TEMPLE UNIVERSITY

中国科学技术大学 University of Science and Technology of China

Multi-Task Assignment for CrowdSensing in Mobile Social Networks

Mingjun Xiao[a], Jie Wu[b], Liusheng Huang[a]
Yunsheng Wang[c], Cong Liu[d]

[a] University of Science and Technology of China,
[b] Temple University,
[c] Kettering University,
[d] Sun Yat-sen University

Outline

- **Background & Motivation**
- **Model & Problem**
- **Solution**
- **Simulation**
- **Conclusion**

Background

- **Mobile CrowdSensing**
  - **Smart devices:**

    iPads, smart phones, portable devices, etc.

    (light sensor, GPS, camera, digital compass, etc.)
  - **A new paradigm:**

    mobile users exploit their carried smart devices to conduct complex computation and sensing tasks.
  - **Applications:**

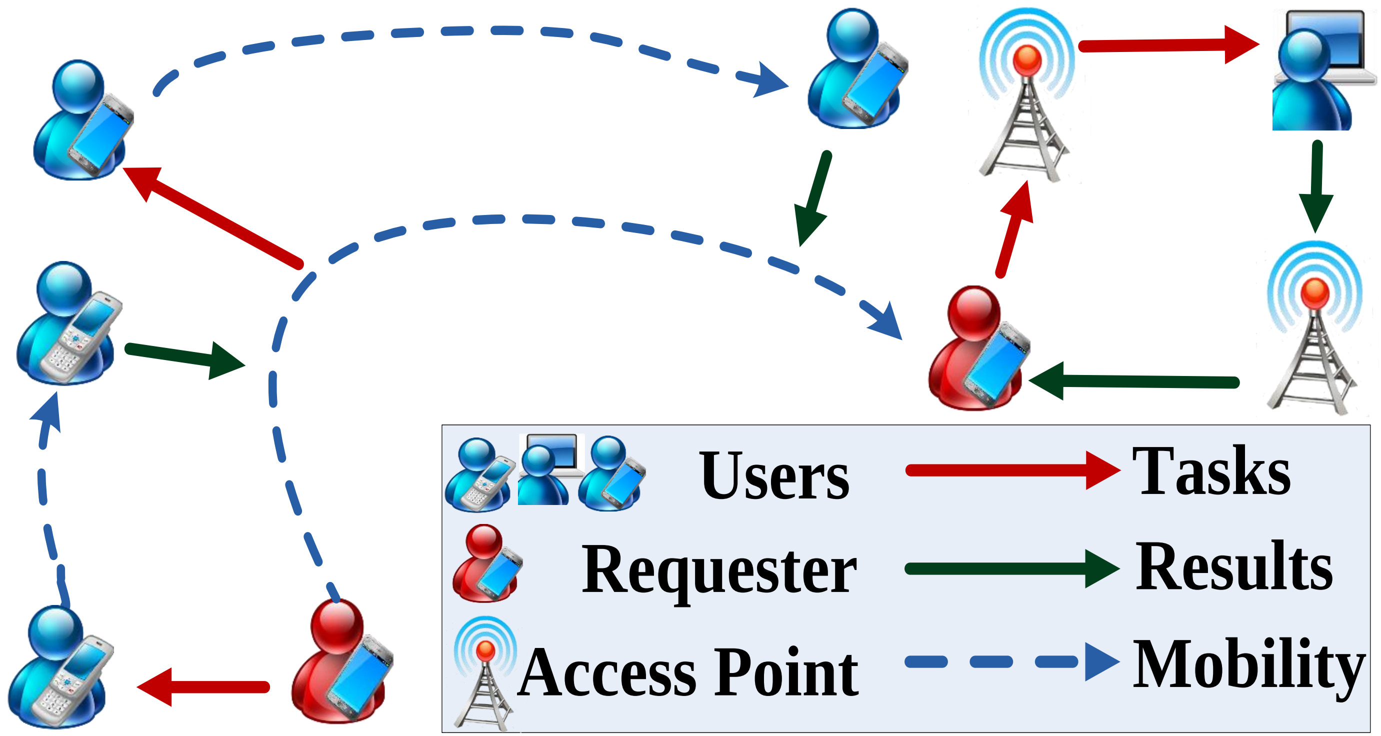
    urban WiFi characterization, map labelling, traffic information mapping, etc.

Motivation

- **CrowdSensing in Mobile Social Networks**
  - **Mobile Social Network (MSN):**
    - 3G/4G communication model
    - Short-distance communication model (WiFi, Bluetooth)
- **Existing CrowdSensing Algorithms**
  - **Task assignment**
    - LRBA [Infocom'14]
    - Fair energy-efficient allocation algorithm [Infocom'14]
  - **Other issues**
    - Incentive mechanisms, Privacy-preserving schemes,…

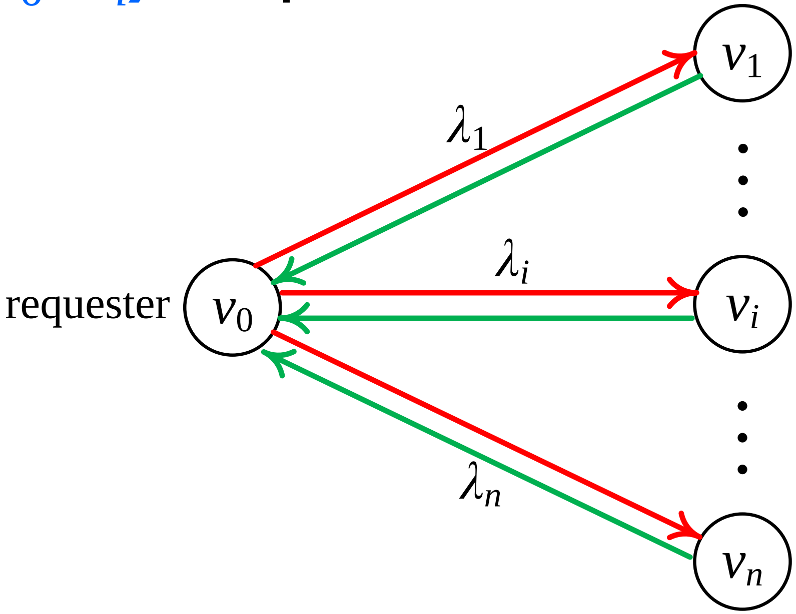
Motivation

- **Time-sensitive task assignment**



Model & Problem

- **Network Model**
  - **MSN users:** $V=\{v_0, v_1, \ldots, v_n\}$
  - **Requester:** v_0
  - **Communication model:** WiFi, Bluetooth
  - **Contact (v_0, v_i):** exponential distribution λ_i



Model & Problem

- **Problem**
  - **Tasks:** $J=\{ j_1, j_2, \ldots , j_m\}$
  - **Workloads:** $\tau_1, \tau_2, \ldots , \tau_m$
  - **Makespan M(j):** the time that the requester finally receives the result of the task j.
  - **Task assignment strategy:**

$$\Pi=\{J_1, J_2, \ldots , J_n\}$$

$J_i=\{\ldots, j, \ldots , j', \ldots\}$ is an ordered subset of J

$j \le j'$: j will be processed prior to j'

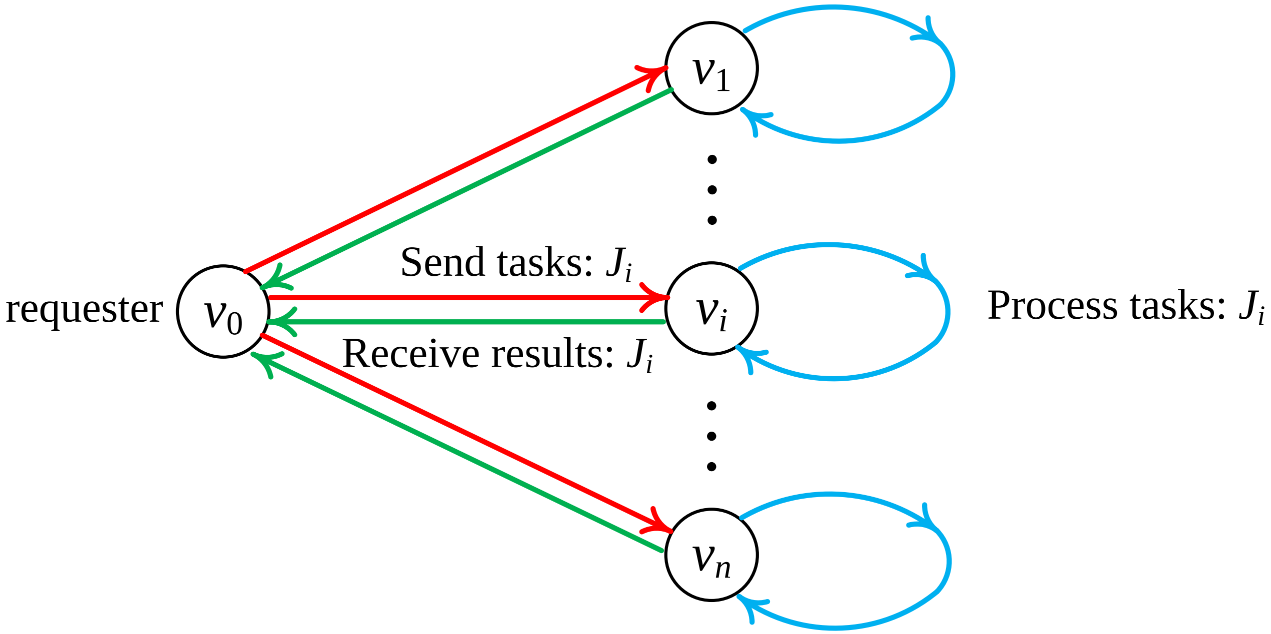
$$\sum_{i=1}^{n} J_i = J,\ \ J_i \cap J_{i'} = \varnothing,\ \forall J_i, J_{i'} \in \Pi$$

Model & Problem

- **Problem**
  - **Average Makespan $AM(\Pi)$:**

$$AM(\Pi) = \frac{1}{m} \sum_{j \in J} M(j)|_{\Pi}$$

  - **Objective: Maximize $AM(\Pi)$**



Solution: oFfline Task Assignment

- **FTA: oFfline Task Assignment**

  the requester makes the task assignment decision before it encounters any other mobile user

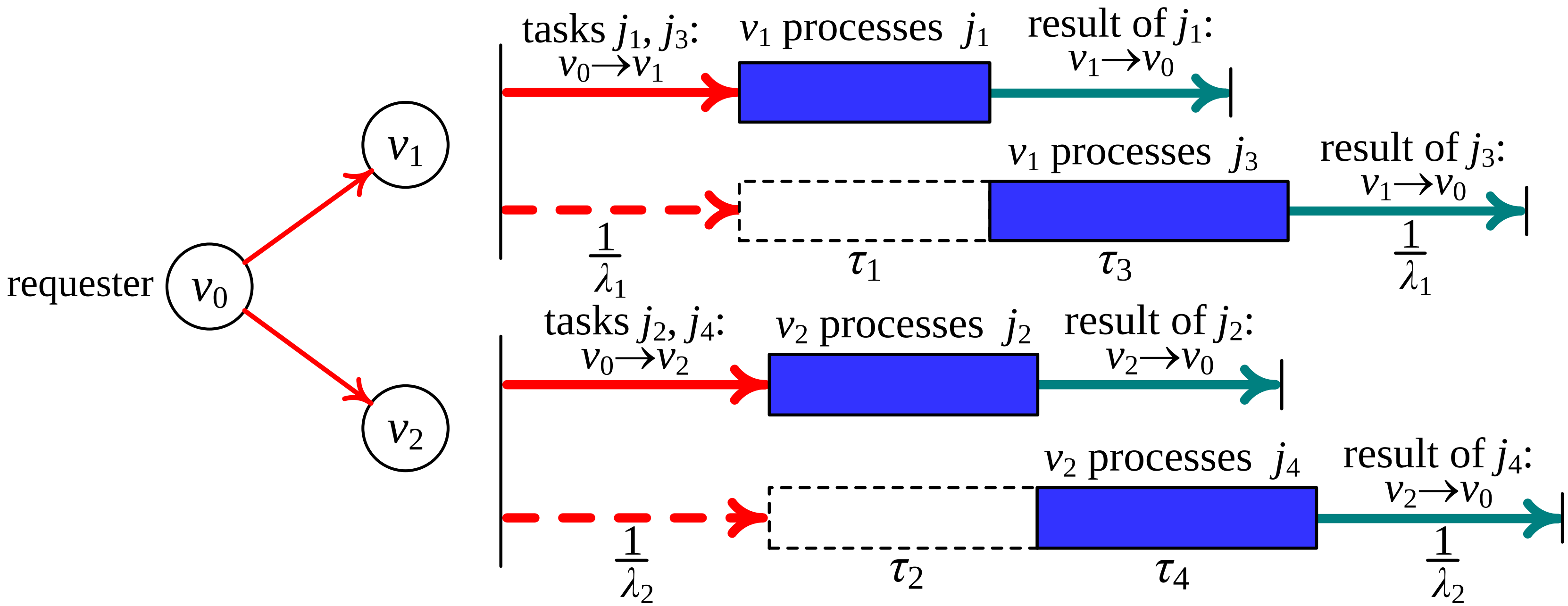
- Basic Formula

  Theorem 1: The average makespan $AM(\Pi_F)$ for an arbitrary offline task assignment strategy $\Pi_F = \{J_1, J_2, \ldots, J_n\}$ satisfies:

$$AM(\prod_F) = \frac{1}{m}\sum_{i=1}^{n}\sum_{j\in J_i}\left(\frac{2}{\lambda_i} + \sum_{j'\in J_i \wedge j' \leq j}\tau_{j'}\right)$$

Solution: oFfline Task Assignment

- **Example:** $\Pi_F = \{J_1, J_2\}$, $J_1 = \{j_1, j_3\}$, $J_2 = \{j_2, j_4\}$



$$M(j_1) = \frac{2}{\lambda_1} + \tau_1, M(j_2) = \frac{2}{\lambda_1} + \tau_1 + \tau_3, M(j_3) = \frac{2}{\lambda_2} + \tau_2, M(j_4) = \frac{2}{\lambda_2} + \tau_2 + \tau_4$$

$$AM(\Pi_F) = \frac{1}{4}(M(j_1) + M(j_2) + M(j_3) + M(j_4))$$

Solution: oFfline Task Assignment

- Basic Property of Optimal FTA Solution: $\boldsymbol{\Pi^*_F}$
  - **Theorem 2:** Suppose the optimal offline task assignment strategy is $\Pi^*_F=\{J_1, J_2, \ldots, J_n\}$. Then, the tasks with small workloads will be processed first. More specifically, for $\forall j, j' \in J_i$ $(1 \le i \le n)$, if $\tau_j \le \tau_{j'}$, then the order of tasks j and j' in J_i satisfies $j \le j'$.
  - In the last example: $\tau_1 < \tau_2 < \tau_3 < \tau_4$

    $j_1 \le j_3 \quad j_2 \le j_4$

Solution: oFfline Task Assignment

- Basic Property of Optimal FTA Solution: Π^*_F
  - **Expected Processing Time (EPT_i):**

    the expected time for the user v_i to meet the requester, process the tasks in hand, and return the result.

    Example:

    user v_i has tasks $j_{i_1}, j_{i_2}, \cdots, j_{i_k}$

$$EPT_i = \frac{2}{\lambda_i} + \tau_{i_1} + \tau_{i_2} + \cdots + \tau_{i_k}$$

Solution: oFfline Task Assignment

- Basic Property of Optimal FTA Solution: Π^*_F

Theorem 3: suppose that the workloads of the j_1, j_2, …, j_m satisfy $\tau_1 \le \tau_2 \le \ldots \le \tau_m$, among which the tasks j_1, j_2, …, $j_{k\text{-}1}$ ($1 \le k \le m$) have been assigned. Assume that the current expected processing time of user v_i is EPT_i. Then, the optimal task assignment strategy Π^*_F satisfies:

the task j_k will be assigned to the user who currently has the minimum expected processing time, i.e.,

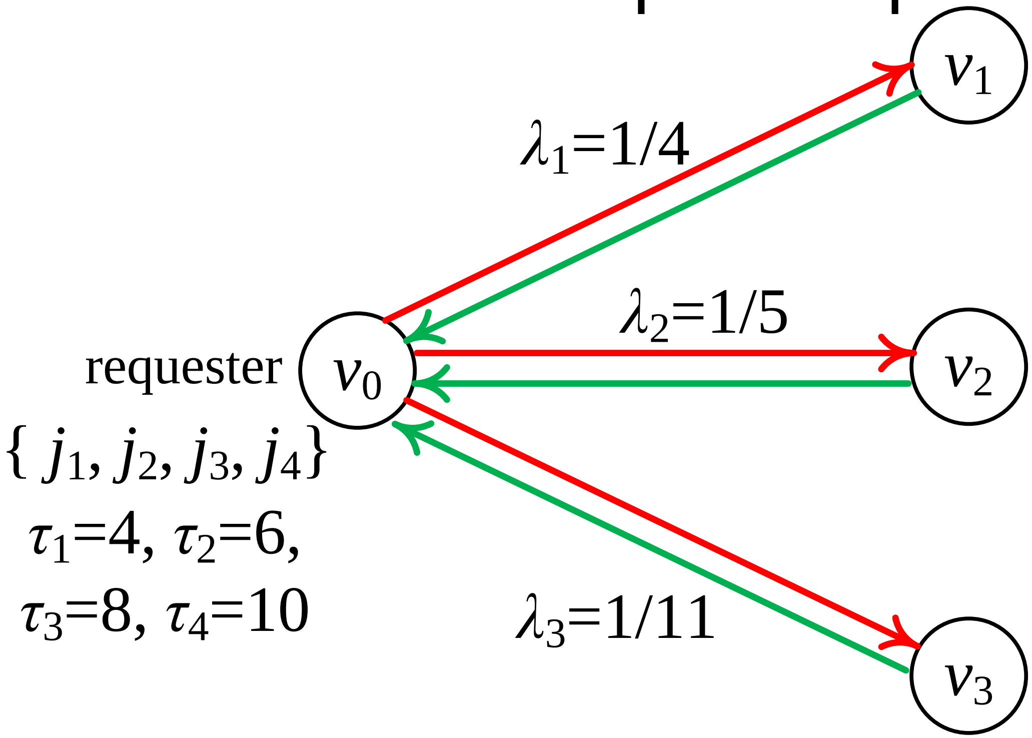
$$EPT_i = \mathrm{Min}\{EPT_1, \ldots, EPT_n\} \Rightarrow j_k \in J_i$$

Solution: oFfline Task Assignment

- Basic Idea:

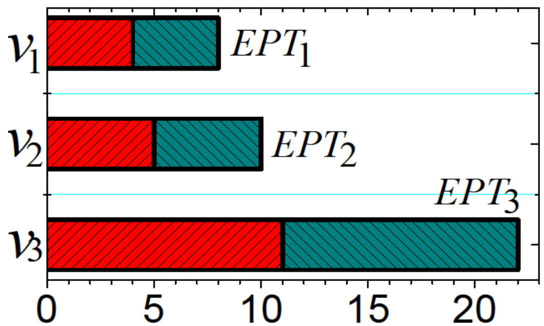
  we always assign the minimum workload task among the tasks that have not been assigned to the user with the smallest expected processing time

  - **Example:**

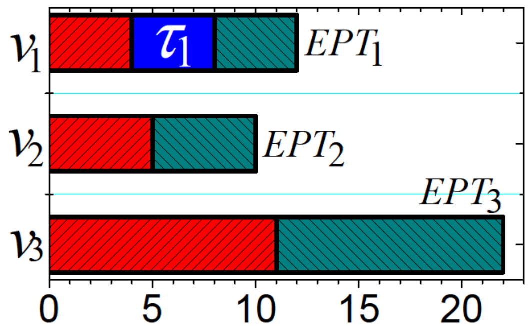


Solution: oFfline Task Assignment

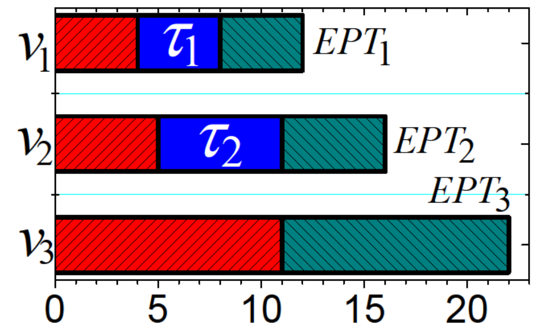
– **Example:**



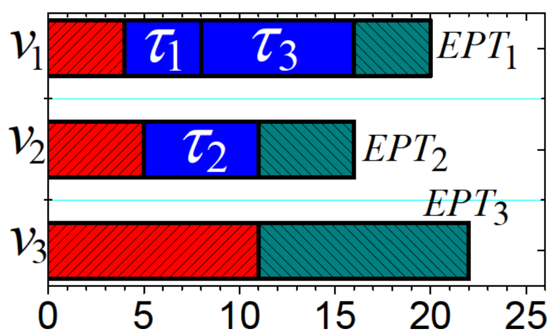
The initial state



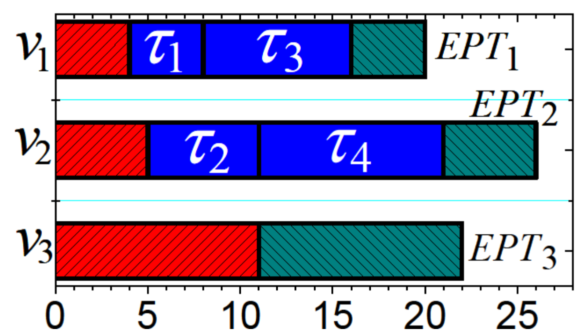
Assign task j_1



Assign task j_2



Assign task j_3



Assign task j_4

Solution: oFfline Task Assignment

- The Detailed FTA Algorithm:

Algorithm 1 The FTA Algorithm

Require: $J = \{j_1, j_2, \cdots, j_m : \tau_1 \leq \tau_2 \leq \cdots \leq \tau_m\}$,
$V = \{v_1, v_2, \cdots, v_n : \lambda_1, \lambda_2, \cdots, \lambda_n\}$.

Ensure: $\Pi_F = \{J_1, J_2, \cdots, J_n\}$

for each user v_i **do**
  $J_i = \emptyset$;
  $EPT_i = \frac{2}{\lambda_i}$;
for task j from j_1 to j_m **do**
  $i_{min} = argmin\{EPT_1, EPT_2, \cdots, EPT_n\}$;
  Assign task j to $v_{i_{min}}$: $J_{i_{min}} = J_{i_{min}} + \{j\}$;
  $EPT_{i_{min}} = EPT_{i_{min}} + \tau_{i_{min}}$;

Corollary 4: The FTA algorithm can achieve the optimal average makespan for the offline task assignment case.

Solution: oNline Task Assignment

- **NTA: oNline Task Assignment**

  the requester dynamically assigns tasks at each time when it encounters a mobile user

- Instant Processing Time (IPT)

  the time for a mobile user, who has just encountered and received some tasks from the requester, to process these tasks and return the results

  Example: user v_i has tasks $j_{i_1}, j_{i_2}, \cdots, j_{i_k}$

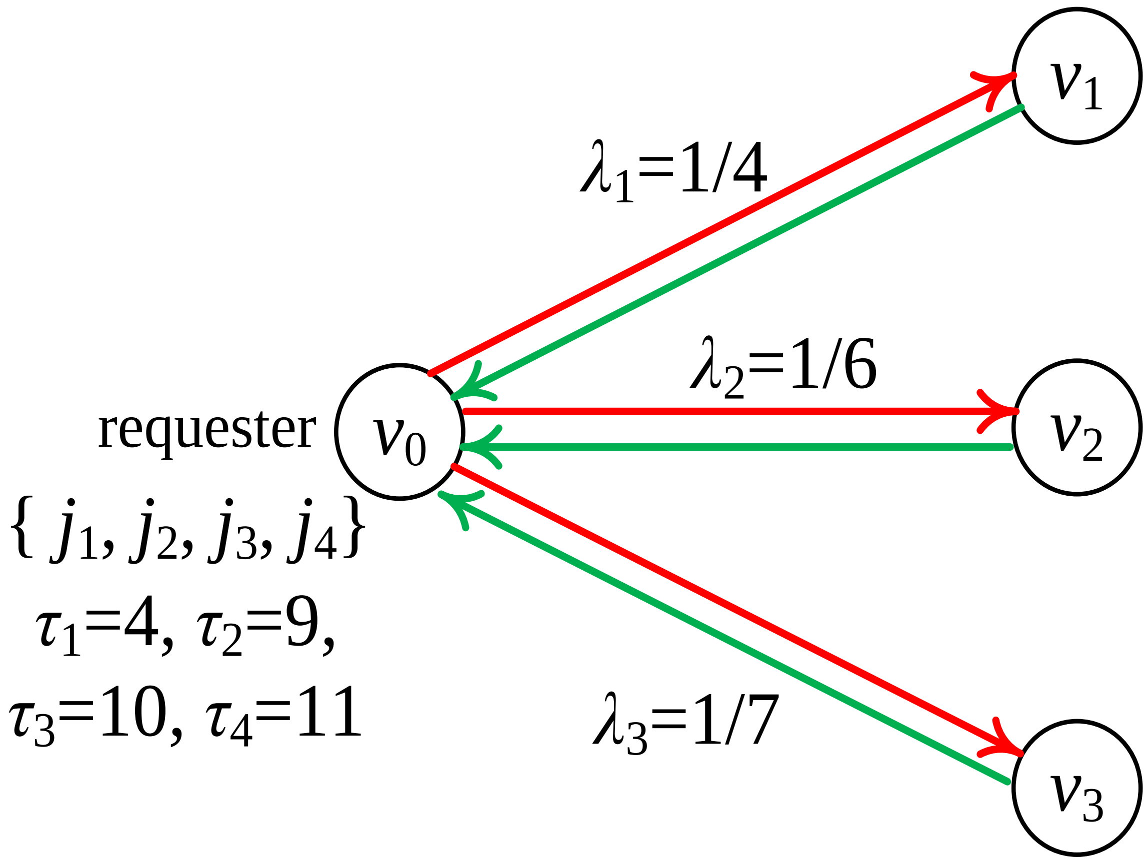
$$IPT_i = \frac{1}{\lambda_i} + \tau_{i_1} + \tau_{i_2} + \cdots + \tau_{i_k}$$

Solution: oNline Task Assignment

- **Basic idea**
  - When the requester encounters a mobile user v_i, it first computes IPT_i and EPT values of other users who have not been met by itself.
  - The requester adopts the similar greedy strategy in FTA to assign tasks, while using IPT_i to replace EPT_i in FTA.
  - The requester will get a result $\{J_1, \ldots, J_i, \ldots, J_n\}$. Then, the requester assigns the tasks in J_i to user v_i, while keeping the remaining tasks (i.e., the tasks in $J - J_i$) in hand.

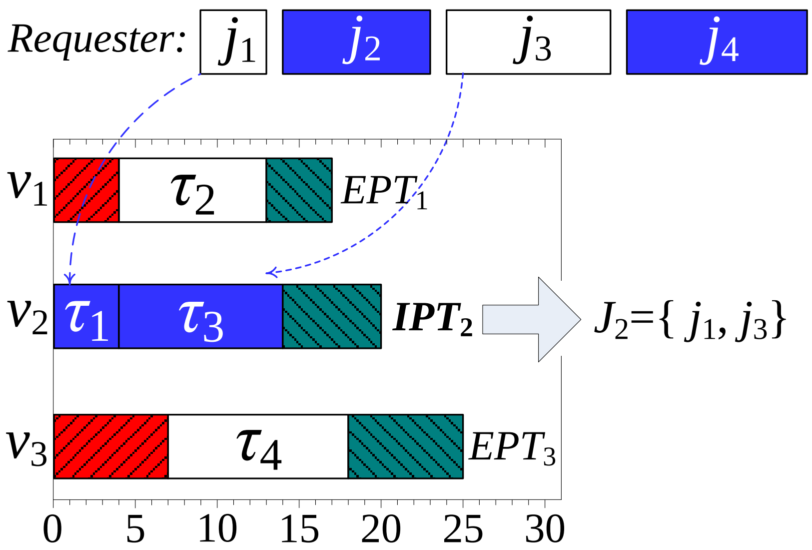
Solution: oNline Task Assignment

- **Example**

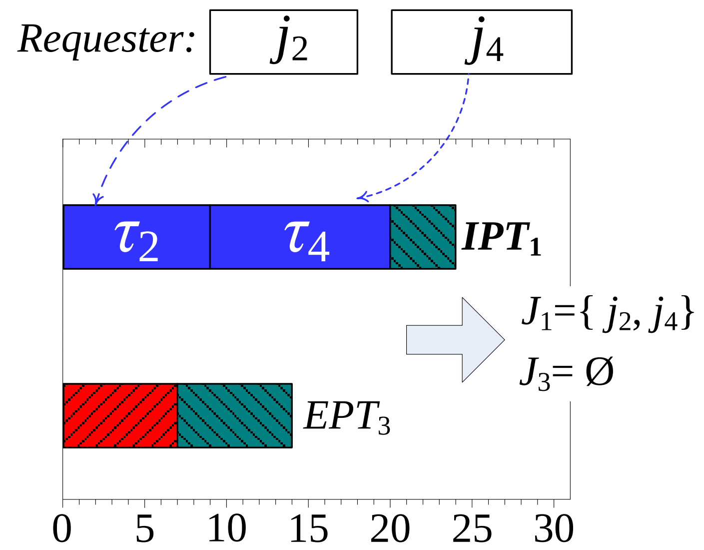


Solution: oNline Task Assignment

- **Example**



the requester determines J_2 when it meets v_2: $J_2 = \{j_1,\ j_3\}$



the requester determines J_1 when it meets v_1: $J_1 = \{j_2,\ j_4\}$

Solution: oNline Task Assignment

- **The Detailed NTA Algorithm**

Algorithm 2 The NTA Algorithm

Require: $J = \{j_1, j_2, \cdots, j_m : \tau_1 \leq \tau_2 \leq \cdots \leq \tau_m\}$,
$V = \{v_1, v_2, \cdots, v_n : \lambda_1, \lambda_2, \cdots, \lambda_n\}$.

Ensure: $\Pi_N = \{J_1, J_2, \cdots, J_n\}$

When the requester meets user v_i **do**

```
IPT_i = 1/λ_i;
J_i = ∅;
V = V − {v_i};
for each other user v_l ∈ V do
    J_l = ∅;
    EPT_l = 2/λ_l;
for task j from j_1 to j_m and j ∈ J do
    i_min = argmin({IPT_i} + {EPT_l | v_l ∈ V});
    if i_min = i then
        Assign task j to v_i: J_i = J_i + {j};
        IPT_i = IPT_i + τ_i;
        J = J − {j};
    else
        J_{i_min} = J_{i_min} + {j};
        EPT_i = EPT_i + τ_{i_min};
return J_i;
```

Where: $IPT_i = \frac{1}{\lambda_i}$; $EPT_l = \frac{2}{\lambda_l}$; $i_{min} = argmin(\{IPT_i\} + \{EPT_l \mid v_l \in V\})$; $J_{i_{min}} = J_{i_{min}} + \{j\}$; $EPT_i = EPT_i + \tau_{i_{min}}$.

The computation overhead is $O(mn^2)$

Solution: oNline Task Assignment

- **Performance analysis**
  - **Theorem 5:** The NTA algorithm can achieve a smaller average makespan than FTA, i.e.,

$$AM(\Pi_N) \leq AM(\Pi^*_F)$$

  - **Theorem 6:** Assume that there is a god, who can foresee at what time the requester will meet which user. Based on this, it can give an ideal optimal task assignment strategy Π_{OPT}. Then, we have:

$$AM(\Pi_N) - AM(\Pi_{OPT}) \leq \sum_{i=1}^{n} \frac{2}{\lambda_i} \qquad \frac{AM(\Pi_N)}{AM(\Pi_{OPT})} \leq 1 + \frac{m \sum_{i=1}^{n} \frac{2}{\lambda_i}}{\sum_{j=1}^{m} \tau_j}$$

Simulation

- **Algorithms in comparison**
  - ***FTA, NTA, OPT***
  - **Water Filling (*WF*) [Mobihoc'12]**
  - **Largest-First (*LF*)**

- **Metrics**
  - **Average makespan**

Simulation

- Real Traces
  - Statistics of traces

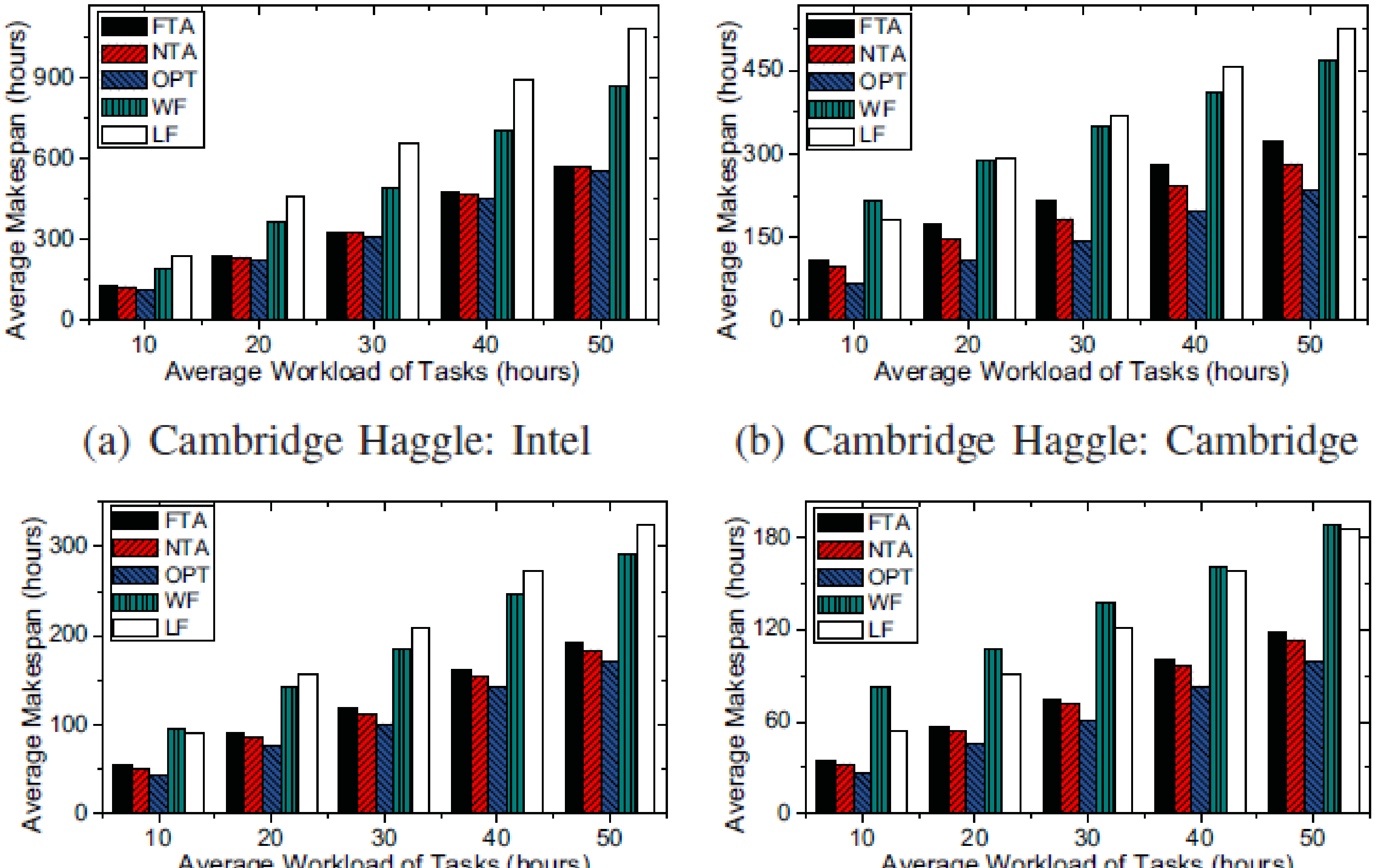
Trace	Contacts	Length (hours)	Requester	Other users
Intel	2,766	99.8	9	128
Cambridge	6,732	145.6	12	223
Infocom	28,216	76.6	41	264
UMassDieselNet	227,657	95.3	4	36

  - Other Settings

Parameter name	Default value	Range
Number of tasks m	300	200-1000
Average workload τ hours)	20	10-50

Simulation

Results: average makespan vs. average workload



(a) Cambridge Haggle: Intel

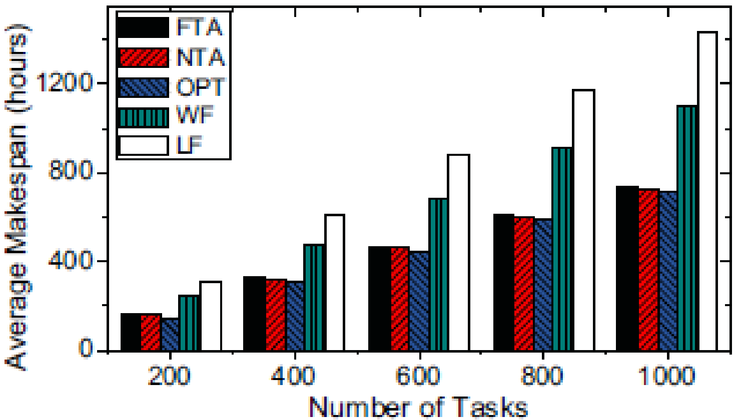
(b) Cambridge Haggle: Cambridge

(c) Cambridge Haggle: Infocom

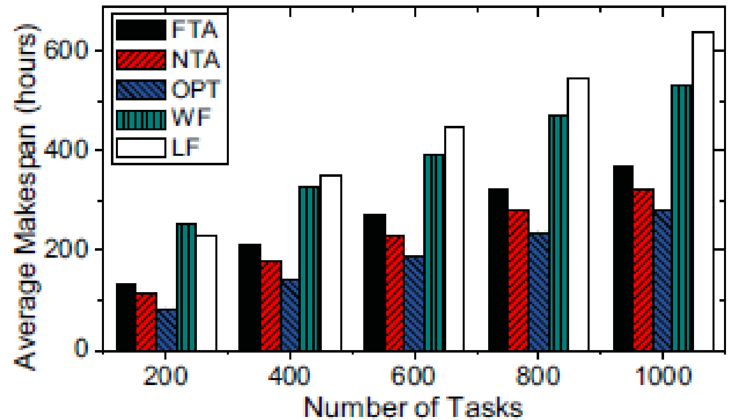
(d) UMassDieselNet

Simulation

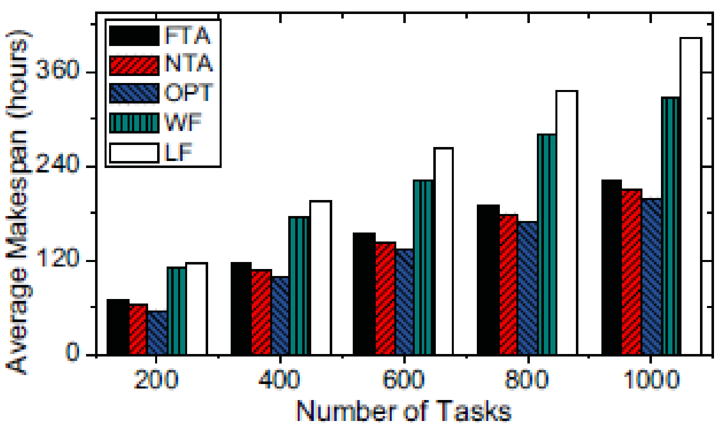
Results: average makespan vs. number of tasks



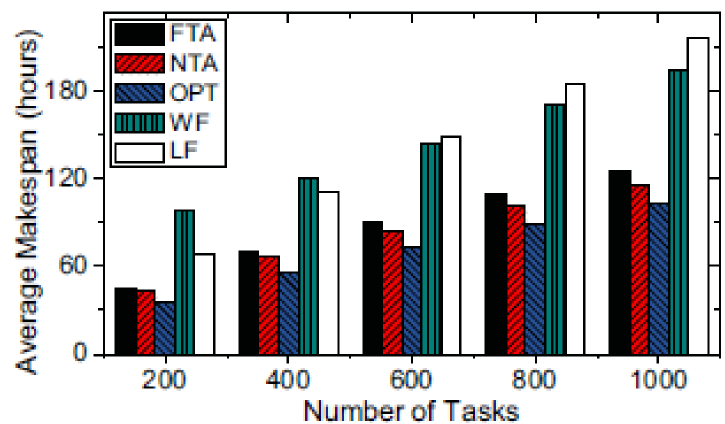
(a) Cambridge Haggle: Intel



(b) Cambridge Haggle: Cambridge



(c) Cambridge Haggle: Infocom



(d) UMassDieselNet

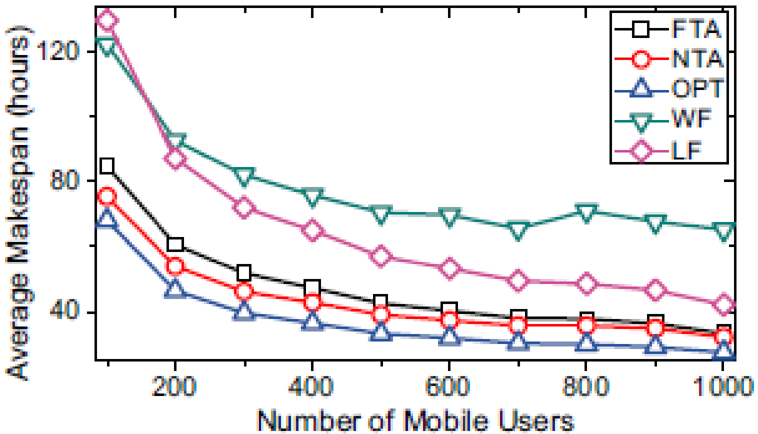
Simulation

- **Synthetic Traces**

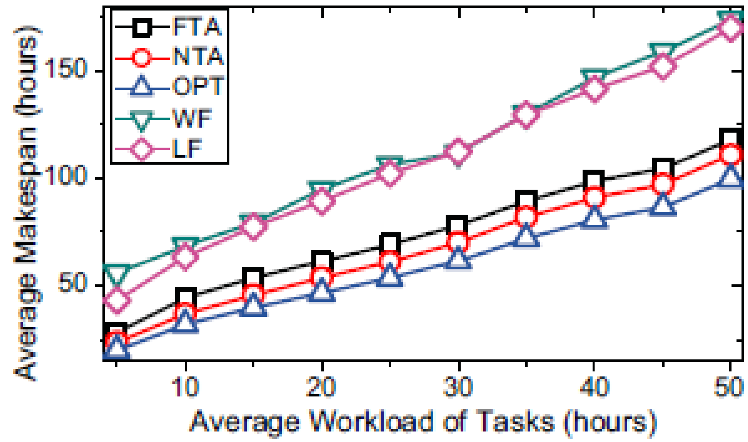
Parameter name	Default value	Range
Number of users n	400	100-1000
Number of requesters	5%	5%-10%
The average rate parameter λ	0.05	0.01-0.1
Number of tasks m	300	100-1000
The average workload	10	5 - 50

Simulation

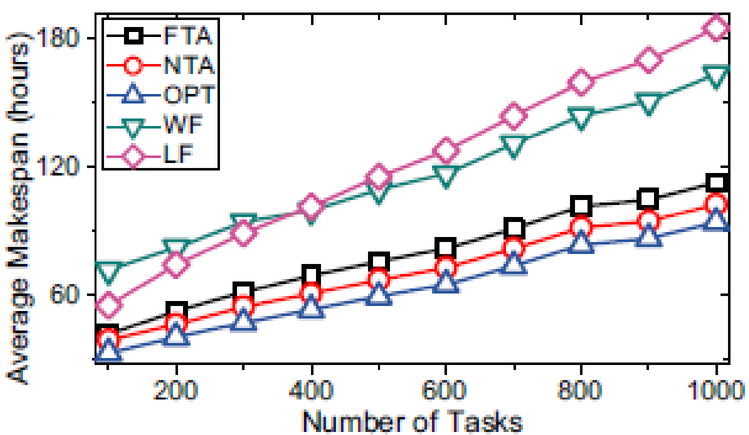
Results:



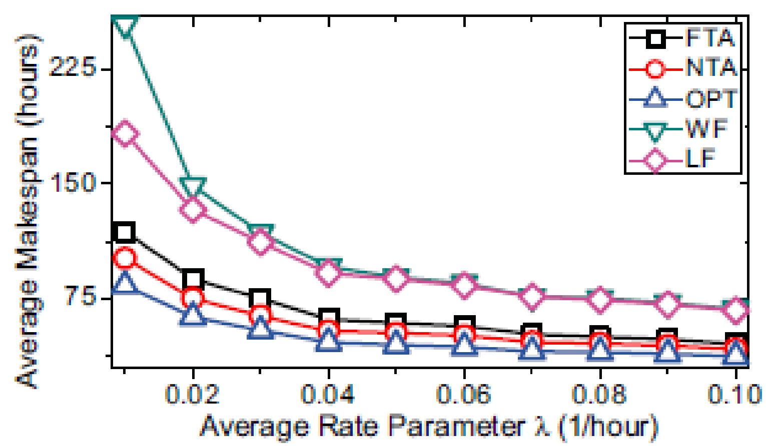
(a) Change the number of users



(b) Change the average workload



(c) Change the number of tasks



(d)Change the average rate parameter

Conclusion

- FTA is the optimal offline task assignment algorithm
- NTA can achieve a smaller average makespan in the online decision case
- The absolute error of NTA mainly depends on the expected meeting time between the requester and other users. When the average expected meeting time is very small, our algorithm can even achieve the nearly optimal result.

Thanks!

Q&A