Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing

Slađana Jošilo, György Dán

Department of Network and Systems Engineering
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

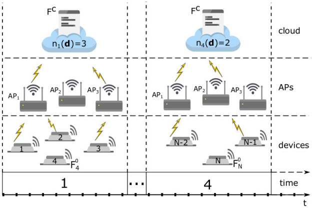
Zürich, May 16, 2018

Mobile Edge Computing (MEC)

Mobile Edge Computing (MEC)

Enabler for IoT Applications

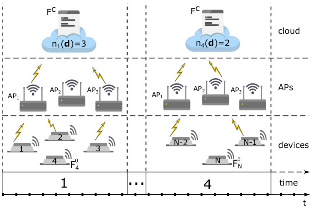
- Many IoT applications are based on deployment of wireless sensors that collect data periodically
  - computationally demanding processing
  - low latency response



Mobile Edge Computing (MEC)

Enabler for IoT Applications

- Many IoT applications are based on deployment of wireless sensors that collect data periodically
  - computationally demanding processing
  - low latency response



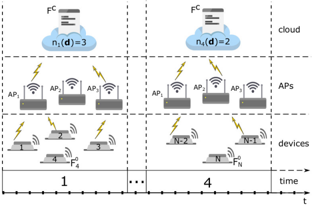
MEC can meet IoT application requirements by providing

- high bandwidth and computing resources close to the end users
- low latency
- real-time access to radio network information
- collaboration between mobile operators and application providers

Mobile Edge Computing (MEC)

Enabler for IoT Applications

- Many IoT applications are based on deployment of wireless sensors that collect data periodically
  - computationally demanding processing
  - low latency response



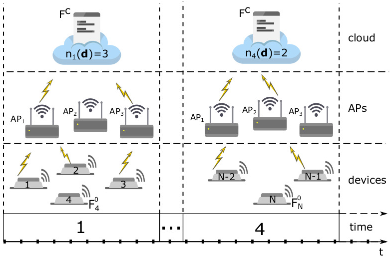
MEC can meet IoT application requirements by providing

- high bandwidth and computing resources close to the end users
- low latency
- real-time access to radio network information
- collaboration between mobile operators and application providers

Important question

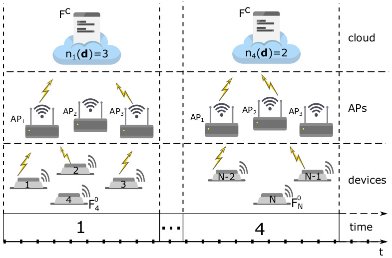
How to utilize mobile edge resources efficiently?

Mobile Cloud Computing System with Periodic Tasks



- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of WDs $\mathcal{N} = \{1, 2, ..., N\}$
- Set of time slots $\mathcal{T} = \{1, 2, \dots, T\}$

Mobile Cloud Computing System with Periodic Tasks

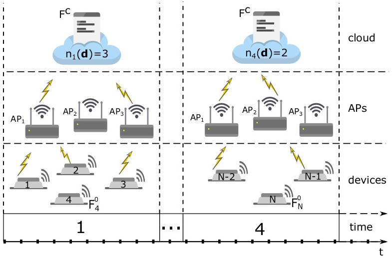


- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of WDs $\mathcal{N} = \{1, 2, ..., N\}$
- Set of time slots $\mathcal{T} = \{1, 2, \ldots, T\}$

Computation Offloading

- Decision of WD i $d_i = \begin{cases} (t, 0), & \text{if performing the task locally in time slot } t \\ (t, a), & \text{if offloading the task using AP } a \text{ in time slot } t \end{cases}$
- Set of decisions for all WDs is a *strategy profile* **d**

Mobile Cloud Computing System with Periodic Tasks



- Cloud server
- Set of APs $\mathcal{A} = \{1, 2, ..., A\}$
- Set of WDs $\mathcal{N} = \{1, 2, ..., N\}$
- Set of time slots $\mathcal{T} = \{1, 2, \ldots, T\}$

Computation Offloading

- Decision of WD i $d_i = \begin{cases} (t, 0), & \text{if performing the task locally in time slot } t \\ (t, a), & \text{if offloading the task using AP } a \text{ in time slot } t \end{cases}$
- Set of decisions for all WDs is a *strategy profile* $\mathbf{d}$
- Number of WDs that offload via AP a in time slot t in a strategy profile $\mathbf{d}$ is $n_{(t,a)}(\mathbf{d})$
- Total number of WDs that offload in time slot t in a strategy profile $\mathbf{d}$ is $n_t(\mathbf{d})$

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,(t,a)}^c(\mathbf{d}) = \gamma_i^T \underbrace{\left(\overbrace{T_{i,(t,a)}^{c,off}(n_{(t,a)}(\mathbf{d}))}^{\text{time to offload}} + \overbrace{T_{i,t}^{c,exe}(n_t(\mathbf{d}))}^{\text{time to execute}}\right)}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,(t,a)}^c(n_{(t,a)}(\mathbf{d}))}_{\text{energy to offload}}$$

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,(t,a)}^c(\mathbf{d}) = \gamma_i^T \underbrace{\Big(\overbrace{T_{i,(t,a)}^{c,off}(n_{(t,a)}(\mathbf{d}))}^{\text{time to offload}} + \overbrace{T_{i,t}^{c,exe}(n_t(\mathbf{d}))}^{\text{time to execute}}\Big)}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,(t,a)}^c(n_{(t,a)}(\mathbf{d}))}_{\text{energy to offload}}$$

Coordinating MEC resources for offloading of periodic tasks

- Interactions between WDs modeled as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,(t,a)}^c(\mathbf{d}) = \gamma_i^T \underbrace{\Big(\overbrace{T_{i,(t,a)}^{c,off}(n_{(t,a)}(\mathbf{d}))}^{\text{time to offload}} + \overbrace{T_{i,t}^{c,exe}(n_t(\mathbf{d}))}^{\text{time to execute}}\Big)}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,(t,a)}^c(n_{(t,a)}(\mathbf{d}))}_{\text{energy to offload}}$$

Coordinating MEC resources for offloading of periodic tasks

- Interactions between WDs modeled as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$
- Is there a task allocation in which all selfish WDs are satisfied?

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C_{i,(t,a)}^c(\mathbf{d}) = \gamma_i^T \underbrace{(\overbrace{T_{i,(t,a)}^{c,off}(n_{(t,a)}(\mathbf{d}))}^{\text{time to offload}} + \overbrace{T_{i,t}^{c,exe}(n_t(\mathbf{d}))}^{\text{time to execute}})}_{\text{delay}} + \gamma_i^E \underbrace{E_{i,(t,a)}^c(n_{(t,a)}(\mathbf{d}))}_{\text{energy to offload}}$$

Coordinating MEC resources for offloading of periodic tasks

- Interactions between WDs modeled as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$
- Is there a task allocation in which all selfish WDs are satisfied?
- Can it be computed using a decentralized algorithm?

Local Computing Cost

$$C_i^0 = \gamma_i^T \underbrace{T_i^0}_{\text{delay}} + \gamma_i^E \underbrace{E_i^0}_{\text{energy}}$$

Cloud Offloading Cost

$$C^c_{i,(t,a)}(\mathbf{d}) = \gamma_i^T \underbrace{\Big(\overbrace{T^{c,off}_{i,(t,a)}(n_{(t,a)}(\mathbf{d}))}^{\text{time to offload}} + \overbrace{T^{c,exe}_{i,t}(n_t(\mathbf{d}))}^{\text{time to execute}}\Big)}_{\text{delay}} + \gamma_i^E \underbrace{E^c_{i,(t,a)}(n_{(t,a)}(\mathbf{d}))}_{\text{energy to offload}}$$

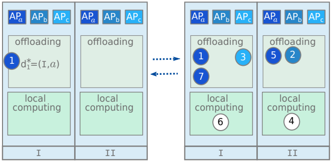
Coordinating MEC resources for offloading of periodic tasks

- Interactions between WDs modeled as a strategic game $\Gamma = <\mathcal{N}, (\mathfrak{D}_i)_i, (C_i)_i>$
- Is there a task allocation in which all selfish WDs are satisfied?
- Can it be computed using a decentralized algorithm?
- What is the complexity of the algorithm?
- How good are the system performance?

Coordinated Myopic Alternating Best (MB)

- WDs enter the game one at a time and implement BR over all time slots

- Induction phase - starting from an empty system, WDs enter the game one at a time and play BR

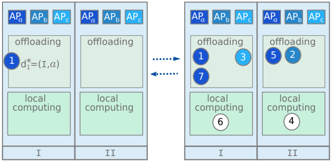


- Update phase - consists of two types of BR sequences that are played alternatingly
  (1) WDs are not allowed to replace the previous deviators
  (2) WDs are only allowed to replace the previous deviators

Coordinated Myopic Alternating Best (MB)

- WDs enter the game one at a time and implement BR over all time slots

- Induction phase - starting from an empty system, WDs enter the game one at a time and play BR



- Update phase - consists of two types of BR sequences that are played alternatingly
  (1) WDs are not allowed to replace the previous deviators
  (2) WDs are only allowed to replace the previous deviators

Random Slot (RS) Equilibrium

- WDs enter the game one at a time and choose a time slot at random
- WDs implement BR within the chosen time slot

Main Results

Computability

- MB algorithm computes a NE of game Γ in $O(N^2 \times T \times A)$ steps

Main Results

Computability

- MB algorithm computes a NE of game Γ in $O(N^2 \times T \times A)$ steps
- RS algorithm computes a strategy profile in which WDs implement a NE in time slots
    - Strategy profile computed by the RS algorithm is not necessarily a NE of game Γ
    - Computational complexity of the RS algorithm is $O(N^2 \times A)$

Main Results

Computability

- MB algorithm computes a NE of game Γ in $O(N^2 \times T \times A)$ steps
- RS algorithm computes a strategy profile in which WDs implement a NE in time slots
  - Strategy profile computed by the RS algorithm is not necessarily a NE of game Γ
  - Computational complexity of the RS algorithm is $O(N^2 \times A)$

Price of Anarchy (PoA) Bounds

- Upper bound on the PoA for the computation offloading game:
  - $N \leq T$: $PoA = 1$
  - $N > T$: $PoA \leq \dfrac{\sum_{i \in \mathcal{N}} C_i^0}{\sum_{i \in \mathcal{N}} \min\{C_i^0, C_{i,1}^{\bar{c}}, ..., C_{i,A}^{\bar{c}}\}}$
  - $C_{i,a}^{\bar{c}}$ is the offloading cost of WD i via AP a when she offloads alone

Performance Evaluation - Simulations

Evaluation scenario

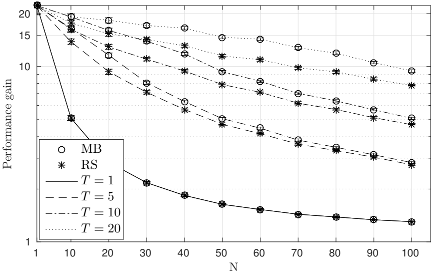
- $A = 4$ APs, $F^c = 100$ *Gcycles* and $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Tasks: data size $\sim \mathcal{U}(0.42, 2)$ Mb , complexity $\sim \mathcal{U}(0.1, 0.8)$ *Gcycles*

Performance Evaluation - Simulations

Evaluation scenario

- $A = 4$ APs, $F^c = 100$ *Gcycles* and $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Tasks: data size $\sim \mathcal{U}(0.42, 2)$ Mb , complexity $\sim \mathcal{U}(0.1, 0.8)$ *Gcycles*

System Cost Performance



- *Performance gain* defined w.r.t. a strategy profile in which all WDs perform computation locally
- *Performance gain* of the MB algorithm is higher than that of the RS algorithm for $T > 1 \implies$ coordination is important

Performance Evaluation - Simulations

Evaluation scenario

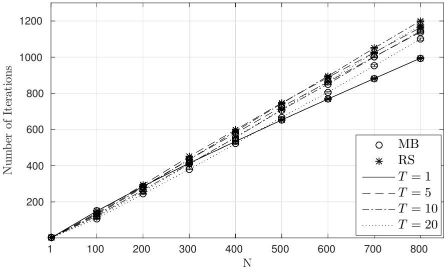
- $A = 4$ APs, $F^c = 100$ *Gcycles* and $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Tasks: data size $\sim \mathcal{U}(0.42, 2)$ Mb , complexity $\sim \mathcal{U}(0.1, 0.8)$ *Gcycles*

Performance Evaluation - Simulations

Evaluation scenario

- $A = 4$ APs, $F^c = 100$ *Gcycles* and $F_i^0 \sim \mathcal{U}(0.5, 1)$ *Gcycles*
- Tasks: data size $\sim \mathcal{U}(0.42, 2)$ Mb , complexity $\sim \mathcal{U}(0.1, 0.8)$ *Gcycles*

Computational complexity



- Number of iterations scales approximately linearly with the number of WDs for both algorithms

Summary and Future Work

- Provided a game theoretical treatment of computation offloading for periodic tasks
- Proved the existence of equilibrium allocations
- Provided a polynomial time decentralized algorithm for computing an equilibrium

Summary and Future Work

- Provided a game theoretical treatment of computation offloading for periodic tasks
- Proved the existence of equilibrium allocations
- Provided a polynomial time decentralized algorithm for computing an equilibrium

- Interesting extensions
  - WDs with heterogeneous periodicities
  - resource allocation from the perspective of mobile cloud providers
  - D2D collaborative computation offloading

Decentralized Scheduling for Offloading of Periodic Tasks in Mobile Edge Computing

Slađana Jošilo, György Dán

Department of Network and Systems Engineering
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Zürich, May 16, 2018