes will have multiple choice to forward data at the same time. In addition, compared with D-FIFO, our proposed D-DCF algorithm can achieve much more network flow at the cost of a little more storage at some nodes.

Finally, to evaluate the average storage occupancy of all the nodes, Fig. 15 and Fig. 16 are presented to show the average node cache for the D-DCF and D-FIFO method. It is clear from Fig.15 that the average size of the stored data at the nodes gradually raises with the increasing of the given time period, which further confirms the impact of the given time periods on network storage. Besides, as shown in Fig. 16, the amount of average stored data changes slightly with the increasing of the nodes' number. Thus, with a given time period, the change of network topology size does not increase the requirements of node cache. Furthermore, with given storage resources, the D-DCF can achieve higher network throughput than the S-FIFO, which means that D-DCF algorithms can achieve better storage utilization than D-FIFO.

VIII. Conclusions and Future Work

In this paper, we studied the problem of two-commodity max-flow in DTN networks with the help of an undirected time-varying graph. Through analysing the relationship of the two commodities, the maximum two-commodity flow

theorem could simplify the coupling two-commodity flow problem into two single-commodity flow ones. With the STAG, we constructed a pair of flow graphs (addition flow graph and subtraction flow graph) to describe the reduced two single-commodity flows (addition flow and subtraction flow). To reduce computational complexity, a STAG-based dynamic combined flow algorithm was proposed and described in detail to maximize two-commodity flow in DTNs. Finally, an illustrative example and simulations were presented to demonstrate the effectiveness of the proposed algorithm. In our future work, we will apply the constructed STAG and STAG-based maximum flow algorithms for more practical application scenarios.

REFERENCES

[1] J. Fang, C. Su, Z. Chen, and P. Lund, "Power system structural vulnerability assessment based on an improved maximum flow approach," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 777–785, Mar. 2018.

[2] C. Karsten, D. Pisinger, S. Ropke, and B. Brouer, "The time constrained multi-commodity network flow problem and its application to liner shipping network design," *Transport. Res. E-Log.*, vol. 76, pp. 122–138, Apr. 2015.

[3] M. A. Mollah, X. Yuan, S. Pakin, and M. Lang, "Rapid calculation of max-min fair rates for multi-commodity flows in fat-tree networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 156–168, Jan. 2018.

[4] M. Darayi, K. Barker, and J. R. Santos, "Component importance measures for multi-industry vulnerability of a freight transportation network," *Netw. Spatial Econ.*, vol. 17, no. 4, pp. 200–214, Oct. 2017.

[5] L. Yan, H. Shen, and K. Chen, "TSearch: Target-oriented low-delay node searching in DTNs with social network properties," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3841–3855, Dec. 2016.

[6] P. Sommer, B. Kusy, P. Valencia, R. Dungavell, and R. Jurdak, "Delay-tolerant networking for long-term animal tracking," *IEEE Internet Comput.*, vol. 22, no. 1, pp. 62–72, Jan. 2018.

[7] T. Zhang, H. Li, S. Zhang, and J. Li, "A storage-time-aggregated graph-based QoS support routing strategy for satellite networks," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.

[8] D. Raj, M. V. Ramesh, and S. Duttagupta, "Delay tolerant routing protocol for heterogeneous marine vehicular mobile ad-hoc network," in *Proc. IEEE ICPCCW*, Mar. 2017, pp. 461–466.

[9] Y. Li, P. Hui, D. Jin, and S. Chen, "Delay-tolerant network protocol testing and evaluation," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 258–266, Jan. 2015.

[10] X. Zhang, C. Moore, and M. E. J. Newman, "Random graph models for dynamic networks," *Eur. Phys. J. B*, vol. 90, no. 10, pp. 1111–1136, Dec. 2017.

[11] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in temporal networks" in *Proc. ACM WSDM*, Feb. 2017, pp. 601–610.

[12] L. Ford, Jr., and D. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton Univ. Press, 1962.

[13] J. L. R. Ford and D. R. Fulkerson, "Constructing maximal dynamic flows from static flows," *Oper. Res.*, vol. 6, no. 3, pp. 419–433, 1958.

[14] T. C. Hu, "Multi-commodity network flows," *Oper. Res.*, vol. 11, no. 3, pp. 344–360, 1963.

[15] A. Itai, "Two-commodity flow," *J. ACM*, vol. 25, no. 4, pp. 596–611, 1978.

[16] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, Dec. 1987.

[17] R. K. Ahuja and J. B. Orlin, "A capacity scaling algorithm for the constrained maximum flow problem," *Networks*, vol. 25, no. 2, pp. 89–98, Mar. 1995.

[18] A. Hall, S. Hippler, and M. Skutella, "Multicommodity flows over time: Efficient algorithms and complexity," *Theor. Comput. Sci.*, vol. 379, no. 3, pp. 387–404, Jun. 2007.

[19] M. Gro and M. Skutella, "Maximum multicommodity flows over time without intermediate storage," in *Proc. Algorithm-ESA*. Berlin, Germany: Springer, Sep. 2012, pp. 539–550.

[20] G. Borradaile, P. N. Klein, S. Mozes, Y. Nussbaum, and C. Wulff-Nilsen, "Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time," *SIAM J. Comput.*, vol. 46, no. 4, pp. 1280–1303, Aug. 2017.

[21] G. Iosifidis, I. Koutsopoulos, and G. Smaragdakis, "Distributed storage control algorithms for dynamic networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1359–1372, Jun. 2017.

[22] H. Li, T. Zhang, Y. Zhang, K. Wang, and J. Li, "A maximum flow algorithm based on storage time aggregated graph for delay-tolerant networks," *Ad Hoc Netw.*, vol. 59, pp. 63–70, May 2017.

[23] M. Huang, S. Chen, Y. Zhu, and Y. Wang, "Topology control for time evolving and predictable delay-tolerant networks," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2308–2321, Nov. 2013.

[24] E. Kohler, K. Langkau, and M. Skutellaet, "Time-expanded graphs for flow-dependent transit times," in *Proc. 10th Annu. Eur. Symp. Algorithms*, Sep. 2002, pp. 599–611.

[25] G. Araniti *et al.*, "Contact graph routing in DTN space networks: Overview, enhancements and performance," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 38–46, Mar. 2015.

[26] R. B. R. Loureno, G. B. Figueiredo, M. Tornatore, and B. Mukherjee, "Post-disaster data evacuation from isolated data centers through leo satellite networks," in *Proc. IEEE ICC*, May 2017, pp. 1–5.

[27] X. Jia, T. Lv, F. He, and H. Huang, "Collaborative data downloading by using inter-satellite links in leo satellite networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1523–1532, Mar. 2017.

[28] B. George and S. Kim, "Spatio-temporal network databases and routing algorithms: A summary of results," in *Proc. SSTD*, Jul. 2007, pp. 460–477.

[29] M. Sheng, Y. Wang, J. Li, R. Liu, D. Zhou, and L. He, "Toward a flexible and reconfigurable broadband satellite network: Resource management architecture and strategies," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 127–133, Aug. 2017.

[30] C. Glacet, M. Fiore, and M. Gramaglia, "Temporal connectivity of vehicular networks: The power of store-carry-and-forward," in *Proc. IEEE VNC*, Dec. 2015, pp. 52–59.

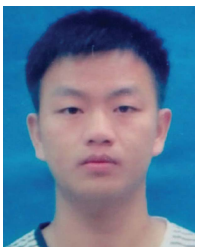
[31] I. Fragkos, J. F. Cordeau, and R. Jans, "The multi-period multicommodity network design problem," CIRRELT, Montreal, QC, Canada, Tech. Rep. CIRRELT 2017-63, Oct. 2017.

[32] K. A. Ravindra, L. M. Thomas, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. New York, NY, USA: Pearson, 1993.

[33] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," *Soviet Math. Doklady*, vol. 11, no. 5, pp. 1277–1280, 1970.

[34] A. V. Goldberg and R. E. Tarjan, "Efficient maximum flow algorithms," *Commun. ACM*, vol. 57, no. 8, pp. 82–89, Aug. 2014.

[35] C. E. Fossa, R. A. Raines, G. H. Gunsch, and M. A. Temple, "An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system," in *Proc. IEEE NAEC*, Jul. 1998, pp. 152–159.



Tao Zhang received the B.Eng. degree in telecommunication engineering from the HeFei University of Technology, Hefei, China, in 2015. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Service Networks, Institute of Information and Science, Xidian University, Xi'an. He is also a Visiting Ph.D. Student with the Department of Computer Science, University of Virginia, from 2017 to 2019. His research interests include satellite networks, time-varying graph theory, routing, and resource allocation.

Hongyan Li (M'08) received the M.S. degree in control engineering from Xi'an Jiaotong University, Xi'an, China, in 1991, and the Ph.D. degree in signal and information processing from Xidian University, Xi'an, in 2000. She is currently a Professor with the State Key Laboratory of Integrated Service Networks, Xidian University. Her research interests include spatial information networks, cognitive networks, integration of heterogeneous networks, and mobile ad hoc networks.

Jiandong Li (SM'05) received the B.E., M.S., and Ph.D. degrees in communications engineering from Xidian University, Xi'an, China, in 1982, 1985, and 1991, respectively. He has been a faculty member with the School of Telecommunications Engineering, Xidian University, since 1985, where he is currently a Professor and the Vice Director of the Academic Committee of the State Key Laboratory of Integrated Service Networks. He was a Visiting Professor with the Department of Electrical and Computer Engineering, Cornell University, from 2002 to 2003. His major research interests include wireless communication theory, cognitive radio, and signal processing. He was a recipient of the Distinguished Young Researcher from NSFC and the Changjiang Scholar from the Ministry of Education, China. He has served as the General Vice Chair for ChinaCom 2009 and the TPC Chair for the IEEE ICCC 2013.

Shun Zhang received the B.S. degree in communication engineering from Shandong University, Jinan, China, in 2007, and the Ph.D. degree in communications and signal processing from Xidian University, Xi'an, China, in 2013. He is currently with the State Key Laboratory of Integrated Services Networks, Xidian University. His research interests include MIMO-OFDM systems, relay networks, and detection and parameter estimation theory.

Haiying Shen (M'06–SM'13) received the B.S. degree in computer science and engineering from Tongji University, China, in 2000, and the M.S. and Ph.D. degrees in computer engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor with the Department of Computer Science, University of Virginia. Her research interests include distributed computer systems, cloud computing, big data, and cyber-physical systems. She is a Microsoft Faculty Fellow of 2010 and a Senior Member of the ACM.