APPENDIX A
EXAMPLES FOR BLOCKING ANALYSIS

In Section IV and Section V, we presented how to compute the worst-case response time of each task $\tau_i \in \mathcal{T}$ and test the schedulability of $\mathcal{T}$ under POMLP and POMIP. In this appendix, we first use examples to better illustrate the blocking analysis procedure under both POMLP and POMIP, then we compare the properties between these two protocols.

POMLP: We take τ_i and τ_k shown in Fig. 1 as an example to illustrate the blocking analysis under POMLP. All critical sections in both τ_i and τ_k need to access the same resource ℓ_q. The basic parameters of τ_i and τ_k are shown in Table I.

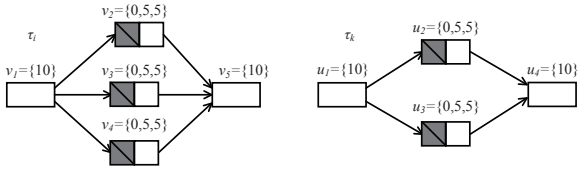

Fig. 1: An example for blocking analysis under POMLP.

TABLE I: Basic task parameters of τ_i and τ_k.

Task	h	$\mathcal{C}_h$	$\mathcal{L}_h$	$N_{h,q}$	$L_{h,q}$	D_h	T_h
τ_i	i	50	30	3	5	60	90
τ_k	k	40	30	2	5	60	70

Based on Equation (11) and Algorithm 1 in Section V-A, the procedure of the schedulability-test for τ_i and τ_k is shown in Table II. As for the computation of $\mathcal{F}^I(N^\lambda_{i,q})$ and $\mathcal{F}^O(N^\lambda_{i,q})$ in Equation (11), we directly use Equation (2) and Equation (3) in Section V-A, respectively.

In the following, we give the detailed computation procedure for the response time of τ_i and τ_k in the iterations.

- **Iteration 1:** Initially, we set $m_i = \lceil \frac{50-30}{60-30} \rceil = 1$ and $m_k = \lceil \frac{40-30}{60-30} \rceil = 1$; And $\mathcal{R}_i$ and $\mathcal{R}_k$ can be computed by Equation (11) with $m_i = 1$ and $m_k = 1$.
  For $\mathcal{R}_i$ computation, we get $\alpha = 1$ and $\eta^q_{i,k} = \lceil \frac{60+60}{70} \rceil \times 2 = 4$, and we need to enumerate $N^\lambda_{i,q} \in [0, 3]$ to obtain the maximum value of the sum of $\mathcal{F}^I(N^\lambda_{i,q})$ and $\mathcal{F}^O(N^\lambda_{i,q})$.
  - When $N^\lambda_{i,q} = 0$:
    By Equation (2), we get $\mathcal{F}^I(N^\lambda_{i,q}) = 0$;
    By Equation (3), we get $\mathcal{F}^O(N^\lambda_{i,q}) = 15$;
    Hence, $\sum_{\ell_q \in \Theta_i} \{\mathcal{F}^I(N^\lambda_{i,q}) + \mathcal{F}^O(N^\lambda_{i,q})\} = 15$
  - Similarly, when $N^\lambda_{i,q} = 1$ or $N^\lambda_{i,q} = 2$ or $N^\lambda_{i,q} = 3$:
    We have $\mathcal{F}^I(N^\lambda_{i,q}) = 0$ and $\mathcal{F}^O(N^\lambda_{i,q}) = 15$;

  Hence, in Iteration 1,

$$\sum_{\ell_q \in \Theta_i} \max_{N^\lambda_{i,q} \in [0, N_{i,q}]} \{\mathcal{F}^I(N^\lambda_{i,q}) + \mathcal{F}^O(N^\lambda_{i,q})\} = 15$$

  By Equation (11), we have $\mathcal{R}_i = \frac{50+0\times30+15}{1} = 65 \geq D_i$. Therefore, we assign one more processor to τ_i, i.e., $m_i \leftarrow 2$.
  For $\mathcal{R}_k$ computation, we get $\alpha = 1$ and $\eta^q_{k,i} = \lceil \frac{60+60}{90} \rceil \times 3 = 6$, and we need to enumerate $N^\lambda_{k,q} \in [0, 2]$ to obtain the maximum value of the sum of $\mathcal{F}^I(N^\lambda_{k,q})$ and $\mathcal{F}^O(N^\lambda_{k,q})$.
  - When $N^\lambda_{k,q} = 0$:
    By Equation (2), we get $\mathcal{F}^I(N^\lambda_{k,q}) = 0$;
    By Equation (3), we get $\mathcal{F}^O(N^\lambda_{k,q}) = 10$;
    Hence, $\sum_{\ell_q \in \Theta_k} \{\mathcal{F}^I(N^\lambda_{k,q}) + \mathcal{F}^O(N^\lambda_{k,q})\} = 10$
  - Similarly, when $N^\lambda_{k,q} = 1$ or $N^\lambda_{k,q} = 2$:
    We have $\mathcal{F}^I(N^\lambda_{k,q}) = 0$ and $\mathcal{F}^O(N^\lambda_{k,q}) = 10$;

  Hence, in Iteration 1,

$$\sum_{\ell_q \in \Theta_k} \max_{N^\lambda_{k,q} \in [0, N_{k,q}]} \{\mathcal{F}^I(N^\lambda_{k,q}) + \mathcal{F}^O(N^\lambda_{k,q})\} = 10$$

  By Equation (11), we have $\mathcal{R}_k = \frac{40+0\times30+10}{1} = 50 \leq D_k$. τ_k is deemed as schedulable by Algorithm 1, so no extra processors need to be assigned to its cluster.
- **Iteration 2:** By the end of Iteration 1, the numbers of processors have been updated as $m_i = 2$ and $m_k = 1$;
  For $\mathcal{R}_i$ computation, we get $\alpha = 2$ and $\eta^q_{i,k} = 4$, and
  - When $N^\lambda_{i,q} = 0$:
    By Equation (2), we get $\mathcal{F}^I(N^\lambda_{i,q}) = 10$;
    By Equation (3), we get $\mathcal{F}^O(N^\lambda_{i,q}) = 15$;
    Hence, $\sum_{\ell_q \in \Theta_i} \{\mathcal{F}^I(N^\lambda_{i,q}) + \mathcal{F}^O(N^\lambda_{i,q})\} = 25$
  - Similarly, when $N^\lambda_{i,q} = 1$:
    We have $\mathcal{F}^I(N^\lambda_{i,q}) = 10$ and $\mathcal{F}^O(N^\lambda_{i,q}) = 20$;
  - When $N^\lambda_{i,q} = 2$: $\mathcal{F}^I(N^\lambda_{i,q}) = 5$ and $\mathcal{F}^O(N^\lambda_{i,q}) = 25$;
  - When $N^\lambda_{i,q} = 3$: $\mathcal{F}^I(N^\lambda_{i,q}) = 0$ and $\mathcal{F}^O(N^\lambda_{i,q}) = 30$;

  In conclusion, in Iteration 2,

$$\sum_{\ell_q \in \Theta_i} \max_{N^\lambda_{i,q} \in [0, N_{i,q}]} \{\mathcal{F}^I(N^\lambda_{i,q}) + \mathcal{F}^O(N^\lambda_{i,q})\} = 30$$

  By Equation (11), we have $\mathcal{R}_i = \frac{50+1\times30+30}{2} = 55 \leq D_i$. τ_i is deemed as schedulable by Algorithm 1, so no extra processors need to be assigned to its cluster.
  For $\mathcal{R}_k$ computation, we get $\alpha = 1$ and $\eta^q_{k,i} = 6$, and
  - When $N^\lambda_{k,q} = 0$:
    By Equation (2), we get $\mathcal{F}^I(N^\lambda_{k,q}) = 0$;
    By Equation (3), we get $\mathcal{F}^O(N^\lambda_{k,q}) = 20$;
    Hence, $\sum_{\ell_q \in \Theta_k} \{\mathcal{F}^I(N^\lambda_{k,q}) + \mathcal{F}^O(N^\lambda_{k,q})\} = 20$
  - Similarly, when $N^\lambda_{k,q} = 1$ or $N^\lambda_{k,q} = 2$:
    We have $\mathcal{F}^I(N^\lambda_{k,q}) = 0$ and $\mathcal{F}^O(N^\lambda_{k,q}) = 20$;

  Hence, in Iteration 2,

$$\sum_{\ell_q \in \Theta_k} \max_{N^\lambda_{k,q} \in [0, N_{k,q}]} \{\mathcal{F}^I(N^\lambda_{k,q}) + \mathcal{F}^O(N^\lambda_{k,q})\} = 20$$

  By Equation (11), we have $\mathcal{R}_k = \frac{40+0\times30+20}{1} = 60 \leq D_k$. τ_k is also deemed as schedulable, i.e., the whole task set is schedulable. So the iteration procedure stops.

TABLE II: Blocking analysis procedure of τ_i and τ_k.

	Iteration	m_i	$\mathcal{F}^I(N^\lambda_{i,q})$	$\mathcal{F}^O(N^\lambda_{i,q})$	$\mathcal{R}_i$	$R_i \leq D_i$?
	1	1	0	15	65	×
τ_i	2	2	10	20	55	√
	Iteration	m_k	$\mathcal{F}^I(N^\lambda_{k,q})$	$\mathcal{F}^O(N^\lambda_{k,q})$	$\mathcal{R}_k$	$R_k \leq D_k$?
	1	1	0	10	50	√
τ_k	2	1	0	20	60	√

In conclusion, by our schedulability-test in Algorithm 1, a platform with no fewer than 3 processors could guarantee the task set in Fig. 1 to be schedulable under POMLP.

POMIP: We take the two tasks τ_i and τ_k in Fig. 2 as an example to illustrate the blocking analysis under POMIP. Critical sections in v_2 and v_3 of τ_i need to access resources ℓ_1

and ℓ_2 respectively. Critical sections in v_4 of τ_i and u_2, u_3 of τ_k need to access resource ℓ_3 (as marked on the bottom left corner of these critical sections). The basic task parameters of τ_i and τ_k are shown in Table III.

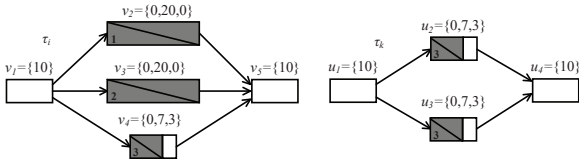

Fig. 2: An example for blocking analysis under POMIP.

TABLE III: Basic task parameters of τ_i and τ_k.

Task	h	$\mathcal{C}_h$	$\mathcal{L}_h$	$N_{h,q}$	$L_{h,q}$	D_h	T_h
τ_i	i	70	40	{1,1,1}	{20,20,7}	75	90
τ_k	k	40	30	{0,0,2}	{0,0,7}	52	70

Based on Equation (11) and Algorithm 1 in Section V-A, the procedure of the schedulability-test for τ_i and τ_k is shown in Table IV. As for the computation of $\mathcal{F}^I(N^\lambda_{i,q})$ and $\mathcal{F}^O(N^\lambda_{i,q})$ in Equation (11), we directly use Equation (7) and Equation (8) in Section V-B, respectively. Since the computation procedure for the response time of τ_i and τ_k in the iterations is basically the same as it is under POMLP except with different equations, we do not give the detailed computation this time.

TABLE IV: Blocking analysis procedure of τ_i and τ_k.

	Iteration	m_i	$\mathcal{F}^I(N^\lambda_{i,q})$	$\mathcal{F}^O(N^\lambda_{i,q})$	$\mathcal{R}_i$	$R_i \leq D_i$?
τ_i	1	1	{0,0,0}	{0,0,28}	98	$\times$
	2	2	{0,0,0}	{0,0,35}	72.5	$\surd$
	3	2	{0,0,0}	{0,0,35}	72.5	$\surd$
	Iteration	m_k	$\mathcal{F}^I(N^\lambda_{k,q})$	$\mathcal{F}^O(N^\lambda_{k,q})$	$\mathcal{R}_k$	$R_k \leq D_k$?
τ_k	1	1	{0,0,0}	{0,0,14}	54	$\times$
	2	2	{0,0,7}	{0,0,28}	52.5	$\times$
	3	3	{0,0,14}	{0,0,42}	52	$\surd$

In conclusion, by our schedulability-test in Algorithm 1, a platform with no fewer than 5 processors could guarantee the task set in Fig. 2 to be schedulable under POMIP.

Differences between the properties of the two protocols:

The properties of POMLP and POMIP are introduced in the beginning of Section V-A and Section V-B, respectively. In this appendix, we compare the properties of these two protocols to highlight the differences between them.

- POMIP allows a lock-holding job to migrate to other clusters for executing when it can not be scheduled on its own cluster and some requests are blocked by it. While under POMLP, migrations among clusters are not allowed.
- The second difference is that POMLP guarantees that the number of simultaneously issued requests to all resources from a cluster with m_i processors is upper bounded by m_i, while under POMIP the vertices can freely issue their requests. Hence, a request on the key path can be blocked by requests for other resources under POMLP, which is not the case under POMIP.
- Another difference is that POMLP guarantees that a lock-holding job is always scheduled, but when a request for ℓ_q is holding the lock under POMIP, it may not be scheduled. This means that requests in POMIP would suffer the ready-waiting time of lock-holding requests. Nevertheless, the problem is relived by the migrational mechanism under POMIP with which a task can not be interfered by requests for resources it does not access.