Supplementary Material for the Paper: "Deterministic Multicast via Temporal Graph-based Routing and Scheduling over Non-terrestrial Networks"

Keyi Shi[1], Hongyan Li[1*], Peng Wang[2], Fei Liu[1]
[1]State Key Laboratory of Integrated Service Networks, Xidian University, Xian, 710071, China
[2]pillar of ISTD, Singapore University of Technology and Design, 487372, Singapore
Email: {kyshi10091@163.com, hyli@xidian.edu.cn, pengwangclz@163.com, feiliu3@stu.xidian.edu.cn}

In this document, we present the detailed derivations for Lemma 1, Theorem 1, Theorem 2, and Theorem 3.

Appendix A
Proof of Lemma 1

We verify ***Lemma* 1** using proof by contradiction. Assume that the data of m passes through a loop consisting of links $(v, w), (w, r), \ldots$, and (u, v). It follows from (12) that $L(v) \leq (h_{v,w} - 1) \cdot |\tau|, L(w) \leq (h_{w,r} - 1) \cdot |\tau|, \ldots$, and $L(u) \leq (h_{u,v} - 1) \cdot |\tau|$ hold, where $h_{v,w}, h_{w,r}, \ldots$, and $h_{u,v}$ denote the cycles in which the data is transmitted along $(v, w), (w, r), \ldots$, and (u, v), respectively. Based on (10), we have $L(w) = (h_{v,w} - 1) \cdot |\tau| + l_{v,w}^{h_{v,w}}, L(r) = (h_{w,r} - 1) \cdot |\tau| + l_{w,r}^{h_{w,r}}, \ldots$, and $L(v) = (h_{u,v} - 1) \cdot |\tau| + l_{u,v}^{h_{u,v}}$. Since the delays of these links, denoted as $l_{v,w}^{h_{v,w}}, l_{w,r}^{h_{w,r}}, \ldots$, and $l_{u,v}^{h_{u,v}}$, are non-negative, we can obtain $(h_{v,w}-1)\cdot|\tau| < L(w), (h_{w,r}-1)\cdot|\tau| < L(r), \ldots$, and $(h_{u,v}-1)\cdot|\tau| < L(v)$. We further derive that $L(v) < (h_{u,v}-1)\cdot|\tau|$, contradicting $(h_{u,v} - 1) \cdot |\tau| < L(v)$ above. Therefore, the assumption does not hold, and the data transmission is loop-free.

The proof is complete.

Appendix B
Proof of Theorem 1

Assume that $\mathcal{G}$ is stored in an adjacency list. First, the space of $O(|\mathcal{V}|)$ is needed to maintain each node in the set $\mathcal{V}$. Then, we create a list entry for each node, e.g., $u^h \in \mathcal{V}$, to record the delay and capacity information of all transmission edges and storage edges with u^h as the origin node. Denoting the out-degree of u^h as $\deg(u^h)$, the required space is $O(2 \cdot \deg(u^h)) = O(\deg(u^h))$. Finally, the overall space complexity of $\mathcal{G}$ reaches $O\left(|\mathcal{V}| + \sum_{u^h \in \mathcal{V}} \deg(u^h)\right) = O(|\mathcal{V}| + |\mathcal{E}|)$. Generally, since $\mathcal{G}$ is a connected graph, $|\mathcal{E}| + 1 \geq |\mathcal{V}|$ holds and the space complexity can be further reduced to $O(|\mathcal{E}|)$.

Considering that $\mathcal{G}$ represents the considered NTN in a time-slotted manner, initially, the satellites and links are replicated into H copies, and storage edges are introduced between the same satellites in adjacent cycles. Thus, we have $|\mathcal{V}| = |V| \cdot H$ and $|\mathcal{E}| = |E| \cdot H + |V| \cdot (H - 1)$. After the pruning and enhancing process, no fewer than $|V|-1$ nodes will be removed, and the transmission and storage edges will be reduced by at least $|E|$ and $|V| - 1$, respectively. Together with the virtual node and $N \cdot (H - 1)$ virtual edges added for each of the N destination satellites, we can deduce that $|\mathcal{V}| = |V| \cdot (H-1) + 2$ and $|\mathcal{E}| = |E| \cdot (H-1) + |V| \cdot (H-2) + N \cdot (H-1) - 1$ in the worst case. Therefore, the space complexity of $\mathcal{G}$ becomes $O(|E| \cdot (H - 1) + |V| \cdot (2 \cdot H - 3) + N \cdot (H - 1) + 1) = O((|V| + |E| + N) \cdot H)$. Since the NTN is a connected network and all N destination satellites are selected from V, $|E| + 1 \geq |V| \geq N$ holds, and the space complexity can be further reduced to $O(|E| \cdot H)$.

The proof is complete.

Appendix C
Proof of Theorem 2

For **Algorithm 1**, the key is to demonstrate that each node extracted from $\mathcal{Q}$ has determined its maximum bottleneck capacity. This assertion remains valid for $s^{\check{h}}$ with $C(s^{\check{h}}) = \infty$. Moving on to the K-th extracted node, denoted as u^h (or d'_n), we identify that its bottleneck capacity cannot be further improved by relaying via any node out of $\mathcal{Q}$ (steps 8 to 9, 13 to 14, and 16 to 17). Regarding potential relaying via a node in $\mathcal{Q}$, we adopt proof by contradiction for analysis, accounting for three cases:

i) If reaching u^h via $w^i \in \mathcal{Q}$ along $(w^i, u^i) \in \mathcal{E}_t$ enables $C(u^h) < \min\left\{C(w^i), c_{w,u}^i\right\} \leq C(w^i)$, where $h = \left\lceil \frac{1}{|\tau|} \cdot ((i-1) \cdot |\tau| + l_{w,u}^i) \right\rceil$, a contradiction arises because $C(w^i) \leq C(u^h)$ is enforced by step 4;

ii) If reaching u^h via $u^{h-1} \in \mathcal{Q}$ along $(u^{h-1}, u^h) \in \mathcal{E}_s$ enables $C(u^h) < \min\left\{C(u^{h-1}), c_{u,u}^h\right\} = C(u^{h-1}$, a contradiction also arises due to $C(u^{h-1}) \leq C(u^h)$;

iii) If reaching d'_n via $d_n^i \in \mathcal{Q}$ along $(d_n^i, d'_n) \in \mathcal{E}_v$ enables $C(d'_n) < \min\left\{C(d_n^i), c_{d_n,d'_n}^i\right\} = C(d_n^i)$, it encounters a contradiction with $C(d_n^i) \leq C(d'_n)$.

Therefore, the bottleneck capacity of the K-th extracted node cannot be improved.

Intuitively, the capacity of the output TF tree $\mathcal{T}$, calculated as $C(\mathcal{T}) = \min_{(u^h, v^h) \in \mathcal{T}} c_{u,v}^h = \min_{d'_n \in \mathcal{V}_v} C(d'_n)$, must reach its maximum since each $C(d'_n)$ is maximized when the algorithm terminates. Additionally, the input $\mathcal{G}$ has ensured that $\mathcal{T}$ does not contain edges with insufficient capacity or unsatisfied delay, and step 6 enforces the correct forwarding timing at each hop.

The proof is completed.

APPENDIX D
PROOF OF THEOREM 3

In **Algorithm 1**,we assume that the input $\mathcal{G}$ and the introduced $\mathcal{Q}$ are stored in an adjacency list and a binary heap, respectively. The initialization in step 2 takes $O(|\mathcal{V}|)$ time. During each iteration from steps 3 to 19, it requires $O(1)$ time to extract the node u^h with the maximum bottleneck capacity from $\mathcal{Q}$ and $O(\log |\mathcal{V}|)$ time to update $\mathcal{Q}$ (in step 4). Furthermore, looking up each edge (i.e., transmission edge, storage edge, or virtual edge) and updating the parameters (i.e., bottleneck capacity, delay, and pre-node) of the node actually reached along that edge takes $O(\log |\mathcal{V}|)$ time. Denoting the out-degree of u^h as $\deg\left(u^h\right)$, the time complexity of the process from step 5 to 18 is at most $O\left(\deg(u^h) \cdot \log |\mathcal{V}|\right)$. At worst, we must traverse all nodes in $\mathcal{V}$ once before extracting all the virtual nodes $d'_n \in \mathcal{V}_v$ from $\mathcal{Q}$. Therefore, the time complexity reaches $O\left(\sum_{u^h \in \mathcal{V}} \deg(u^h) \cdot \log |\mathcal{V}|\right) = O(|\mathcal{E}| \cdot \log |\mathcal{V}|)$. Additionally, the backtracking process takes at most $O(|\mathcal{E}|)$ time. Thus, the time complexity of **Algorithm 1** can be calculated as $O(|\mathcal{V}| + |\mathcal{E}| \cdot \log |\mathcal{V}| + |\mathcal{E}|)$. Generally, since $\mathcal{G}$ is a connected graph, $|\mathcal{E}| + 1 \geq |\mathcal{V}| \geq 2$ holds, and the time complexity can be further reduced to $O(|\mathcal{E}| \cdot \log |\mathcal{V}|)$.

The proof is completed.