Ad Hoc Networks 59 (2017) 63–70

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A maximum flow algorithm based on storage time aggregated graph for delay-tolerant networks

Hongyan Li[a], Tao Zhang[a,*], Yangkun Zhang[b], Kan Wang[a], Jiandong Li[a]

[a] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[b] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 1 May 2016
Revised 7 December 2016
Accepted 29 January 2017
Available online 31 January 2017

Keywords:
Delay-tolerant networks
Time aggregated graph
Maximum flow

ABSTRACT

Delay-tolerant networks (DTNs) (e.g., Internet, satellite networks, sensor networks, ad hoc networks) have attracted considerable attentions in both academia and industry. As a fundamental problem, the maximum flow is of vital importance for routing and service scheduling in networks. For solving the maximum flow problem of the DTN, an appropriate model should be built first. Compared to the conventional snapshot approach to model the DTN topology, the time aggregated graph (TAG) is capable of accurately characterizing the intermittent connectivity and time-varying capacity for each edge, and thus has been acted as a suitable model for modeling DTNs. However, existing TAG-related works only focus on solving the shortest path problem, and neither the correlation between time intervals nor nodes storage of a DTN are described in TAG, resulting in a non-trivial maximum flow problem in TAG. In this paper, we study the maximum flow problem through our proposed storage time aggregated graph (STAG) for DTNs. First, an intermediate quantity named bidirectional storage transfer series is introduced to each node in STAG, and the corresponding transfer rule is also designed for this series to model the correlation between time intervals. Next, on the basis of the storage transfer series, a STAG-based algorithm is proposed and described in detail to maximize the network flow. In addition, we analyze the effectiveness of the proposed algorithm by giving an illustrative example.

1. Introduction

Recently, Delay-Tolerant Networks (DTNs) [1] have drawn much research attentions due to its wide application in Internet [2], satellite networks [3,4], ad hoc networks [5], sensor networks [6], Internet of Things (IOT) [7] and many other communication networks [8,9]. As a fundamental problem, the maximum flow is of vital importance for routing and service scheduling in networks. Especially for DTN networks, there exists no permanent end-to-end path since the topology and links' characteristics are time-varying. The study on the maximum flow problem could not only adapt to the network dynamics and ensure reliable transmissions, but also provide powerful guarantee for network management (e.g., network planning and optimization), thus playing a significant role in DTN.

In particular, the graph theory has been viewed as an efficient approach to study DTNs in many existing works [10–18]. In graph theory terminology, snapshots are utilized to model the DTN [11], where each snapshot corresponds to the topology of a DTN at a particular time interval. However, there exists no correlation between snapshots, which can not be utilized to solve the dynamic maximum flow problem. What is more, it would result in a prohibitively large number of snapshots with the time increasing. Time expanded graphs [12,13], which have been used to model dynamic networks (e.g., DTNs), employ replication of networks across time intervals, also resulting in high storage overhead and computationally complex algorithms. In contrast, time aggregated graphs (TAG), which have been proposed by Betsy George et al. [15], allow the properties of edges to be modeled as a time series. Since the model does not need to replicate the entire graph for each time interval, it uses less memory and the algorithms for common operation are computationally more efficient than those for time expanded graph.

Indeed, time expanded graphs are essentially an expansion of static graphs, and hence many standard flow maximization algorithms (e.g., generic augmenting path algorithm [10]) can be applied to time expanded graphs. In particular, Iosifidis et al. in [17] iteratively updated the minimum cut of the time expanded graph and derived a joint storage capacity management to max-

* Corresponding author.
E-mail addresses: hyli@xidian.edu.cn (H. Li), taozhangfsz@gmail.com (T. Zhang), hubert.zyk@gmail.com (Y. Zhang), kanwangkw@outlook.com (K. Wang), jdli@xidian.edu.cn (J. Li).

http://dx.doi.org/10.1016/j.adhoc.2017.01.006
1570-8705/

imize the amount of data transferred to the destination. Nevertheless, the time expanded graphs connect any two time intervals of the same node via one link and the storage size of this node can be deemed as the link's capacity, while our proposed TAG approach strives to construct the storage of the node as a time series, namely, bidirectional storage transfer series. Moreover, the augmenting path algorithm only considers single capacity of an edge both in time expanded graphs and traditional static graphs, and thus can not be easily applied to TAG where an edge has a capacity series. Furthermore, existing TAG-related works only focus on solving the shortest path problem [14,15], and neither the correlation between time intervals nor nodes storage of a DTN are described in TAG, leading to the sub-optimal rather than the optimal solution for the maximum flow problem in a DTN. To the best of our knowledge, the maximum flow problem for a DTN through TAG has not been studied in previous works.

In this paper, we modify the existing TAG as STAG and then exploit it to study the maximum flow problem of a DTN. In order to ameliorate the model, we introduce an intermediate quantity named *bidirectional storage transfer series* to each node in STAG, and the corresponding transfer rule is also designed for this series to model the correlation between time intervals. As such, it is possible for us to solve the maximum flow problem of a DTN. Furthermore, on the basis of the bidirectional storage transfer series, a STAG-based algorithm is proposed and described in detail to effectively solve the maximum flow problem.

The distinctive features of this paper are summarized as follows:

- We model the DTN by STAG (a modified TAG) where the time-variant topology (capacity) of a DTN is incorporated. Meanwhile, the bidirectional storage transfer series is introduced to each node in STAG, which could describe the data storage process of each node and the correlation between time intervals of each edge.
- A transfer rule is formulated to describe bidirectional storage transfer series. It could act as a storage strategy and incorporates two new storage functions, namely forward storage function and reciprocal storage function. Accordingly, some storage transfer series-based definitions are also presented.
- To solve the DTN's maximum flow problem with high computational complexity, we propose a *Max flow-STAG* algorithm on the basis of STAG. The proposed algorithm can obtain the DTN's maximum flow by giving a feasible routing scheme.
- The theoretical analysis is presented to validate the proposed algorithm. We analyze the effectiveness of the algorithm by giving an illustrative example.

The remainder of this paper is organized as follows. In Section 2, the system model is presented. Following it, Section 3 provides some basic definitions in STAG. In Section 4, we propose and describe the STAG-based maximum flow algorithm in detail. In Section 5, we provide the validness of the proposed algorithm. Finally, we conclude this work and discuss the future works in Section 6.

2. System model

In this section, the system model for a DTN with some predictable characteristics is presented. For the sake of presentation and without loss of generality, as in [19], we neglect both the transmission delay and the propagation delay on the edge.

Consider a DTN with predictable characteristics [20], e.g., the motion period, the topology structure and the capacity of each edge. Assume a large time period $T = [t_0, t_h)$, V and E are the set of nodes and edges, respectively. Edge $e \in E$ is a directed edge between two nodes in V, and has a constant capacity on a small time scale to carry commodity flow. We further partition T into a set of h small time intervals $\tau_1, \ldots, \tau_q, \ldots, \tau_h$. To guarantee the time continuity, each time interval is set as left closed and right open, namely, $\tau_q = [t_{q-1}, t_q)$.

It follows that the DTN can be modeled by a graph as STAG $= \{(V, E, T, C_T^{u,v}, N_T^v) | u \in V, v \in V, (u, v) \in E\}$, where

- V : the set of nodes.
- E : the set of edges.
- T : the given time period.
- $C_T^{u,v}$: a capacity time series of an edge (u, v). $C_T^{u,v} = (c_{\tau_1}^{u,v}, \ldots, c_{\tau_q}^{u,v}, \ldots, c_{\tau_h}^{u,v})$ and $c_{\tau_q}^{u,v} = \int_{\tau_q} w_{u,v}(t)dt$, where $w_{u,v}(t)$ is the capacity of the edge (u, v) at a time instant $t \in \tau_q$. Thus, $c_{\tau_q}^{u,v}$ is the total capacity during the time interval $\tau_q = [t_{q-1}, t_q)$.
- N_T^v : a bidirectional storage transfer series of node v. $N_T^v = [n_{\tau_1,\tau_2}^v, \ldots, n_{\tau_{q-1},\tau_q}^v, \ldots, n_{\tau_{h-1},\tau_h}^v]$, where n_{τ_{q-1},τ_q}^v is the amount of flow transferred between the adjacent time interval τ_{q-1} and τ_q. Note here that, the original values of N_T^v for all nodes are initialized to zero.

In practical terms, DTNs may arise in networks with known connectivity terms such as Low-Earth Orbiting Satellites (LEO) [20], and thus STAG is proposed for modeling predictable DTNs. We make some rationality hypothesis and create a general example, as shown in Fig. 1, where each edge possess a capacity time series $C_T^{u,v}$ to aggregate its capacity at different time intervals in the order of time. For instance, the capacity time series of edge (S, A) is $C_5^{S,A} = (2, 0, 3, 0, 0)$, where the number 2 represents the total capacity of edge (S, A) at time interval τ_1. Each node is marked with a bidirectional storage transfer series N_T^v, and the original values are initialized as $N_T^v = [0, 0, 0, 0]$.

We modify the existing TAG as STAG, and an intermediate quantity named *bidirectional storage transfer series* is designed for modeling the storage process. As such, the store-carry-and-forward mechanism [21](which is widely utilized to cope with the sporadic connectivity of mobile nodes in DTNs, namely, some data are temporarily stored at a node until an appropriate communication opportunity arises) of nodes can be well described in STAG. In particular, our proposed scheme, characterized by the dynamic storage process, could describe the bidirectional data transfer between two adjacent time intervals for each node. For instance, during time interval τ_q and due to the intermittent connectivity of links, a total of m units of data (reaching node v) must be stored and carried in v until next time interval τ_{q+1}. Then, during time interval τ_{q+1}, these m units of data could be finally transferred out of node v. In conclusion, the whole storage procedure can be described as a transfer process, namely, $n_{\tau_q,\tau_{q+1}}^v = [m]$. And the bidirectional storage transfer series incorporate two new storage functions: one is the forward storage function that records the amount of data transferred from τ_q to τ_{q+1}, and the other one is the reciprocal storage function that offers the data transferred from τ_{q+1} to τ_q on the basis of existing storage transfer series. Note that, the reciprocal storage function is equivalent to an offset or a correction of the forward storage process.

We want to from a specific path l, not for all paths in STAG, to explain how the bidirectional storage transfer series generate. Hence, the forward storage transfer series $N_T^v = [n_{\tau_1,\tau_2}^v, \ldots, n_{\tau_{q-1},\tau_q}^v, \ldots, n_{\tau_{h-1},\tau_h}^v]$ of node v can be obtained by Eq. (1) as

$$n_{\tau_{q-1},\tau_q}^v = \begin{cases} f_{\tau_{q-1}}^{l_{u,v}} - f_{\tau_{q-1}}^{l_{v,w}}, & q = 2, \\ n_{\tau_{q-2},\tau_{q-1}}^v + f_{\tau_{q-1}}^{l_{u,v}} - f_{\tau_{q-1}}^{l_{v,w}}, & q = 3, \ldots, h, \end{cases} \quad (1)$$

where l denotes any path connecting u, v and w in STAG, $f_{\tau_q}^{l_{u,v}}$ and $f_{\tau_q}^{l_{v,w}}$ respectively correspond to the amount of the feasible flow

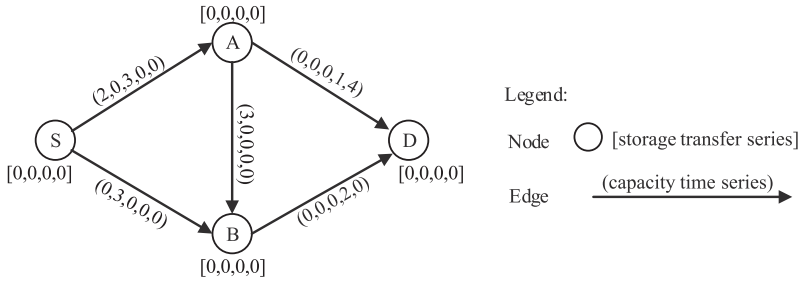

Fig. 1. Storage time aggregated graph.

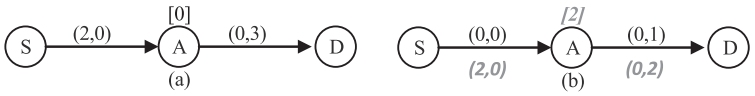

Fig. 2. Nodes with the forward storage function.

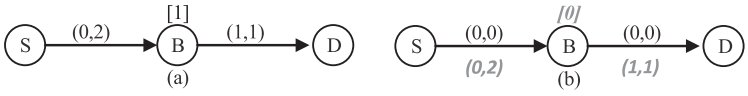

Fig. 3. Nodes with the reciprocal storage function.

into and out of node v during the time interval τ_q for the path l, $n^v_{\tau_{q-1},\tau_q}$ represents the amount of data transferred from τ_{q-1} to τ_q.

Given a storage transfer series $N^v_T = [n^v_{\tau_1,\tau_2}, \ldots, n^v_{\tau_{q-1},\tau_q}, \ldots, n^v_{\tau_{h-1},\tau_h}]$ and for an arbitrary time interval τ_q, we can derive the reciprocal storage function, which permits some data limited by N^v_T to transfer from τ_q to the previous time interval. By choosing the smallest one from the subset $\{n^v_{k-1,k} | (q \geq k \geq 2)\} \subset N^v_T$, the reciprocal transfer data β_{τ_q,τ_p} $(1 \leq p < q)$ can be obtained as follows:

$$\beta_{\tau_q,\tau_p} = \min\{n^v_{\tau_{k-1},\tau_k} | p+1 \leq k \leq q\}, \tag{2}$$

where min S denotes the minimum number in set S, τ_q and τ_p respectively represent the current and previous time interval. k acts to identify those elements $n^v_{\tau_{k-1},\tau_k}$ with $p+1 \leq k \leq q$, and the reciprocal transfer data β_{τ_q,τ_p} is equal to the minimum one of those elements $n^v_{\tau_{k-1},\tau_k}$ with $p+1 \leq k \leq q$. For instance, given $N^v_T = [2, 0, 1, 3]$ and assume $\tau_q = \tau_5$, from Eq. (2) the reciprocal transfer data β_{τ_5,τ_p} $(1 \leq p < 5)$ can be obtained, namely, $\beta_{\tau_5,\tau_4} = 3$ (means 3 units of data could be transferred from τ_5 to τ_4), $\beta_{\tau_5,\tau_3} = 1$, $\beta_{\tau_5,\tau_2} = 0$ and $\beta_{\tau_5,\tau_1} = 0$, respectively.

As shown in Fig. 2(a), there is one path in STAG where a flow with two units of data goes into node A at time interval τ_1 and out of node A at time interval τ_2. It is obvious the node A takes a storage action. From Eq. (1), the storage process of node A can be described as a forward storage transfer series $N^A_2 = [n^A_{\tau_1,\tau_2}] = [2]$, marked in gray as in Fig. 2(b). Considering another case, there is also a path in STAG in Fig. 3(a) where node B already has a storage series $N^B_2 = [n^B_{\tau_1,\tau_2}] = [1]$. From Eq. (2), it can be obtained that $\beta_{\tau_2,\tau_1} = 1$, implying that one unit of data could be transferred from τ_2 to τ_1. Hence, on the basis of $\beta_{\tau_2,\tau_1} = 1$, we can obtain a feasible flow with two units of data rather than one unit, as shown in Fig. 3(b).

3. Basic definitions in STAG

Let s be the source of the network, and d be the sink. The feasible flow in STAG is also a series $f_T = (f_{\tau_1}, \ldots, f_{\tau_q}, \ldots, f_{\tau_h})$, satisfying the following two properties:

1) Capacity constraint:

$$0 \leq f^{u,v}_{\tau_q} \leq c^{u,v}_{\tau_q} \qquad \forall 1 \leq q \leq h, \forall (u,v) \in E. \tag{3}$$

2) Flow conservation:

$$\sum_{v \in V} \sum_{q=1}^{h} f^{u,v}_{\tau_q} - \sum_{v \in V} \sum_{q=1}^{h} f^{v,u}_{\tau_q} = \begin{cases} \sum_{q=1}^{h} f_{\tau_q}, & u = s, \\ 0, & u \neq s, d, \\ -\sum_{q=1}^{h} f_{\tau_q}, & u = d. \end{cases} \tag{4}$$

Intuitively, given a flow network STAG and a flow f_T, the residual network STAG (rSTAG) consists of new added edges with capacities (for amending the flow), and nodes with the bidirectional storage transfer series. Consider a pair of vertices u, $v \in V$, we define the residual capacity time series $rc^{u,v}_T = (rc^{u,v}_{\tau_1}, \ldots, rc^{u,v}_{\tau_q}, \ldots, rc^{u,v}_{\tau_h})$ by

$$rc^{u,v}_{\tau_q} = c^{u,v}_{\tau_q} - f^{u,v}_{\tau_q}, \qquad \forall 1 \leq q \leq h, \quad \text{if}(u,v) \in E. \tag{5}$$

Correspondingly, the reverse counterpart (v, u) of edge (u, v) can also obtain a residual capacity time series $rc^{v,u}_T = (rc^{v,u}_{\tau_1}, \ldots, rc^{v,u}_{\tau_q}, \ldots, rc^{v,u}_{\tau_h})$ where

$$rc^{v,u}_{\tau_q} = \begin{cases} c^{v,u}_{\tau_q} + f^{u,v}_{\tau_q}, & \forall 1 \leq q \leq h, \quad \text{if}(v,u) \in E, \\ f^{u,v}_{\tau_q}, & \forall 1 \leq q \leq h, \quad \text{if}(v,u) \notin E. \end{cases} \tag{6}$$

As illustrated in Fig. 4(a), there exists a path $l = S - A - B - D$ in STAG, and the feasible flow in this path satisfies both the capacity constraint and the flow conservation. Following Eqs. (5) and (6), the residual network rSTAG is available as in Fig. 4(b). It can be found that, for each edge in path l, the flow summation over intervals is equal. Taking path l in Fig. 4(a) as an example, the feasible flow of edge (A, B) is $f^{l_{A,B}}_5 = (2, 0, 0, 0, 0)$, while the feasible flow of the link (B, D) is $f^{l_{B,D}}_5 = (0, 0, 0, 2, 0)$. This feature is very different from the common feasible flow, because the STAG incorporates the storage transfer series, which acts an important role in solving the maximum flow of STAG.

4. Maximum flow algorithm for storage time aggregated graph

The maximum flow problem can be stated as that we wish to send as much flow as possible between two special nodes (e.g., a

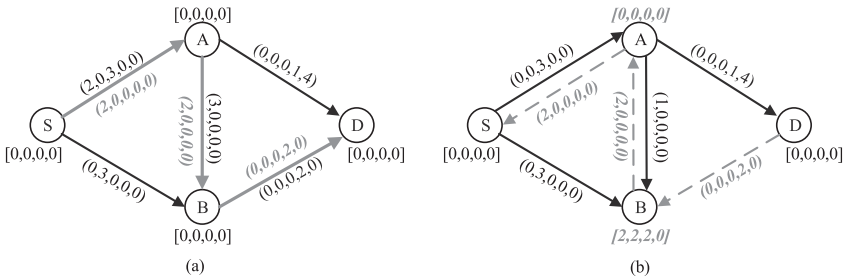

Fig. 4. (a) Path with a flow in STAG. (b) The residual network rSTAG.

source node s and a sink node d), under the capacity constraints in STAG. That is,

$$f_T^M = \max \sum_{l_i} f_T^{l_i} = \max \sum_{l_i} \sum_{q=1}^{h} f_{\tau_q}^{l_i}, \tag{7}$$

where l_i is the ith augmenting path in the residual network rSTAG, and $f_T^{l_i} = (f_{\tau_1}^{l_i}, \ldots, f_{\tau_q}^{l_i}, \ldots, f_{\tau_h}^{l_i})$ is the maximum feasible flow of l_i (which also satisfies the aforementioned two properties of the feasible flow). Moreover, f_T^M represents the maximum flow of STAG, and is equal to the summation of all augmenting paths' maximum feasible flow.

Algorithm 1 Max flow-STAG Algorithm.

Require: STAG = $\{(V, E, T, C_T^{u,v}, N_T^v) | u \in V, v \in V, (u, v) \in E\}$
Ensure: The maximum flow f_T^M from s to d for a given time period T

1 **Initialize** both flow f_T^M and the storage transfer series N_T^v to 0, and the original residual network rSTAG is equivalent to STAG;
2 **Repeat**
  1) **Adopt** the Depth First Search(DFS) method combined with the storage strategy to acquire an augmenting path l_i from s to d in the residual network;
  2) **Augment** flow f_T^M along l_i :
    (1) Compute the maximum flow $f_T^{l_i}$ and add it to f_T^M as in (7);
    (2) Update the residual network and the storage transfer series of all nodes along path l_i;
  3) **Return** f_T^M;
3 **Until** there exist no augmenting paths from s to d in the residual network.

The proposed bidirectional storage transfer series can be leveraged to exploit the correlation between time intervals and to seek as many augmenting paths as possible. Henceforth, on the basis of bidirectional storage transfer series and the maximum flow algorithm of the static network [22], we propose the *Max flow-STAG algorithm* to solve problem (7), with the following three steps: seeking an augmenting path, computing its maximum flow as well as getting its residual network. The proposed algorithm would iterate between the three steps until there is no one more augmenting path in rSTAG. As such, the maximum flow of STAG can be obtained.

4.1. Seeking an augmenting path in rSTAG

The augmenting path is a simple path from the source node to the sink node in STAG, and the Depth First Search [22] combined with the bidirectional storage transfer series can be utilized to get it. The key principle behind searching for the augmenting path is to use the storage transfer series to adjust the routing start time, from which an adjacent node is chosen as the next hop.

Algorithm 2 Augmenting path algorithm.

Require: STAG = $(V, E, T, C_T^{u,v}, N_T^v)$
Ensure: An augmenting path l
  $a_l \leftarrow s$; // set source s as the current node a_l of l
  $t_s \leftarrow \tau_q$; // the original routing start time t_s of a_l is τ_q
  $t \leftarrow q$;
  while $a_l \neq d$ **do**
    $v \leftarrow a_l$;
    for $j = t$ to 2 **do**
      if $(n_{\tau_{j-1},\tau_j}^v == 0)$ **then**
        $t_s \leftarrow \tau_j$;
        break;
      else
        $t_s \leftarrow \tau_{j-1}$;
      end if
    end for
    $z_t \leftarrow \min\{\tau_k | C_{\tau_k}^{a_l,w} \neq 0 \quad \&\& \quad \tau_k \geq t_s, (a_l, w) \in E\}$;
    $\Theta \leftarrow \{w | C_{z_t}^{a_l,w} \neq 0\}$;
    $a_l \leftarrow \min \Theta$;
    $t \leftarrow k$;
  end while

Next we illustrate Algorithm 2 via an exemplary instance. As shown in Fig. 5, node A, with a storage transfer series $N_5^A = [n_{\tau_1,\tau_2}^A, n_{\tau_2,\tau_3}^A, n_{\tau_3,\tau_4}^A, n_{\tau_4,\tau_5}^A] = [0, 2, 1, 3]$, has an original routing start time $t_s = \tau_4$. By choosing n_{τ_3,τ_4}^A (related to time interval τ_4) out of N_5^A, and the corresponding time interval τ_2 can be taken as the new routing start time of node A ($t_s = \tau_2$) since n_{τ_1,τ_2}^A is the first zero element before n_{τ_3,τ_4}^A. Following the new routing start time, the available connected time intervals of all adjacent edges of A are $t(A, B) = \tau_3$, $t(A, C) = \tau_4$ and $t(A, D) = \tau_2$, respectively. By choosing the earliest available connected time interval ($t(A, D) = \tau_2$), node D can be set as the next hop of node A. And the original routing start time of node D is marked as $t_s = \tau_2$. Repeat the above process until the sink d and thus the augmenting path l in rSTAG is eventually found.

4.2. Computing the maximum flow of the augmenting path

For the discovered augmenting path l in Section 4.1, we compute its maximum flow f_T^l and thus the residual capacity of each edge among it is available. What is more, the storage transfer series of all nodes in the path are also updated. For sake of presentation, we compute the maximum flow of an augmenting path only with two edges, although the computation process can be extended to the cases with more edges effortlessly. As shown in Fig. 6, there are three nodes u, v and w, and the capacity time series of edges (u, v) and (v, w) are $C_T^{u,v} = (c_{\tau_1}^{u,v}, \ldots, c_{\tau_q}^{u,v}, \ldots, c_{\tau_h}^{u,v})$

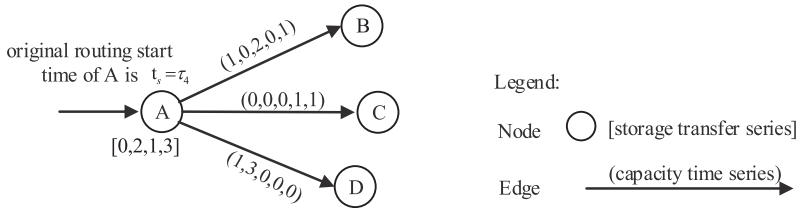

Fig. 5. Node with multiple choices to find the next hop.

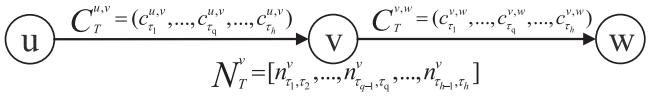

Fig. 6. Augmenting path with two links.

and $C_T^{v,w}=(c_{\tau_1}^{v,w},\ldots,c_{\tau_q}^{v,w},\ldots,c_{\tau_h}^{v,w})$, respectively. Moreover, node v is marked with an bidirectional storage transfer series $N_T^v=[n_{\tau_1,\tau_2}^v,\ldots,n_{\tau_{q-1},\tau_q}^v,\ldots,n_{\tau_{h-1},\tau_h}^v]$.

The temporary feasible flow $tf_T^{l_{v,w}}=(tf_{\tau_1}^{l_{v,w}},\ldots,tf_{\tau_q}^{l_{v,w}},\ldots,tf_{\tau_h}^{l_{v,w}})$ can be set by the capacity time series of edge (v, w), namely, $tf_T^{l_{v,w}}=C_T^{v,w}$. After that, we can obtain the temporary feasible flow $tf_T^{l_{u,v}}=(tf_{\tau_1}^{l_{u,v}},\ldots,tf_{\tau_q}^{l_{u,v}},\ldots,tf_{\tau_h}^{l_{u,v}})$ of edge (u, v), which is determined by $C_T^{u,v}$, $tf_T^{l_{v,w}}$ and N_T^v. Moreover, define a duplicate of N_T^v, i.e., $tN_T^v=N_T^v$.

The big principle behind computing the temporary feasible flow is to traverse the entire time interval reversely. In other words, for each feasible $\tau_q(h\geq q\geq 1)$, we need to compute the reciprocal feasible flow $bf_{\tau_q}^{l_{u,v}}$ as well as the forward feasible flow $sf_{\tau_q}^{l_{u,v}}$. Next, these two processes will be elaborated in detail respectively.

4.2.1. Reciprocal feasible flow $bf_{\tau_q}^{l_{u,v}}$

First, for each feasible $\tau_q(h\geq q\geq 1)$ and $\tau_p(1\leq p<q)$, we can get the reciprocal transfer data β_{τ_q,τ_p} from tN_T^v from Eq. (2). Then the corresponding reciprocal flow b_{τ_q,τ_p} is given by

$$b_{\tau_q,\tau_p}=\min\{c_{\tau_q}^{u,v},\beta_{\tau_q,\tau_p},tf_{\tau_p}^{l_{v,w}}\}. \tag{8}$$

Next, update $C_T^{u,v}$, $tf_T^{l_{v,w}}$ and tN_T^v by b_{τ_q,τ_p} as follows:

$$\begin{aligned} &c'^{u,v}_{\tau_q}=c_{\tau_q}^{u,v}-b_{\tau_q,\tau_p},\\ &tf'^{l_{v,w}}_{\tau_p}=tf_{\tau_p}^{l_{v,w}}-b_{\tau_q,\tau_p},\\ &n'^{v}_{\tau_{k-1},\tau_k}=n_{\tau_{k-1},\tau_k}^{v}-b_{\tau_q,\tau_p},\qquad \forall p+1\leq k\leq q. \end{aligned} \tag{9}$$

Following Eq. (9), $C_T^{u,v}=(c_{\tau_1}^{u,v},\ldots,c'^{u,v}_{\tau_q},\ldots,c_{\tau_h}^{u,v})$, $tf_T^{l_{v,w}}=(tf_{\tau_1}^{l_{v,w}},\ldots,tf'^{l_{v,w}}_{\tau_p},\ldots,tf_{\tau_q}^{l_{v,w}},\ldots,tf_{\tau_h}^{l_{v,w}})$ and $tN_T^v=[n_{\tau_1,\tau_2}^v,\ldots,n'^{v}_{\tau_p,\tau_{p+1}},\ldots,n'^{v}_{\tau_{q-1},\tau_q},\ldots,n_{\tau_{h-1},\tau_h}^v]$. Furthermore, we can acquire $bf_{\tau_q}^{l_{u,v}}$ by summing b_{τ_q,τ_p} over $\tau_p(1\leq p<q)$, namely,

$$bf_{\tau_q}^{l_{u,v}}=\sum_{p=1}^{q-1}b_{\tau_q,\tau_p}. \tag{10}$$

Especially, we set the $bf_{\tau_1}^{l_{u,v}}=0$ for $\tau_q=\tau_1$.

4.2.2. Forward feasible flow $sf_{\tau_q}^{l_{u,v}}$

To compute the forward feasible flow $sf_{\tau_q}^{l_{u,v}}$, we first assume that

$$bf_{\tau_k}^{e}=sf_{\tau_k}^{e}=0,\qquad k\notin\{1,\ldots,h\},e\in l. \tag{11}$$

Substituting $bf_{\tau_q}^{l_{u,v}}$ into Eq. (12), $sf_{\tau_q}^{l_{u,v}}$ is given by

$$sf_{\tau_q}^{l_{u,v}}=\min\{c_{\tau_q}^{u,v}-bf_{\tau_q}^{l_{u,v}},\sum_{k=q}^{h}tf_{\tau_k}^{l_{v,w}}-\sum_{i=q+1}^{h}sf_{\tau_i}^{l_{u,v}}\}. \tag{12}$$

The temporary feasible flow $tf_{\tau_q}^{l_{u,v}}$ is equal to the summation of $bf_{\tau_q}^{l_{u,v}}$ and $sf_{\tau_q}^{l_{u,v}}$ from Eq. (13), namely,

$$tf_{\tau_q}^{l_{u,v}}=bf_{\tau_q}^{l_{u,v}}+sf_{\tau_q}^{l_{u,v}}. \tag{13}$$

Eventually, on the basis of the temporary feasible flow $tf_T^{l_{u,v}}=(tf_{\tau_1}^{l_{u,v}},\ldots,tf_{\tau_q}^{l_{u,v}},\ldots,tf_{\tau_h}^{l_{u,v}})$, the path l's maximum flow is available, namely, $f_T^l=tf_T^{l_{u,v}}$.

The aforementioned algorithm could be also extended to the augmenting path with more than two edges. In reality, starting from the last edge of the path, we can compute the temporary feasible flow of its preceding one. Repeat the process until the first edge is computed, and then we can get its temporary feasible flow and take it as the maximum flow of the path. Due to the space limitation, nevertheless, we omit the detailed process here.

4.3. Getting the residual network

In order to get the residual network, it is of great importance to compute the feasible flow of each edge, which is similar to the process of computing the maximum flow of the path. We also take the augmenting path in Fig. 6 as an actual example.

Note here that, we start from the first edge of the augmenting path to compute the relevant feasible flow and the residual capacity. Since the maximum flow f_T^l is available from Section 4.2, set it as the feasible flow of edge (u, v), namely, $f_T^{l_{u,v}}=f_T^l$. Hence, the residual capacity of edge (u, v) and the reverse one (v, u) can be obtained by

$$\begin{cases} rc_{\tau_p}^{u,v}=c_{\tau_p}^{u,v}-f_{\tau_p}^{l_{u,v}}, & 1\leq p\leq h,\\ rc_{\tau_p}^{v,u}=c_{\tau_p}^{v,u}+f_{\tau_p}^{l_{u,v}}, & 1\leq p\leq h, \end{cases} \tag{14}$$

respectively. After that, we can obtain the feasible flow $f_T^{l_{v,w}}=(f_{\tau_1}^{l_{v,w}},\ldots,f_{\tau_p}^{l_{v,w}},\ldots,f_{\tau_h}^{l_{v,w}})$ of edge (v, w), which is determined by $f_T^{l_{u,v}}$, $c_T^{l_{v,w}}$ and N_T^v.

Then, we traverse the entire time interval orderly to compute $f_T^{l_{v,w}}$. For each feasible $\tau_p(1\leq p\leq h)$, we need to compute the reciprocal feasible flow $af_{\tau_p}^{l_{v,w}}$ as well as the forward feasible flow $df_{\tau_p}^{l_{v,w}}$. It should be noted here that these two processes are different from the processes in Section 4.2.1 and Section 4.2.2, since the latter ones focus on time interval τ_q while the former ones concentrate on τ_p. Next, these two processes also will be elaborated in detail respectively.

4.3.1. Reciprocal feasible flow $af_{\tau_p}^{l_{v,w}}$

First, for each feasible $\tau_p(1\leq p\leq h)$ and $\tau_q(h\geq q>p)$, we can get the amount of reciprocal transfer data β_{τ_q,τ_p} from N_T^v by

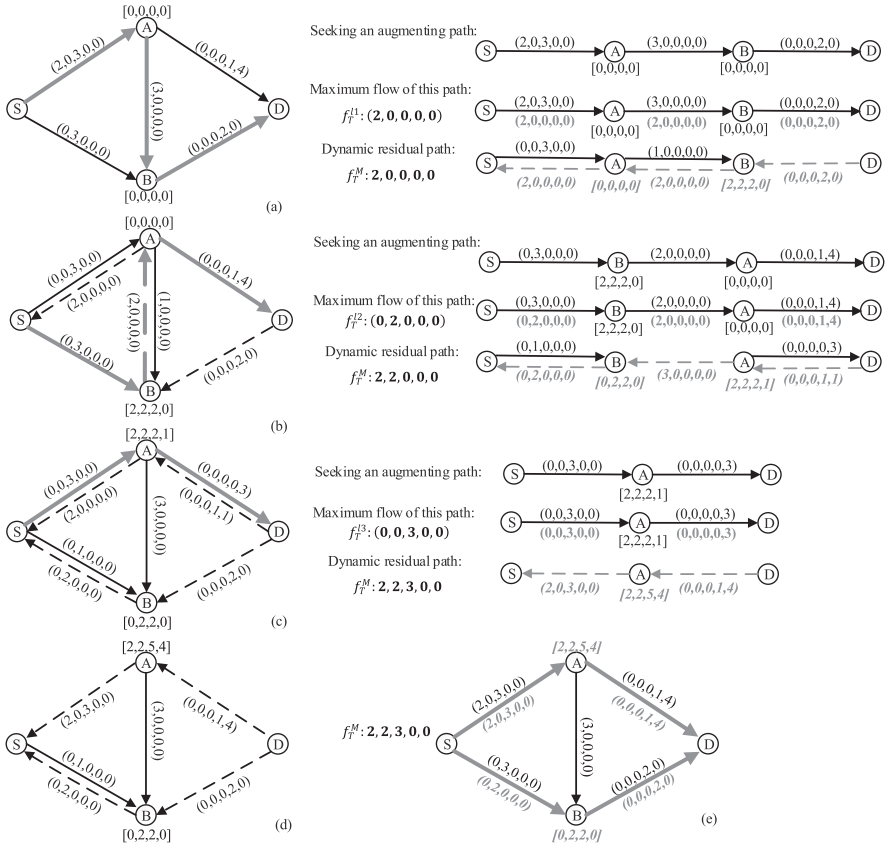

Fig. 7. Example of Max flow-STAG algorithm.

Eq. (2). Then the corresponding reciprocal flow a_{τ_q,τ_p} is given by

$$a_{\tau_q,\tau_p} = \min\{f^{l_{u,v}}_{\tau_q}, \beta_{\tau_q,\tau_p}, c^{v,w}_{\tau_p}\}. \tag{15}$$

Next, update $f^{l_{u,v}}_T$, $C^{v,w}_T$ and N^v_T by a_{τ_q,τ_p} as follows:

$$\begin{aligned} &f'^{l_{u,v}}_{\tau_q} = f^{l_{u,v}}_{\tau_q} - a_{\tau_q,\tau_p},\\ &c'^{v,w}_{\tau_p} = c^{v,w}_{\tau_q} - a_{\tau_q,\tau_p},\\ &n'^{v}_{\tau_{k-1},\tau_k} = n^{v}_{\tau_{k-1},\tau_k} - a_{\tau_q,\tau_p}, \qquad \forall p+1 \le k \le q. \end{aligned} \tag{16}$$

From Eq. (16), $f^{l_{u,v}}_T = (f^{l_{u,v}}_{\tau_1}, \ldots, f'^{l_{v,w}}_{\tau_q}, \ldots, f^{l_{v,w}}_{\tau_h})$, $C^{v,w}_T = (c^{u,v}_{\tau_1}, \ldots, c'^{u,v}_{\tau_p}, \ldots, c^{u,v}_{\tau_q}, \ldots, c^{u,v}_{\tau_h})$ and $N^v_T = [n^v_{\tau_1,\tau_2}, \ldots, n'^v_{\tau_p,\tau_{p+1}}, \ldots, n'^v_{\tau_{q-1},\tau_q}, \ldots, n^v_{\tau_{h-1},\tau_h}]$. Moreover, we can acquire $af^{l_{v,w}}_{\tau_p}$ by summing a_{τ_q,τ_p} over $\tau_q (p < q \le h)$, namely,

$$af^{l_{v,w}}_{\tau_p} = \sum_{q=p+1}^{h} a_{\tau_q,\tau_p}. \tag{17}$$

Especially, we set $af^{l_{v,w}}_{\tau_h} = 0$ for $\tau_p = \tau_h$.

4.3.2. Forward feasible flow $df^{l_{v,w}}_{\tau_p}$

To compute the forward feasible flow $df^{l_{v,w}}_{\tau_p}$, we also assume that

$$af^e_{\tau_k} = df^e_{\tau_k} = 0, \qquad k \notin \{1, \ldots, h\}, e \in l. \tag{18}$$

Substituting $af^{l_{v,w}}_{\tau_p}$ into Eq. (19), $df^{l_{v,w}}_{\tau_p}$ is given by

$$df^{l_{v,w}}_{\tau_p} = \min\{c^{v,w}_{\tau_p} - af^{l_{v,w}}_{\tau_p}, \sum_{k=1}^{p} f^{l_{u,v}}_{\tau_k} - \sum_{i=1}^{p-1} df^{l_{v,w}}_{\tau_i}\}. \tag{19}$$

The feasible flow $f^{l_{v,w}}_{\tau_p}$ is equal to the summation of $af^{l_{v,w}}_{\tau_p}$ and $df^{l_{v,w}}_{\tau_p}$ from Eq. (20), namely,

$$f^{l_{v,w}}_{\tau_p} = af^{l_{v,w}}_{\tau_p} + df^{l_{v,w}}_{\tau_p}. \tag{20}$$

Furthermore, the residual capacity of edge (v, w) and the reverse one (w, v) can be obtained by

$$\begin{cases} rc^{v,w}_{\tau_p} = c^{v,w}_{\tau_p} - f^{l_{v,w}}_{\tau_p}, & 1 \le p \le h,\\ rc^{w,v}_{\tau_p} = c^{w,v}_{\tau_p} + f^{l_{v,w}}_{\tau_p}, & 1 \le p \le h. \end{cases} \tag{21}$$

Finally, on the basis of $f^{l_{u,v}}_T$, $f^{l_{v,w}}_T$ and updated N^v_T, we can obtain a new bidirectional storage transfer series $zN^v_T = [zn^v_{\tau_1,\tau_2}, \ldots, zn^v_{\tau_{q-1},\tau_q}, \ldots, zn^v_{\tau_{h-1},\tau_h}]$ of node v as follows:

$$zn^v_{\tau_{q-1},\tau_q} = \begin{cases} n^v_{\tau_{q-1},\tau_q} + f^{l_{u,v}}_{\tau_{q-1}} - f^{l_{v,w}}_{\tau_{q-1}}, & q = 2,\\ zn^v_{\tau_{q-2},\tau_{q-1}} + n^v_{\tau_{q-1},\tau_q} + f^{l_{u,v}}_{\tau_{q-1}} - f^{l_{v,w}}_{\tau_{q-1}}, & q = 3, \ldots, h. \end{cases} \tag{22}$$

5. Analysis of Max flow-STAG algorithm

5.1. Algorithm complexity analysis

Theorem 1. *The time complexity of the Max flow-STAG algorithm is $O((m + nh)|f^M_T|)$, where h is the number of time intervals (for a given time period T, we partition T into a set of h small time intervals), f^M_T denotes the value of the maximum flow in a DTN, n is the number of nodes and m represents the number of edges in STAG.*

Proof. We combine the depth-first search method with the earliest start time of nodes to find an augmenting path in rSTAG, and the time complexity for this step reaches $O(m + n)$. Besides, both

the time for computing the maximum flow of a augmenting path and getting the residual network rSTAG are $O((m+nh)$. Hence, the total time complexity for each augmenting path is $O(m+nh)$. Assuming f_T^M is the maximum flow in a DTN, if we set unit value of flow augmentation for each iteration, then the total running time of the Max flow-STAG algorithm reaches $O((m+nh)|f_T^M|)$.

5.2. Case analysis

For a better understanding of the proposed Max flow-STAG algorithm, an example will be given to illustrate its application in solving the maximum flow problem of the DTN.

As shown in Fig. 7(a), the STAG models a DTN, where the source is S, the sink is D, and the given time period is $T=[t_0,t_h)=[0,5)$. Note here that, we partition T into 5 small time intervals, and assume that the initial feasible flow is equal to zero. First, we use Algorithm 2 (Augmenting Path Algorithm) to seek an augmenting path $l_1=S-A-B-D$, which is marked in gray. Then, we compute the maximum flow of path l_1 as $f_T^{l_1}=(2,0,0,0,0)$. Finally, on the basis of $f_T^{l_1}=(2,0,0,0,0)$, the augmented maximum flow f_T^M along l_1 and thus the residual network are acquired.

Fig. 7(b) describes the residual network rSTAG after the first augmentation, where the dotted line represents the first augmented path. Via Algorithm 2, an augmenting path $l_2=S-B-A-D$ (also marked in gray) is available, and the path l_2's maximum flow turns out to be $f_T^{l_2}=(0,2,0,0,0)$. Following it, we augment maximum flow f_T^M along l_2 and acquire the residual network.

Fig. 7(c) and (d) describe the residual network rSTAG after the second and third augmentation, respectively. From Fig. 7(d), it can observed that there exists no augmenting path any more in rSTAG, and hence the algorithm terminates. Therefore, the DTN's maximum flow f_T^M is equal to the summation of $f_T^{l_i}$ of all augmenting paths, as shown in Fig. 7(e). Besides, it can also provide a routing scheme with the maximum flow along the paths $S-A-D$ and $S-B-D$ to transfer data.

6. Conclusion and future work

In this paper, we described a model named as STAG to represent the DTN. First, the bidirectional storage transfer series was introduced to each node in STAG. Second, a bidirectional storage transfer series-based maximum flow algorithm was proposed and described in detail to maximize the network flow. Finally, the effectiveness of the proposed algorithm was also illustrated.

As an innovative approach, the proposed STAG-based maximum flow algorithm (i.e., Max-flow STAG algorithm) has broken through the limitation that the TAG could not solve the maximum flow problem of the DTN. In practice, the motion of the satellite would result in the time-varying network topology and link, and thus the satellite networks are typical DTNs. However, there exist some large data services (e.g., observation service, video service) that need to be transmitted in real time. To ensure the efficient and reliable transmission of the services, it is necessary to construct a time-varying model for the satellite networks, followed by the corresponding routing schemes. The STAG could sufficiently describe the satellite networks' time-varying characteristics, and the graph could be simpler and less memory by aggregating the link and node attributes. Furthermore, the proposed Max-flow STAG algorithm could provide an efficient routing scheme for the services, and which can make full use of the scarce link resources to realize the efficient and reliable transmission of large data services. In addition, DTNs (e.g., sensor networks, ad hoc networks) are also faced with the same problem, i.e., how to ensure the efficient transmission of large data services in the time-varying networks. In a word, the constructed model STAG and the proposed STAG-based maximum flow algorithm would be widely applied to the time-varying communication networks and transportation systems.

Future works will be extended to the case that nodes possess limited storage capacity, which is a more practical scenario in real-world DTNs.

Acknowledgments

This work is supported by the National Science Foundation (91338115, 61231008), National S&T Major Project (2015ZX03002006), the Fundamental Research Funds for the Central Universities (WRYB142208, JB140117), Program for Changjiang Scholars and Innovative Research Team in University (IRT0852), the 111 Project (B08038), SAST (201454). Furthermore, we thank the reviewers for their detailed reviews and constructive comments, which have helped to improve the quality of this paper.

References

[1] J. Rodrigues, V. Soares, An introduction to delay and disruption-tolerant networks, in: Advances in Delay-Tolerant Networks (DTNs), Woodhead Publishing, Oxford, 2015, pp. 1–21.
[2] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proc. ACM SIGCOM, Karlsruhe, Germany, 2003, pp. 27–34.
[3] C. Caini, et al., Delay and disruption tolerant networking: an alternative solution for future satellite networking applications, Proc. IEEE 99 (2011) 1980–1997.
[4] J. Fraire, J. Finochietto, Routing-aware fair contact plan design for predictable delay tolerant networks, Ad Hoc Netw. 25, Part B (2015) 303–313.
[5] Y. Lu, X. Li, et al., Information-centric delay-tolerant mobile ad-hoc networks, in: Computer Communications Workshops (INFOCOM WKSHPS), Toronto, Canada, 2014, pp. 428–433.
[6] H. Chen, W. Lou, On protecting end-to-end location privacy against local eavesdropper in wireless sensor networks, Pervasive Mob. Comput. 16, Part A (2015) 36–50.
[7] M. Auzias, et al., Coap over bp for a delay-tolerant internet of things, in: Future Internet of Things and Cloud (FiCloud), Rome, Italy, 2015, pp. 118–123.
[8] J. Vijayanathan, P. Vijayanathan, Delay tolerant social networking (dtsn): dual architecture for dtn based social networking, in: Telecommunication Systems, Services, and Applications (TSSA), 2011, pp. 55–58.
[9] Z. Wang, J. Liao, et al., Friendbook: a semantic-based friend recommendation system for social networks, IEEE Trans. Mob. Comput. 14 (3) (2015) 538–551.
[10] L. Ford, D. Fulkerson, Flows in Networks, Princeton U. Press, Princeton, NJ, 1962.
[11] M. Werner, A dynamic routing concept for atm-based satellite personal communication networks, IEEE J. Sel. Areas Commun. 15 (8) (1997) 1636–1648.
[12] E. Köhler, et al., Time-expanded graphs for flow-dependent transit times, in: Proc. 10th Annu. Eur. Symp. Algorithms, 2002, pp. 49–56.
[13] A. Ferreira, Building a reference combinatorial model for manets, IEEE Netw. 18 (5) (2004) 24–29.
[14] B. George, S. Kim, Spatio-temporal network databases and routing algorithms: a summary of results, in: Advances in Spatial and Temporal Databases, Boston, 2007, pp. 460–477.
[15] B. George, S. Shekhar, Time-aggregated graphs for modeling spatio-temporal networks, in: Journal on Data Semantics XI, Springer, 2008, pp. 191–212.
[16] D. Hay, P. Giaccone, Optimal routing and scheduling for deterministic delay tolerant networks, in: Wireless On-Demand Network Systems and Services, 2009, pp. 27–34.
[17] G. Iosifidis, et al., The impact of storage capacity on end-to-end delay in time varying networks, in: INFOCOM, 2011 Proceedings IEEE, 2011, pp. 1494–1502.
[18] G. Konidaris, et al., Primal decomposition and online algorithms for flow optimization in wireless dtns, in: Global Communications Conference (GLOBECOM), 2013, pp. 84–90.
[19] S. Jain, et al., Routing in a delay tolerant network, SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 145–158.
[20] Z. Wu, G. Hu, et al., Agent-based dynamic routing in the packet-switched leo satellite networks, in: Wireless Communications Signal Processing (WCSP), 2015, pp. 1–6.
[21] C. Glacet, M. Fiore, M. Gramaglia, Temporal connectivity of vehicular networks: The power of store-carry-and-forward, in: Vehicular Networking Conference (VNC), 2015 IEEE, 2015, pp. 52–59.
[22] K.A. Ravindra, L.M. Thomas, et al., Network Flows: Theory, Algorithms and Application, Pearson Education, New York, 1993.

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 wireless networking, cognitive networks, integration of heterogeneous network, and mobile ad hoc networks.

Tao Zhang received the B.S. degree in telecommunication engineering from 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, China. His current research interests include wireless networking, cognitive networks, integration of heterogeneous network, and mobile ad hoc networks.

Yangkun Zhang is currently an undergraduate student in the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, China. His current research interests is network optimization.

Kan Wang received the B.S. degree in broadcasting and television engineering from Zhejiang University of Media and Communications, Hangzhou, China, in 2009. He is currently working toward the Ph.D. degree in military communications with the State Key Lab of ISN, Xidian University, Xi'an, China. From Oct. 2014 to Oct. 2015, he was also with Carleton University, Ottawa, ON, Canada, as a visiting scholar funded by China Scholarship Council (CSC). His current research interests include 5G cellular networks, resource management, and interference alignment.

Jiandong Li 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 of the School of Telecommunications Engineering at Xidian University since 1985, where he is currently a Professor and Vice Director of the Academic Committee of State Key Laboratory of Integrated Service Networks. He was a Visiting Professor to the Department of Electrical and Computer Engineering at Cornell University from 2002 to 2003. He served as the General Vice Chair for ChinaCom 2009 and TPC Chair of IEEE ICCC 2013. He was awarded as Distinguished Young Researcher from NSFC and Changjiang Scholar from Ministry of Education, China, respectively. His major research interests include wireless communication theory, cognitive radio, and signal processing.