Supplementary Information for "A Joint offloading and Energy Cooperation Scheme for Edge Computing Networks"

Jieyi Zhang[1], BilingZhang[1]*, Jiahua Liu[1], Zhu Han[2]
1. School of Network Education, Beijing University of Posts and Telecommunications, P. R. China
2. Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA

1) Computation offloading strategy

Lemma 1.1 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j \geq 0 \end{cases}$, there is $\tau_{i\max} \geq \tau_j$, the offload strategy is

$$\begin{cases} o_i^{ij} = \min\left\{ \frac{Re_j}{K(V_j)^2} - o_j, (\tau_{i\max} - \tau_j) \cdot V_j, o_i - \frac{Re_i}{K(V_i)^2} \right\} \\ o_i^l = o_i - o_i^{ij} \\ o_j^l = o_j \end{cases}$$

when $\tau_{i\max} < \tau_j$, $o_i^l = o_i, o_j^l = o_j$.

Proof: In this case, both BS_i and BS_j have the ability to complete all calculation tasks independently. According to Proposition 1, after consuming Re_i, use BS_j to calculate, and consume as much Re_j as possible, that is, to split the data to more Re side.

When $\tau_{i\max} \geq \tau_j$, BS_j can accept the offloading of BS_i. There are three uninstall strategies, the minimum value of the three is o_i^{ij}:

① EC_i helps j calculate until Re_j is used up, from this, it can be follow as $Re_j - K(V_j)^2 o_j = K(V_j)^2 o_i^{ij'}$, simplifying to get $o_i^{ij'} = \frac{Re_j}{K(V_j)^2} - o_j$.

② Regardless of Re, before the maximum delay limit of the o_i calculation task is reached, the amount of tasks that EC_j can help EC_i calculation can be written as $o_i^{ij''} = (\tau_{i\max} - \tau_j) \cdot V_j$.

③ If the Re_i is consumed, all the remaining calculation tasks are completed at EC_j. At this time, the carrying capacity from EC_i to EC_j is $o_i^{ij'''} = o_i - \frac{Re_i}{K(V_i)^2}$.

When $\tau_{i\max} < \tau_j$, EC_j can't help EC_i perform the calculation, all calculation tasks are done locally. ■

Lemma 1.2 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j < 0 \end{cases}$, we have $\begin{cases} o_i = o_i^l \\ o_j = o_j^l \end{cases}$

Proof: Both i and j lack of energy. At this point the calculation task is done locally and the analysis is done in the same way as Lemma 1.1. ■

Lemma 1.3 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j \geq 0 \end{cases}$, we have $\begin{cases} o_i = o_i^l \\ o_j = o_j^l \end{cases}$

Proof: In this case, i and j can both complete the computational tasks independently under the latency constraints, in order to reduce transmission loss, the computational tasks are prioritized to be completed locally. So we have $o_k = o_k^l$ ■

Lemma 1.4 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j < 0 \end{cases}$, simply swap i and j, and the method of analysis is the same as Lemma 1.3.

Proof: Replace i and j, the method of proof is the same as 1.3. ■

Lemma 1.5 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j \geq 0 \end{cases}$.

(1) $\tau_{imax} \geq \tau_{jmax}$. The set of equation can be written as

$$\begin{cases} o_i^l = \tau_{i\max} \cdot V_i \\ o_i^{ij} = \min\{(\tau_{imax} - \tau_{j\max}) \cdot V_j, \quad o_i - o_i^l\} \\ o_i^{iC} = o_i - (o_i^l + o_i^{ij}) \end{cases}$$

$$\begin{cases} o_j^l = \tau_{jmax} \cdot V_j \\ o_j^{jC} = o_j - o_j^l \\ o_j^{ji} = 0 \end{cases}$$

(2) $\tau_{imax} < \tau_{jmax}$. Simply swap i and j, and the method of analysis is the same as Lemma 1.5.

Proof: Neither i and j can complete the computational task under the experimental constraints, but since the computational tasks of i, j have different time delays, the time delay difference can be used for MEC offloading and then offloading to the cloud. Because i and j are unable to complete the local task, Le'_i does not affect the policy at this time.

When $\tau_{imax} \geq \tau_{jmax}$,At this point, the time delay (maximum) of i is greater than that of j, j completes part of the task of local o_j before the maximum time delay arrives. Since the local computation task without j after τ_{jmax} is finished, its computation resources are used to offload computation task of i to j. The offloading amount is $o_i - o_i^l$ when it represents i, j collaboration can be completed. When unloading to C, the unloading amount can be given as $o_i^{ij''} = (\tau_{imax} - \tau_{jmax}) * V_j$. When $\tau_{imax} < \tau_{jmax}$, simply swap i and j, and the method of proof is the same as Lemma 1.5. ■

Lemma 1.6 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j < 0 \end{cases}$, the result is the same as 1.5.

Proof: At this case, the analysis is done in the same way as Lemma 1.5. ■

Lemma 1.7 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j \geq 0 \end{cases}$, the result is the same as 1.5.

Proof: At this case, the analysis is done in the same way as Lemma 1.5. ■

Lemma 1.8 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j < 0 \end{cases}$, the result is the same as 1.5.

Proof: At this case, the analysis is done in the same way as Lemma 1.5. ■

Lemma 1.9 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j \geq 0 \end{cases}$,

(1) $\tau_{imax} \geq \tau'_j$. The set of equation can be written as

$$\begin{cases} o_i^l = \tau_{imax} \cdot V_i \\ o_i^{ij} = \min\left\{(\tau_{imax} - \tau_j) \cdot V_j, o_i - o_i^l\right\} \\ o_i^{iC} = o_i - \left(o_i^l + o_i^{ij}\right) \\ o_j^l = o_j \end{cases}$$

(2) When $\tau_{imax} < \tau'_j$, the set of equation can be given as

$$\begin{cases} o_i^l = \tau_{imax} \cdot V_i \\ o_i^{iC} = o_i - o_i^l \\ o_i^{ij} = 0 \\ o_j^l = o_j \end{cases}$$

Proof: In this case, i unable to complete all tasks of o_i under time delay constraints. When $\tau_{imax} \geq \tau'_j$, after i offloads to j, if still remaining, then offloads to C cloud. When $\tau_{imax} < \tau'_j$, i unable to offload to j, offloads to C cloud. ■

Lemma 1.10 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j < 0 \end{cases}$, The result is the same as 1.9 *Proof*: In this case, i unable to complete all tasks of o_i under time delay constraints. When $\tau_{imax} \geq \tau'_j$, after i offloads to j, if still remaining, then offloads to C cloud. When $\tau_{imax} < \tau'_j$, i unable to offload to j, offloads to C cloud.The proof method is the same as 1.9. ■

Lemma 1.11 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j \geq 0 \end{cases}$, it follows that:

(1) $\tau_{imax} \geq \tau'_j$. It can occur unload from i to j.

1) If $o_i \geq (\tau_{imax} - \tau_j) \cdot V_j + \tau_{imax} \cdot V_i$., the set of equation can be formulated as

$$\begin{cases} o_i^l = \tau_{imax} \cdot V_i \\ o_i^{ij} = (\tau_{imax} - \tau_j) \cdot V_j \\ o_i^{iC} = o_i - \left(o_i^{ij} + o_i^l\right) \\ o_j^l = o_j \end{cases}$$

2) $o_i < (\tau_{imax} - \tau_j) \cdot V_j + \tau_{imax} \cdot V_i$, the set of equation can be written as

$$\begin{cases} o_i^{ij} = \max(o_i - \tau_{imax} \cdot V_i), \min\left\{\dfrac{Re_j}{K(V_j)^2} - o_j \right. \\ \left. (\tau_{imax} - \tau_j) \cdot V_j, o_i - \dfrac{Re_i}{K(V_i)^2}\right\} \\ o_i^l = o_i - o_i^{ij} \\ o_j^l = o_j \end{cases}$$

(2) When $\tau_{imax} < \tau'_j$, the set of equation can be given as

$$\begin{cases} o_i^l = \tau_{imax} \cdot V_i \\ o_i^{ij} = 0 \\ o_i^{iC} = o_i - o_i^l \\ o_j^l = o_j \end{cases}$$

Proof: In this case, i can neither possible to complete o_i task nor lack Re.

When $\tau_{imax} \geq \tau'_j$, it can occur unload from i to j. Under the above conditions, when $o_i \geq (\tau_{imax} - \tau_j) \cdot V_j + \tau_{imax} \cdot V_i$, that is, before τ_{imax} arrives, the total task volume of i and j collaboration is less than o_i and cannot complete the computation task under the latency constraint, it needs to be offloaded to the C cloud.

If $o_i < (\tau_{imax} - \tau_j) \cdot V_j + \tau_{imax} \cdot V_i$, task volume of i and j collaboration can reach o_i. At this point it is known that EC_j has both excess computing power and excess energy. Therefore, it follows from Theorem 1 that in Lemma of exhaustion of Re_i, more energy of Re_j is used. If Re_j energy is exhausted, the overdraft energy calculation is given priority in the original MEC_i in order to reduce the transmission (information) energy consumption, and the overdraft energy replenishment and adjustment will be expressed in the energy strategy, and the content representation is the same as Lemma 1.1.

When $\tau_{imax} < \tau'_j$, it does occur unloaded between i and j. ■

Lemma 12. For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} < \tau'_i \\ \tau_{jmax} \geq \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j < 0 \end{cases}$, we can get the conclusion:

(1) If $\tau_{imax} \geq \tau'_j$, the set of the equation can be written as

$$\begin{cases} o_i^l = \tau_{imax} \cdot V_i \\ o_i^{ij} = \min\left\{o_i - o_i^l, (\tau_{imax} - \tau_j) \cdot V_j\right\} \\ o_i^{iC} = o_i - \left(o_i^l + o_i^{ij}\right) \\ o_j^l = o_j \end{cases}$$

(2) If $\tau_{imax} < \tau'_j$, the set of the equation can be written as

$$\begin{cases} o_i^l = \tau_{imax} \cdot V_i \\ o_i^{ij} = 0 \\ o_i^{iC} = o_i - o_i^l \\ o_j^{ji} = \min\left\{\frac{Re_i}{K(V_i)^2} - o_i^2, \quad (\tau_{jmax} - \tau_{imax}) \cdot V_i\right\} \\ o_j^l = o_j - o_j^{ji} \end{cases}$$

Proof: In this case, MEC_i unable to complete o_i task, however EC_j can complete, but energy of j is not enough. If $\tau_{imax} \geq \tau'_j$, priority is given to EC offloading between calculations and offloading without considering Le'. If $\tau_{imax} < \tau'_j$, from Theorem 1, it follows that i unable to perform computational offloading to j, but since i has more Re_i, j unload to i (reverse unload). ■

Lemma 1.13 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j \geq 0 \end{cases}$, replacing i and j, the result is the same as 1.9.

Proof: Replacing i and j, the method of proof is the same as 1.9. ■

Lemma 1.14 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j < 0 \end{cases}$, replacing i and j, the result is the same as 1.10.

Proof: Replacing i and j, the method of proof is the same as 1.10. ■

Lemma 1.15 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i < 0 \\ Le'_j \geq 0 \end{cases}$, replacing i and j, the result is the same as 1.11.

Proof: Replacing i and j, the method of proof is the same as 1.11. ■

Lemma 1.16 For any o_k , τ_{kmax} and Le'_j , when $\begin{cases} \tau_{imax} \geq \tau'_i \\ \tau_{jmax} < \tau'_j \end{cases}$ and $\begin{cases} Le'_i \geq 0 \\ Le'_j < 0 \end{cases}$, replacing i and j, the result is the same as 1.12.

Proof: Replacing i and j, the method of proof is the same as 1.12. ■

Summarizing the above we can obtain the optimal strategy for calculating the unloading. From section II, we can obtain $Ce(\boldsymbol{o}(\boldsymbol{t}))$.

2). Energy collaboration strategy

Due to the many changes in energy unloading, in order to make the description clearer, some iterative expressions cannot be omitted.

Lemma 2.1. According to the result of **1)**, if $Le_i \geq 0, Le_j \geq 0$, then $W_k = 0$ $(k \in (i,j))$,

$$m_k = \min\left\{\frac{\left(S_{\max} - s_k\right), Le_k}{\delta}\right\},$$

and when $s_i = S_{max}, s_j = S_{max}$ or $s_i, s_j < S_{max}$ or $s_j < S_{max},\ s_i = S_{max}$, there is

$$s_k \leftarrow s_k + m_k,$$

and when $s_i < S_{max},\ s_j = S_{max}$, there is

$$s_i \leftarrow s_i + \min\left(\left(S_{max} - s_i - m_i\right),\ \theta\left(Le_i - \frac{m_i}{\delta}\right)\right)$$

.

Proof: In this case, in accordance with the principle of maximum storage, the remaining energy is stored locally first, i.e., $m_k = \frac{Le_k}{\delta}$, and if the local energy storage is full, i.e., $Le_k \geq S_{max} - s_k$, the excess energy will be transferred to the storage of another base station as $x_{ji} = Le_j - m_j$. The storage capacity of j, i.e., $m'_j = \min\left\{\frac{\theta x_{ij}, (S_{\max} - s_j)}{\delta}\right\}$, according to the (13) we can get s_i. ■

Lemma 2.2 According to the result of **1)**, if $Le_i \geq 0, Le_j < 0$, we can get the result

(1) If $Le_i \geq 0, Le_j < 0$ and $\theta \geq \delta^2$, BS_i first transmits net energy to BS_j to compensate for the defect. Therefore, the equation can be set as

$$x_{ij} = \min\left\{Le_i, \frac{|Le_j|}{\theta}\right\},$$
$$Le'_j = Le_j + \theta x_{ij}.$$

Now, if $Le'_j = 0$, $m_i = Le'_i = Le_i - x_{ij}$ then $s_i \leftarrow s_i + \delta m_i, s_j \leftarrow s_j, w_i = w_j = 0$.

If $Le'_j < 0$ and $Le''_j = 0$

$$n_j = \min\left\{\frac{\left|Le'_j\right|}{\delta}, s_j\right\},$$
$$Le''_j = Le'_j + \delta n_j,$$
$$s_i \leftarrow s_i, s_j \leftarrow s_j - n_j, w_i = w_j = 0.$$

If $Le'_j < 0$ and $Le''_j \neq 0$,

$$n_i = \min\left\{\frac{\left|Le''_j\right|}{(\delta\theta)}, s_i\right\},$$
$$x_{ij} \leftarrow x_{ij} + \delta n_i,$$
$$Le'''_j = Le''_j + \delta\theta n_i.$$

If $\left(Le_j''' >= 0\right)$, there is $s_i \leftarrow s_i - n_i, s_j = 0, w_i = w_j = 0$.

If $\left(Le_j''' < 0\right)$, $s_i = s_j = 0, w_i = 0, w_j = \left|Le_j'''\right|$.

(2) $Le_i \geq 0$, $Le_j < 0$ and $\theta < \delta^2$. In this Lemma, the BS_i tries to use excess energy to maximize its own storage level to minimize the energy required to compensate for the BS_j. The best optimal strategy identified in the following two sub-scenarios can be written as

1. $|Le_j| \geq \theta Le_i + \delta s_j$ The best strategy is the same as **(1)**.
2. $|Le_j| < \theta Le_i + \delta s_j$ The best strategy is given below

$$
\begin{aligned}
&x_{ij} = \max\left\{\frac{|Le_j| - \delta s_j}{\theta}, Le_i - \frac{S_{max} - s_i}{\delta}, 0\right\},\\
&m_i = Le_i - x_{ij},\\
&n_j = \max\left\{\frac{|Le_j| - \theta x_{ij}}{\delta}, 0\right\},\\
&m_j = \min\left\{\frac{S_{max} - s_j}{\delta}, \theta x_{ij} - |Le_j|\right\},\\
&s_i \leftarrow s_i + \delta m_i,\\
&s_j \leftarrow s_j + \delta m_j - n_j, w_i = w_j = 0.
\end{aligned}
$$

Proof:If $Le_j' < 0$, we first make up the remaining deficit by storing on BS_j. If $Le_j'' = 0$, it means that this situation has been completed. otherwise, we will compensate from the storage in BS_i.If a deficit is left $\left(Le_j''' < 0\right)$, we compensate and set up by traditional energy consumption.According to [5], the above results can be easily proved. ■

Lemma 2.3 $Le_i < 0, Le_j < 0$ and $\theta \geq \delta^2$. This Lemma is the symmetric Lemma of Lemma 2.2, and is a role reversal of BS_i and BS_j. Therefore, we omit the description of the algorithm here.

Proof:Exchange i and j, the proof of this case is the same as 2.2. ■

Lemma 2.4 $Le_i < 0, Le_j < 0$. In this Lemma, before helping another one, each BS first uses a separate store to compensate. That is, for $k \in \{i, j\}$, the equation can be set as

$$
\begin{aligned}
&n_k = \min\left\{s_k, \frac{|Le_k|}{\delta}\right\},\\
&Le_k' = Le_k + \delta n_k,\\
&s_k \leftarrow s_k - n_k.
\end{aligned}
$$

If it is $Le_i' > 0$ or $Le_j' > 0$, analyzed by the first three cases. If $Le_i' < 0$, we compensate by conventional energy and set as $w_i = |Le_k'|$.

Proof:In this case, the shortage of $Re_k t, k \in (i, j)$ is first replenished through storage, and then lemma is redistributed. If the result of the replenishment is still less than 0, i.e., $Le_i' < 0$, the power shortage will be compensated from the grid. The similar proof method has been described in [5], and we can easily prove the result. ■

The amount of energy stored at the end of this time slot will be the initial value of energy stored in the next time slot.