

# Supplementary Material for the Paper: “Deterministic Multicast Using Temporal Graph-based Routing and Scheduling in Non-terrestrial Networks”

Keyi Shi<sup>1\*</sup>, Xi Jiang<sup>1</sup>, Tang Tang<sup>2</sup>, Xudong Zhang<sup>1</sup>

<sup>1</sup> No.20th Institute of China Electronics Technology Group Corporation, Xi'an 710068

<sup>2</sup>Information and Navigation College, Air Force Engineering University, Xi'an 710048, China.

Email:{kyshi10091@163.com, 20011110239@stu.xidian.edu.cn, tangt\_edu@126.com, wwwzxddn@nwsuaf.edu.cn}

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

## APPENDIX A PROOF OF **Lemma 1**

We verify **Lemma 1** using proof by contradiction. Assume that the task data is transmitted along a loop consisting of the links  $(v, w), (w, r), (r, p), \dots, (q, u)$  and  $(u, v)$ . According to Equation (10), we can obtain:  $L(w) = (h_{v,w} - 1) \cdot |\tau| + l_{v,w}^{h_{v,w}}$ ,  $L(r) = (h_{w,r} - 1) \cdot |\tau| + l_{w,r}^{h_{w,r}}$ ,  $L(p) = (h_{r,p} - 1) \cdot |\tau| + l_{r,p}^{h_{r,p}}$ , ...,  $L(u) = (h_{q,u} - 1) \cdot |\tau| + l_{q,u}^{h_{q,u}}$  and  $L(v) = (h_{u,v} - 1) \cdot |\tau| + l_{u,v}^{h_{u,v}}$ , where  $h_{v,w}, h_{w,r}, h_{r,p}, \dots, h_{q,u}$  and  $h_{u,v}$  represent the cycles within which task data transmitted along different links, and  $l_{v,w}^{h_{v,w}}, l_{w,r}^{h_{w,r}}, l_{r,p}^{h_{r,p}}, \dots, l_{q,u}^{h_{q,u}}$  and  $l_{u,v}^{h_{u,v}}$  denote the delay of these links. Since link delays are nonnegative, it follows that:  $(h_{v,w} - 1) \cdot |\tau| < L(w)$ ,  $(h_{w,r} - 1) \cdot |\tau| < L(r)$ ,  $(h_{r,p} - 1) \cdot |\tau| < L(p)$ , ...,  $(h_{q,u} - 1) \cdot |\tau| < L(u)$  and  $(h_{u,v} - 1) \cdot |\tau| < L(v)$ . According to Inequality (12),  $L(v) \leq (h_{v,w} - 1) \cdot |\tau|$ ,  $L(w) \leq (h_{w,r} - 1) \cdot |\tau|$ ,  $L(r) \leq (h_{r,p} - 1) \cdot |\tau|$ , ...,  $L(q) \leq (h_{q,u} - 1) \cdot |\tau|$ , and  $L(u) \leq (h_{u,v} - 1) \cdot |\tau|$  hold. Combining the above inequality groups, we derive two key results:

- On one hand,  $h_{v,w} < h_{w,r}, h_{w,r} < h_{r,p}, \dots, h_{q,u} < h_{u,v}$  hold, indicating that  $h_{v,w} < h_{u,v}$ .
- On the other hand,  $(h_{u,v} - 1) \cdot |\tau| < L(v)$  and  $L(v) \leq (h_{v,w} - 1) \cdot |\tau|$  hold, implying that  $h_{u,v} < h_{v,w}$ .

Since the two results are contradictory, the initial assumption does not hold. Therefore, we conclude that the task data transmission is loop-free. The proof is thus complete.

## APPENDIX B APPLICATION OF ALGORITHM 1

As illustrated in Fig. 1(a), the input of **Algorithm 1** consists of the pruned and enhanced time-expanded graph (TEG)  $\mathcal{G}$  and a time-critical (TC) multicast task denoted as  $m = \langle s, \{d_1, d_2\}, 33.33\text{ms}, 0.3\text{Mb}, 1\text{ms}, 11\text{ms} \rangle$ . The detailed procedure of **Algorithm 1** is described as follows:

- **Initialization process.** The desired time-featured (TF) tree  $\mathcal{T}$  is initialized as  $\emptyset$ . The “pend-capacity”  $C(\cdot)$  is set to  $\infty$  for  $s^1$  and to 0 for all other nodes. The “arrival-time”  $L(\cdot)$  is initialized to 1ms for  $s^1$  and to  $\infty$  for all other nodes. In addition, the “pre-node”  $P(\cdot)$  of each node is set to NULL, and the priority queue  $Q$  is initialized to contain the entire node set  $\mathcal{V}$ .
- **Search process.** In **Iteration 1**, node  $s^1$  is extracted from  $Q$  due to its maximal “pending-capacity”,  $C(s^1) = \infty$ . Since the storage edge (SE)  $(s^1, s^2)$  is the only outgoing edge from  $s^1$ , the parameters of  $s^2$  are updated as  $C(s^2) = \infty$ ,  $L(s^2) = 2\text{ms}$ , and  $P(s^2) = s^1$  (steps 13–15). In **Iteration 2**, node  $s^2$  is popped from  $Q$ . Considering the transmission edge (TE)  $(s^2, w^2)$  from  $s^2$ , the arrival cycle number of the task data transmitted along it is determined as  $I(w) = 3$ . Consequently, the actual arrival node through cross-cycle transmission is  $w^3$ , whose “pending-capacity” is improved from 0 to 2Mb, together with updated parameters  $L(w^3) = 5\text{ms}$  and  $P(w^3) = s^2$  (steps 6–11). Furthermore, by checking the SE  $(s^2, s^3)$ , we obtain  $C(s^3) = \infty$ ,  $L(s^3) = 4\text{ms}$ , and  $P(s^3) = s^2$ . During **Iterations 3–6**, nodes  $s^3, s^4$  ( $C(s^4) = \infty$ ,  $L(s^4) = 6\text{ms}$ ,  $P(s^4) = s^3$ ),  $s^5$  ( $C(s^5) = \infty$ ,  $L(s^5) = 8\text{ms}$ ,  $P(s^5) = s^4$ ), and  $s^6$  ( $C(s^6) = \infty$ ,  $L(s^6) = 10\text{ms}$ ,  $P(s^6) = s^5$ ) are sequentially selected. In **Iteration 9**, node  $w^3$  is extracted from  $Q$ . Due to the violation of the *forwarding time constraint* (step 6), the two TEs,  $(w^3, v^3)$  and  $(w^3, d_1^3)$ , are not visited. However, through TE  $(w^3, w^4)$ , the parameters of  $w^4$  are updated to  $C(w^4) = 2\text{Mb}$ ,  $L(w^4) = 6\text{ms}$ , and  $P(w^4) = w^3$ . In **Iteration 10**, the TE  $(w^4, v^4)$  is first examined, yielding  $C(v^5) = 2\text{Mb}$ ,  $L(v^5) = 9\text{ms}$ , and  $P(v^5) = w^4$ . Subsequently, the TE  $(w^4, d_1^4)$  is checked, obtaining  $C(d_1^4) = 1\text{Mb}$ ,  $L(d_1^4) = 7\text{ms}$ , and  $P(d_1^4) = w^4$ . Finally, the SE  $(w^4, w^5)$  is examined, resulting in  $C(w^5) = 2\text{Mb}$ ,  $L(w^5) = 8\text{ms}$ , and  $P(w^5) = w^4$ . In **Iteration 11**, node  $v^5$  is selected as the relay, updating the parameters of  $v^6$  to  $C(v^6) = 2\text{Mb}$ ,  $L(v^6) = 10\text{ms}$ , and  $P(v^6) = v^5$ . In **Iteration 12**, node  $v^6$  is popped, and its unique outgoing edge  $(v^6, d_2^6)$  is checked, resulting in  $C(d_2^6) = 1\text{Mb}$ ,  $L(d_2^6) = 11\text{ms}$ , and  $P(d_2^6) = v^6$ . In **Iteration 13**, node  $w^5$  with  $C(w^5) = 2\text{Mb}$  is extracted from  $Q$ . It is determined that the “pending-capacity” of  $w^6$  cannot be further improved through cross-cycle transmission along TE  $(w^5, v^5)$  (step 8). In contrast, the parameters of  $w^6$  are updated to  $C(w^6) = 2\text{Mb}$ ,  $L(w^6) = 10\text{ms}$ , and  $P(w^6) = w^5$ . Subsequently,  $w^6$  is selected in **Iteration 14** without triggering any parameter updates for other nodes. In **Iterations 15 and 16**, nodes  $d_1^4$  and  $d_2^6$  are selected, producing the corresponding virtual nodes  $d'_1$  and  $d'_2$  with parameters  $C(d'_1) = 1\text{Mb}$ ,  $L(d'_1) = 7\text{ms}$ ,  $P(d'_1) = d_1^4$ ; and  $C(d'_2) = 1\text{Mb}$ ,

$L(d'_2) = 11\text{ms}$ ,  $P(d'_2) = d'_2$ , respectively. In the **final two iterations**, nodes  $d'_1$  and  $d'_2$  are extracted from  $\mathcal{Q}$ , indicating that their “pending-capacities” have been maximized, and the **search process** terminates.

- **Construction process.** Since both  $C(d'_1) > 0$  and  $C(d'_2) > 0$  are satisfied, the final  $\mathcal{T}$  can be constructed through backtracking (steps 21-31). The resulting tree is  $\mathcal{T} = (s^1, s^2), (s^2, w^2), (w^3, w^4), (w^4, d'_1), (w^4, v^4), (v^5, v^6), (v^6, d'_2)$ , as illustrated in Fig. 1(b).

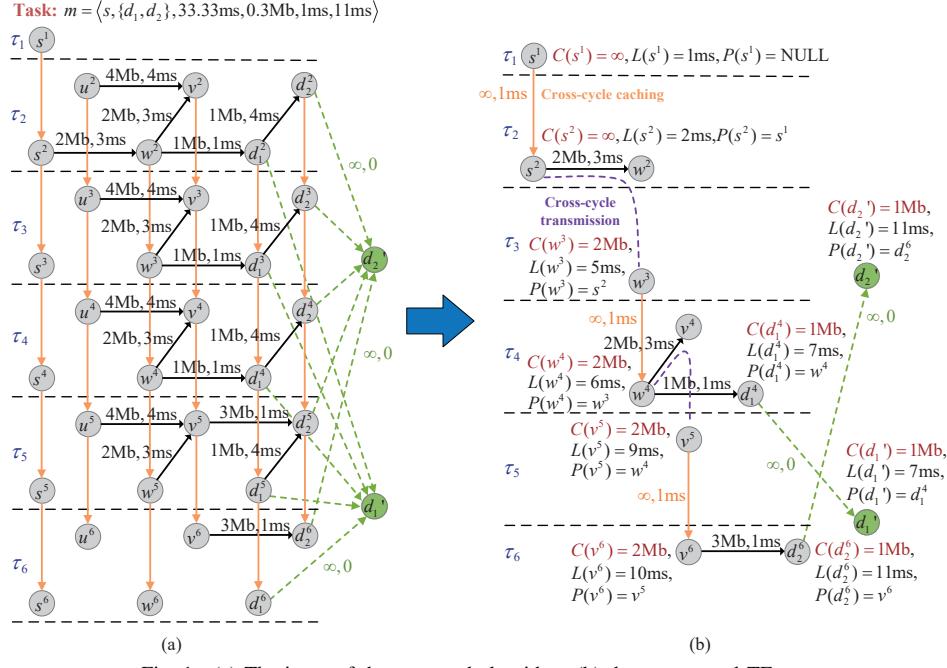


Fig. 1. (a) The input of the proposed algorithm; (b) the constructed TF tree.

## APPENDIX C PROOF OF THEOREM *Theorem 1*

For **Algorithm 1**, the key is to demonstrate that each node extracted from  $\mathcal{Q}$  has determined its maximum “pending-capacity”  $C(\cdot)$ . This assertion remains valid for  $s^h$  with  $C(s^h) = \infty$ . Moving on to the  $K$ -th extracted node, denoted as  $u^h$  (or  $d'_n$ ), we identify that its “pending-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:

- If reaching  $u^h$  via  $w^i \in \mathcal{Q}$  along  $(w^i, u^i) \in \mathcal{E}_t$  enables  $C(u^h) < \min \{C(w^i), c_{w,u}^i\} \leq C(w^i)$ , where  $h = \lceil \frac{1}{|\tau|} \cdot ((i-1) \cdot |\tau| + l_{w,u}) \rceil$ , a contradiction arises because  $C(w^i) \leq C(u^h)$  is enforced by step 4;
- If reaching  $u^h$  via  $u^{h-1} \in \mathcal{Q}$  along  $(u^{h-1}, u^h) \in \mathcal{E}_s$  enables  $C(u^h) < \min \{C(u^{h-1}), c_{u,u}^h\} = C(u^{h-1})$ , a contradiction also arises due to  $C(u^{h-1}) \leq C(u^h)$ ;
- If reaching  $d'_n$  via  $d_n^i \in \mathcal{Q}$  along  $(d_n^i, d'_n) \in \mathcal{E}_v$  enables  $C(d'_n) < \min \{C(d_n^i), c_{d_n,d'_n}^i\} = C(d_n^i)$ , it encounters a contradiction with  $C(d_n^i) \leq C(d'_n)$ .

Therefore, the “pending-capacity” of the  $K$ -th extracted node cannot be improved.

Intuitively, the bottleneck 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 *Theorem 2*

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, examining each edge (i.e., transmission edge, storage edge, or virtual edge) and updating the parameters (i.e., “pending-capacity”  $C(\cdot)$ , “arrival-time”  $L(\cdot)$ , and “pre-node”  $P(\cdot)$ ) of the node actually reached along that edge takes  $O(\log |\mathcal{V}|)$  time. Denoting the out-degree of  $u^h$  as  $\deg(u^h)$ , the time complexity of the process from step 5 to 18 is at most  $O(\deg(u^h) \cdot \log |\mathcal{V}|)$ . 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  $O(\sum_{u^h \in \mathcal{V}} \deg(u^h) \cdot \log |\mathcal{V}|) = O(|\mathcal{E}| \cdot \log |\mathcal{V}|)$ . Additionally, the backtracking process takes at most  $O(|\mathcal{E}|)$  time. Thus, the time complexity of **Algorithm 1** can be calculated as  $O(|\mathcal{V}| + |\mathcal{E}| \cdot \log |\mathcal{V}| + |\mathcal{E}|)$ . Since  $\mathcal{G}$  is a connected graph, it follows that  $|\mathcal{E}| + 1 \geq |\mathcal{V}| \geq 2$  holds, and the time complexity can be further simplified to  $O(|\mathcal{E}| \cdot \log |\mathcal{V}|)$ . The proof is completed.