CrowdCache: A Decentralized Game-Theoretic Framework for Mobile Edge Content Sharing

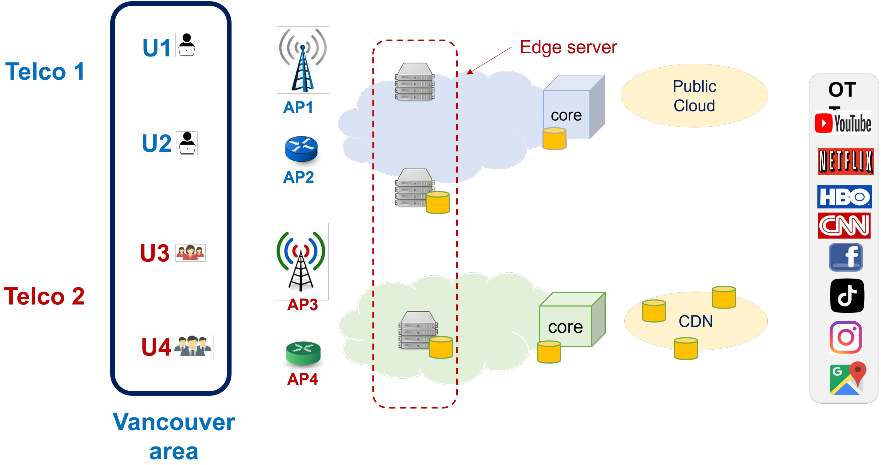
Presenter: Jiaming Cheng
Duong T.A. Nguyen, Duong Tung Nguyen and Angelia Nedić

Electrical and Computer Engineering, University of British Columbia
School of Electrical, Computer and Energy Engineering, Arizona State University

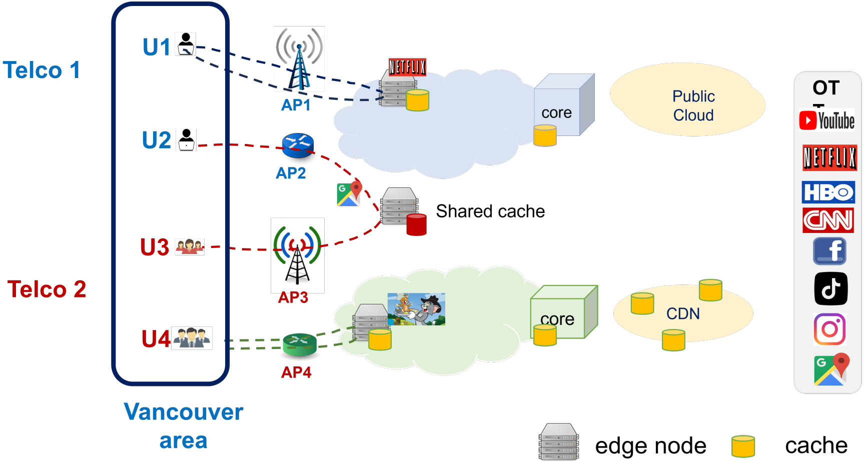
January 11, 2024

Outline

Edge cache


Telco 1
Telco 2
U1
U2
U3
U4
Vancouver area
AP1
AP2
AP3
AP4
Edge server
core
core
Public Cloud
CDN
OT
T

Edge cache

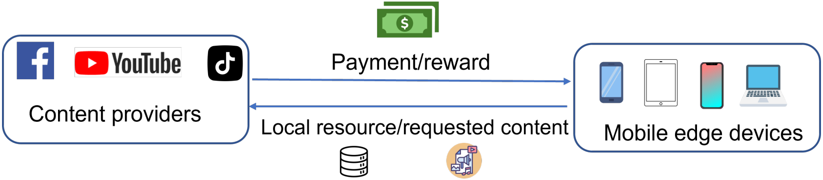

Telco 1
Telco 2
U1
U2
U3
U4
AP1
AP2
AP3
AP4
core
core
Public Cloud
Shared cache
CDN
OT
Vancouver area
edge node
cache

Crowdsourcing content sharing and caching

We consider a framework for mobile edge content sharing and caching via crowdsourcing, known as **CrowdCache**:

with crowdsourcing system:

- CP: recruit mobile edge devices to provide local storage for content cache.
- MEDs: request content services from the CP; earn incentives by providing local storage.



Crowdsourcing content sharing and caching

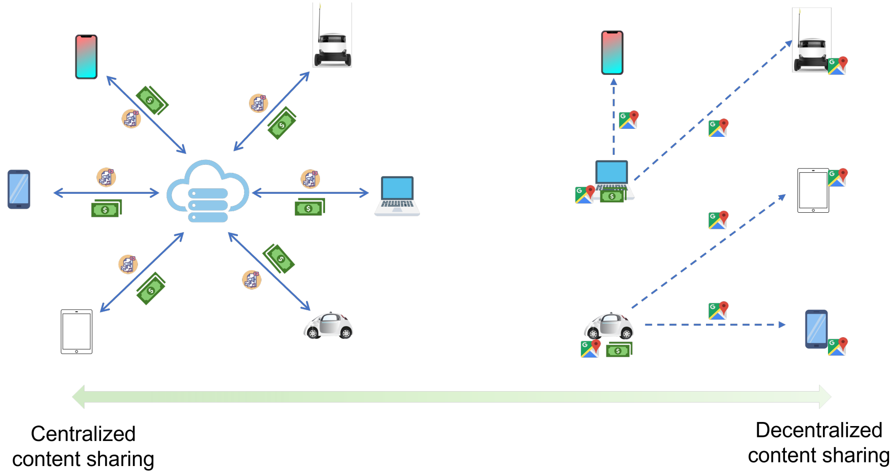
We consider a framework for mobile edge content sharing and caching via crowdsourcing, known as **CrowdCache**:
with crowdsourcing system:

- CP: recruit mobile edge devices to provide local storage for content cache.
- MEDs: request content services from the CP; earn incentives by providing local storage.

Business goal

The CP may incentivize MEDs to share content with each other in a decentralized manner to enhance system efficiency

Decentralized content sharing



Decentralized content sharing

Benefits

(a) Reduce duplicate data transmission;
(b) Decrease backhaul capacity requirement;
(c) Alleviate congestion within backbone network;
(d) Due to the idle resources of numerous MEDs, reduce upfront investment costs for CP and network operators

System model

For MEDs,

- N MEDs indexed by $i = 1, \ldots, N$ (set $\mathcal{I}$)
- x_i denote the storage a MED is willing to provide
- $c_i(x_i)$ cost function of MED i:

$$c_i(x_i) = Q_i x_i^2 + h_i x_i, \quad \forall i$$

  where Q_i is the quadratic cost, and h_i is the linear cost of MED i.
- This quadratic form discourages excessive storage allocation.
- C_i: maximum storage capacity available for cached content on MED i.
- Capacity constraint: $0 \le x_i \le C_i, \ \forall i$.

System model

For MEDs,

- N MEDs indexed by $i = 1, \ldots, N$ (set $\mathcal{I}$)
- x_i denote the storage a MED is willing to provide
- $c_i(x_i)$ cost function of MED i:

$$c_i(x_i) = Q_i x_i^2 + h_i x_i, \quad \forall i$$

  where Q_i is the quadratic cost, and h_i is the linear cost of MED i.
- This quadratic form discourages excessive storage allocation.
- C_i: maximum storage capacity available for cached content on MED i.
- Capacity constraint: $0 \leq x_i \leq C_i, \ \forall i$.

Objective

Each MED aims to maximize it own utility by determining its resource allocation decisions, taking into account costs and rewards

System model

For CP,

- The price for each unit of resource:

$$p(x) = \bar{P} - \gamma \sum_{j \in \mathcal{I}} x_j$$

  where $\bar{P}$ denotes the maximum unit reward offered by the CP
- γ: negative effect of other MEDs' benefits by sharing content, reflecting attitude towards the market demand for shared content:

System model

For CP,

- The price for each unit of resource:

$$p(x) = \bar{P} - \gamma \sum_{j \in \mathcal{I}} x_j$$

  where $\bar{P}$ denotes the maximum unit reward offered by the CP
- γ: negative effect of other MEDs' benefits by sharing content, reflecting attitude towards the market demand for shared content:
- Lower γ: encourage more MEDs to outsource their idle resources
- Higher γ: demand can be sufficiently fulfilled by existing participants

System model

- The price for each unit of resource:

$$p(x) = \bar{P} - \gamma \sum_{j \in \mathcal{I}} x_j$$

where $\bar{P}$ denotes the maximum unit reward offered by the CP

- The reward for MED i provided by the CP:

$$R_i(x_i, x_{-i}) = p(x)x_i, \ \forall i$$

System model

- The price for each unit of resource:

$$p(x) = \bar{P} - \gamma \sum_{j \in \mathcal{I}} x_j$$

  where $\bar{P}$ denotes the maximum unit reward offered by the CP
- The reward for MED i provided by the CP:

$$R_i(x_i, x_{-i}) = p(x)x_i, \ \forall i$$

- function of x_i and other MEDs x_{-i} in the set $\mathcal{I}$.
- Marginal benefit: reward of MED i is diminishing as other intensify the engagement on content sharing.

System model

- The price for each unit of resource:

$$p(x) = \bar{P} - \gamma \sum_{j \in \mathcal{I}} x_j$$

  where $\bar{P}$ denotes the maximum unit reward offered by the CP
- The reward for MED i provided by the CP:

$$R_i(x_i, x_{-i}) = p(x)x_i, \ \forall i$$

Thus, the utility function of each MED is defined as:

$$U_i(x_i, x_{-i}) = R_i(x_i, x_{-i}) - c_i(x_i), \ \forall i$$

Problem Formulation

We model this crowdsourcing mobile edge content caching and sharing problem as a non-cooperative game $(\Gamma = (\mathcal{I}, \{J_i\}, \{X_i\}))$.

- Local cost function for each MED:

$$J(x_i, x_{-i}) = -U_i(x_i, x_{-i}),\ \forall i = 1, \dots N$$

- Action set $x_i \in X_i = [0, C_i] \in \mathbb{R}$
- $x_{-i} \in X_{-i}$: joint action of all MEDs except MED i:
  $X_{-i} = X_1 \times \cdots \times X_{i-1} \times X_{i+1} \times X_N$
- Each MED seeks to minimize their local objective function without coordinating with other MEDs

Problem Formulation

This non-cooperative game $\Gamma = (\mathcal{I}, \{J_i\}, \{X_i\})$:

- Local cost function for each MED:

$$J(x_i, x_{-i}) = -U_i(x_i, x_{-i}),\ \forall i = 1, \dots N$$

- Action set $x_i \in X_i = [0, C_i] \in \mathbb{R}$
- $x_{-i} \in X_{-i}$: joint action of all MEDs except MED i:
  $X_{-i} = X_1 \times \cdots \times X_{i-1} \times X_{i+1} \times X_N$

Nash Equilibrium

A solution to the game Γ is a Nash equilibrium (NE) $x^* \in X_1 \times \cdots \times X_N$ such that for every agent $i \in [N]$, we have:

$$J_i(x_i^*, x_{-i}^*) \leq J_i(x_i, x_{-i}^*), \quad \forall x_i \in X_i. \tag{1}$$

Existence and uniqueness of the NE

Recall the results connecting NE and solutions of variation inequality:

Definition 1: Variation Inequality [1]

For a set $K \subseteq \mathbb{R}^d$ and a mapping $g : K \longrightarrow \mathbb{R}^d$, the variational inequality (VI) problem VI(K,g) is to determine a vector $q^* \in K$ such that

$$\langle g(q^*), q - q^* \rangle \geq 0, \forall q \in K \quad (2)$$

The set of solutions to VI(K,g) is denoted by SOL(K,g)

Existence and uniqueness of the NE

Recall the results connecting NE and solutions of variation inequality:

Proposition 1.4.2 of [1]

Suppose a networked Nash Game Γ, where action sets of the MEDs $\{X_i\}$ are closed and convex, and the cost functions J_i are continuously differentiable and convex in x_i for every fixed x_{-i} from X_i. A vector $x^* \in X$ is a NE for the game Γ if and only if $x^* \in SOL(X, F)$ where F is the game mapping.

Definition 2: Game mapping

The game mapping $F(x) : X \longrightarrow \mathbb{R}^N$ is defined as

$$F(x) \triangleq \left[\nabla_i J_1(x_1, x_{-1}), \ldots, \nabla_N J_N(x_N, x_{-N})\right], \quad \forall x_i \in X_i. \tag{2}$$

Existence and uniqueness of the NE

Recall the results connecting NE and solutions of variation inequality:

Proposition 1.4.2 of [1]

Suppose a networked Nash Game Γ, where action sets of the MEDs $\{X_i\}$ are closed and convex, and the cost functions J_i are continuously differentiable and convex in x_i for every fixed x_{-i} from X_i. A vector $x^* \in X$ is a NE for the game Γ if and only if $x^* \in SOL(X, F)$ where F is the game mapping.

Lemma 2

This game mapping is strongly monotone in X with constant $\mu = 2\min_{i \in \mathcal{I}} Q_i + 2\gamma > 0$. $J_i(x_i, x_{-i}), \forall i$ is strongly convex on $\mathbb{R}$ for every $x_i \in \mathbb{R}^{N-1}$ with constant μ

Existence and uniqueness of the NE

Recall the results connecting NE and solutions of variation inequality:

Proposition 1.4.2 of [1]

Suppose a networked Nash Game Γ, where action sets of the MEDs $\{X_i\}$ are closed and convex, and the cost functions J_i are continuously differentiable and convex in x_i for every fixed x_{-i} from X_i. A vector $x^* \in X$ is a NE for the game Γ if and only if $x^* \in SOL(X, F)$ where F is the game mapping.

Proposition 2.3.3 of [1]

Given VI(K,g), suppose K is closed and convex, and the mapping g is continuous and strongly monotone. Then, the solution set $SOL(X, F)$ is nonempty and is a singleton.

[1] F. Facchinei and J. Pang. Finite-Dimensional Variational Inequalities and Complementarity Problems.

Property

Theorem 3

Consider the game $\Gamma = (\mathcal{I}, \{J_i\}, \{X_i\})$. There exists a unique NE in Γ. Moreover, the NE is the solution of VI(X,F), where F is the game mapping.

Remark: Theorem 3 guarantees the existence and uniqueness of NE in this game Γ. However, the formulation of the NE based on VI(X,F) **does not consider the distributed nature of the proposed problem.**

Property

Next, we present the Lipschitz continuity of the gradient of local objective function J_i for all i. The following Lemma is crucial to ensure the convergence of the distributed algorithm to the NE [2].

Theorem 4

Consider the game $\Gamma = (\mathcal{I}, \{J_i\}, \{X_i\})$.

(a) The mapping $\nabla_i J_i(\cdot, x_{-i})$ is Lipschitz continuous on $\mathbb{R}^{n-N}$ for every $x_{-i} \in \mathbb{R}^{N-1}$ with a uniform constant $L_1 = 2(\max_{i\in\mathcal{I}} Q_I + \gamma) \geq 0$, *for all* $i \in \mathcal{I}$.

(b) The mapping $\nabla_i J_i(x_i, \cdot)$ is Lipschitz continuous on $\mathbb{R}^{N-1}$ for every $x_{-i} \in \mathbb{R}^{N-1}$ with a uniform constant $L_2 = \gamma\sqrt{N-1}$,*for all* $i \in \mathcal{I}$.

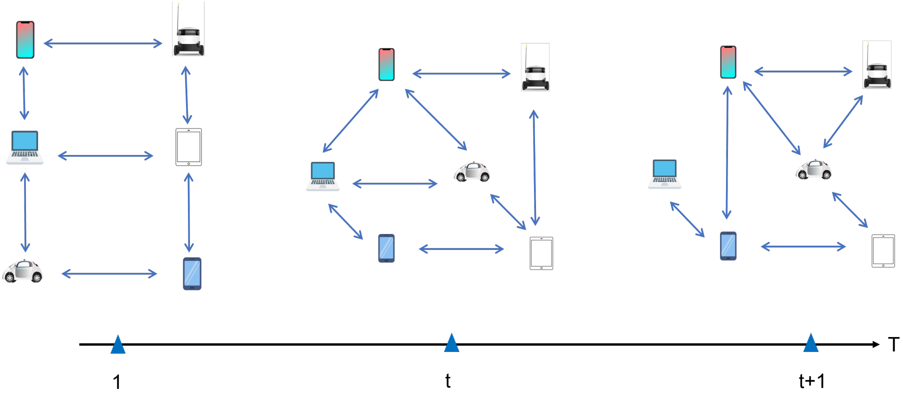
[2] D.T.A. Nguyen, D.T. Nguyen, and A. Nedić. "Distributed Nash equilibrium seeking over time-varying directed communication networks".

Contribution

Based on this crowdsourcing mobile edge content caching and sharing problem, we propose privacy-preserving algorithms to compute the NE:

- **Time-varying communication**: each MED may communicate with different neighbors at each round k.
- **Partial information exchange**: each MED only possesses partial information about their competitors actions through local communications (e.g., WiFi-Direct, LTE-Direct).
- **Decentralized local communication**: information exchange occurs solely between neighboring devices.

Time varying communication graphs



Time varying communication graphs

We define this time-varying communication graph as follows:

- **Time varying undirected graph**: $\mathbb{G}_k = (\mathcal{I}, E_k)$ where $\mathcal{I}$ is the set of MEDs, E_k is undirected link;
- **Unordered link** (i, j): MED i can receive information from MED j, and vice versa.
- **Neighbor set**: neighbour set for every MED i:

$$\mathcal{N}_{i,k} = \{j \in \mathcal{I} \mid (i, j) \in E_k\} \bigcup \{i\}$$

Remark: The sets of neighbors $\mathcal{N}_{i,k}$ always include MED i. Every MED (node) i has a self loop in each graph $\mathbb{G}_k, \forall k$

Time varying communication graphs

Assumption 1

The time-varying undirected graph sequence $\{\mathbb{G}_k\}$ is B-connected. There existis an integer $B \geq 1$ such that the graph with edge set $E_k^B = \bigcup_{i=kB}^{(k+1)B-1} E_i$ is connected for every $k \geq 0$

Remark: This assumption ensures:

(a) After B-rounds of communication, there exits a path between any pair of MEDs in the system.

(b) No MED is isolated from the rest of the system.

Time varying communication graphs

Assumption 1

The time-varying undirected graph sequence $\{\mathbb{G}_k\}$ is B-connected. There existis an integer $B \geq 1$ such that the graph with edge set $E_k^B = \bigcup_{i=kB}^{(k+1)B-1} E_i$ is connected for every $k \geq 0$

Intuition:

(a) This assumption is easily satisfied as long as MEDs are within the coverage area of cellular or Wi-Fi networks.

(b) MEDs must be connected to the network for communication with CP and other MEDs if deciding to share content and local storage.

Nash Equilibrium

Lemma 1

For all MED $i \in [m]$, action set X_i is closed and convex and cost function $J_i(x_i, x_{-i})$ is convex and continuously differentiable in x_i for each $x_{-i} \in X_{-i}$

Definition 1: First-order Optimality Condition

$x^* \in X$ is a NE of the game if and only if for all $i \in [N]$, we have:

$$\langle \nabla_i J_i(x_i^*, x_{-i}^*), x_i - x_i^* \rangle \geq 0, \quad \forall x_i \in X_i. \tag{2}$$

or equivalently,

$$x_i^* = \Pi_{X_i}[x_i^* - \alpha_i \nabla_i J_i(x_i^*, x_{-i}^*)], \quad \forall i \in [N], \tag{3}$$

where $\alpha_i > 0$ is an arbitrary scalar.

Traditional algorithm

Basic algorithm

- Initial point $x_i^0 \in X_i$
- Each agent i updates its decision at time k as follows:

$$x_i^{k+1} = \Pi_{X_i}[x_i^k - \alpha_i \nabla_i J_i(x_i^k, x_{-i}^k)] \quad (4)$$

where $\alpha_i > 0$ is the step-size.

- x_k^i: MED i's actual decision at time k.
- x_k^{-i}: actual decisions of MEDs other than i at time k.

Traditional algorithm

Basic algorithm

- Initial point $x_i^0 \in X_i$
- Each agent i updates its decision at time k as follows:

$$x_i^{k+1} = \Pi_{X_i}[x_i^k - \alpha_i \nabla_i J_i(x_i^k, x_{-i}^k)] \quad (4)$$

where $\alpha_i > 0$ is the step-size.

- x_k^i: MED i's actual decision at time k.
- x_k^{-i}: actual decisions of MEDs other than i at time k.

Remark:

(a) Require that every agent i has access to all other agents' decisions x_{-i}^k at every time k

(b) Require full information about competitors' cost functions (privacy concerns)

Algorithm

Each agent i updates its decision at time k as follows:

Basic algorithm

$$x_{k+1}^{i} = \Pi_{X_i}[x_k^i - \alpha_i \nabla_i J_i(x_k^i, x_k^{-i})].$$

Distributed Algorithm

$$x_{k+1}^{i} = \Pi_{X_i}[z_{k+1}^{ii} - \alpha_i \nabla_i J_i(z_{k+1}^{ii}, z_{k+1}^{i,-i})]. \quad (5)$$

where $\alpha_i > 0$ is the step-size.

- z_k^{ij}: MED i's estimate of the decision x_k^j for MED $j \neq i$.
- $z_k^i = (z_k^{i1}, \dots, z_k^{ii}, z_k^{iN})^\intercal \in \mathbb{R}^N \quad (\mathbf{z_k^{ii}} = \mathbf{x_k^i} \in \mathbb{R})$
- $z_k^{i,-i} \in \mathbb{R}^{N-1}$: the estimate of MED i for the decisions of the other MEDs

Algorithm

Algorithm 1: DCrowdCache

Initialization: arbitrary initial vectors $z_0^{i,-i} \in \mathbb{R}^{N-1}$ and $x_0^i \in \mathbb{R}$
For each time $k = 0, 1, \dots,$ every MED does the following:

(1) Receives z_k^j and $|\mathcal{N}_{jk}|$ from neighbors $j \in \mathcal{N}_{ik}$;

(2) Sends z_k^i and $|\mathcal{N}_{ik}|$ to neighbors $j \in \mathcal{N}_{ik}$;

Algorithm

Algorithm 1: DCrowdCache

Initialization: arbitrary initial vectors $z_0^{i,-i} \in \mathbb{R}^{N-1}$ and $x_0^i \in \mathbb{R}$

For each time $k = 0, 1, \ldots,$ every MED does the following:

(1) Receives z_k^j and $|\mathcal{N}_{jk}|$ from neighbors $j \in \mathcal{N}_{ik}$;

(2) Sends z_k^i and $|\mathcal{N}_{ik}|$ to neighbors $j \in \mathcal{N}_{ik}$;

(3) Calculates the weights using Metropolis weights:

$$[W_k]_{ij} = \begin{cases} 1/(1+\max\{|\mathcal{N}_{ik}|, |\mathcal{N}_{jk}|\}, & \text{if} \quad j \in \mathcal{N}_{ik} \setminus \{i\}, \\ 0, & \text{if} \quad j \notin \mathcal{N}_{ik}, \\ 1 - \sum_{\ell \in \mathcal{N}_{ik}} [W_k]_{il}, & \text{if} \quad j = i; \end{cases}$$

Algorithm

Algorithm 1: DCrowdCache

Initialization: arbitrary initial vectors $z_0^{i,-i} \in \mathbb{R}^{N-1}$ and $x_0^i \in \mathbb{R}$
For each time $k = 0, 1, \ldots,$ every MED does the following:

(1) Receives z_k^j and $|\mathcal{N}_{jk}|$ from neighbors $j \in \mathcal{N}_{ik}$;

(2) Sends z_k^i and $|\mathcal{N}_{ik}|$ to neighbors $j \in \mathcal{N}_{ik}$;

(3) Calculates the weights using Metropolis weights:

(4) Updates the action x_{k+1}^i and estimates $z_{k+1}^{i,-i}$ by

$$x_{k+1}^i = \Pi_{X_i}\left[\sum_{j=1}^N [W_k]_{ij}[z_k^j]_i - \alpha \nabla_i J_i\left(\sum_{j=1}^N [W_k]_{ij} z_k^j\right)\right],$$

$$z_{k+1}^{i,-i} = \sum_{j=1}^N [W_k]_{ij} z_k^{j,-i}, \qquad z_{k+1}^{i,i} = x_{k+1}^i;$$

Algorithm

Algorithm 2: DCrowdCache-m

Initialization: arbitrary initial vectors $z_0^{i,-i}, z_{-1}^{i,-i} \in \mathbb{R}^{N-1}$ and $x_0^i, x_{-1}^i \in \mathbb{R}$
For each time $k = 0, 1, \dots,$ every MED does the following:

(1) Receives z_k^j and $|\mathcal{N}_{jk}|$ from neighbors $j \in \mathcal{N}_{ik}$;

(2) Sends z_k^i and $|\mathcal{N}_{ik}|$ to neighbors $j \in \mathcal{N}_{ik}$;

Algorithm

Algorithm 2: DCrowdCache-m

Initialization: arbitrary initial vectors $z_0^{i,-i}, z_{-1}^{i,-i} \in \mathbb{R}^{N-1}$ and $x_0^i, x_{-1}^i \in \mathbb{R}$

For each time $k = 0, 1, \ldots,$ every MED does the following:

(1) Receives z_k^j and $|\mathcal{N}_{jk}|$ from neighbors $j \in \mathcal{N}_{ik}$;

(2) Sends z_k^i and $|\mathcal{N}_{ik}|$ to neighbors $j \in \mathcal{N}_{ik}$;

(3) Calculates the weights using Metropolis weights:

$$[W_k]_{ij} = \begin{cases} 1/(1+\max\{|\mathcal{N}_{ik}|, |\mathcal{N}_{jk}|\}, & \text{if} \quad j \in \mathcal{N}_{ik} \setminus \{i\}, \\ 0, & \text{if} \quad j \notin \mathcal{N}_{ik}, \\ 1 - \sum_{\ell \in \mathcal{N}_{ik}} [W_k]_{il}, & \text{if} \quad j = i; \end{cases}$$

(3.1) Weight on each edge is inversely proportional to the large degree at its two incident nodes;

(3.2) Self-weights are chosen such that the sum of weights at each node is 1.

Algorithm

Algorithm 2: DCrowdCache-m

Initialization: arbitrary initial vectors $z_0^{i,-i}, z_{-1}^{i,-i} \in \mathbb{R}^{N-1}$ and $x_0^i, x_{-1}^i \in \mathbb{R}$
For each time $k = 0, 1, \ldots,$ every MED does the following:

(1) Receives z_k^j and $|\mathcal{N}_{jk}|$ from neighbors $j \in \mathcal{N}_{ik}$;

(2) Sends z_k^i and $|\mathcal{N}_{ik}|$ to neighbors $j \in \mathcal{N}_{ik}$;

(3) Calculates the weights using Metropolis weights:

(4) Updates the action x_{k+1}^i and estimates $z_{k+1}^{i,-i}$ by

$$x_{k+1}^i = \Pi_{X_i}\left[\sum_{j=1}^N [W_k]_{ij}[z_k^j]_i - \alpha \nabla_i J_i\left(\sum_{j=1}^N [W_k]_{ij} z_k^j\right)\right] + \beta_i(x_k^i - x_{k-1}^i)$$

$$z_{k+1}^{i,-i} = \sum_{j=1}^N [W_k]_{ij} z_k^{j,-i} + \beta(z_k^{i,-i} - z_{k-1}^{i,-i}); \quad z_{k+1}^{i,i} = x_{k+1}^i$$

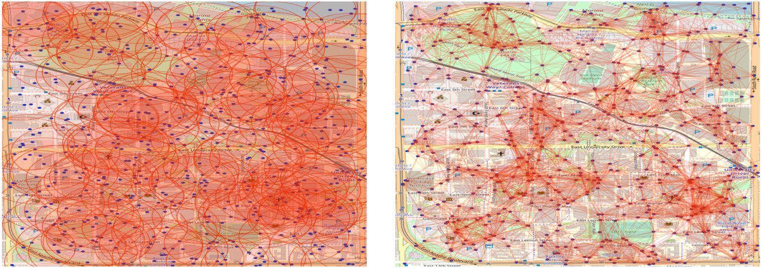
Key takeaways

- **DCrowdCache** and **DCrowdCache-m** share the same structures to compute NE.
- **DCrowdCache-m** integrates the heavy-ball momentum [3,4,5] and consensus-based gradient method, resulting in different updating rules of estimates.

[3] B. Polyak. "Some methods of speeding up the convergence of iteration methods" Comput. Math. & Math. Phys., vol. 4, no. 5, pp. 1–17, 1964

Numerical settings

- Coverage range for each device is within the range of $[150, 200]$ meters [5].
- Generate initial locations of $N = 2^9$ MEDs from the ASU Tempe campus [6].



(a) MEDs coverage areas

(b) MEDs connections

Figure: Mobile edge devices (MEDs) locations at ASU campus

Numerical settings

- Coverage range for each device is within the range of $[150, 200]$ meters [5].
- Generate initial locations of $N = 2^9$ MEDs from the ASU Tempe campus [6].
- In each iteration, the MEDs randomly move within the designated range, and their new locations and coverage ranges are used to create a communication graph among them.
- $Q_i, \forall i$ is randomly generated from $U[0.01, 0.1]\$$ per hour.
- $h_i, \forall i$ is randomly generated from $U[0.05, 0.15]\$$ per hour.
- $c_i, \forall i$ is randomly from the values 16, 32, 48, and 64 GB.
- $\bar{P}$ for each unit of storage is set to 1 (normalization).

[5] https://github.com/swinedge/eua-dataset

[6] https://github.com/duongnguyen1601/CrowdCache-dataset

Performance analysis

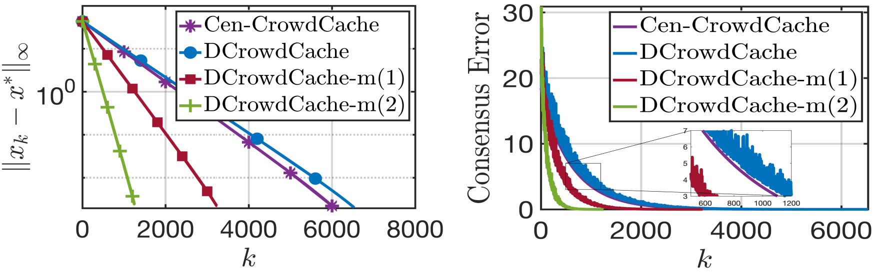


Figure: Convergence properties: The numerical results are computed with $\alpha = 20$. **Cen-CrowdCache**: Centralized communication that braod casts information from all MEDs; **DCrowdCache-m(1)**: $\beta = 0.5$; **DCrowdCache-m(2)**: $\beta = 0.8$

- **DCrowdCache-m** converges faster than the **DCrowdCache**
- Larger momentum (β) leads to faster convergence.

Sensitivity analysis

- **Proposed**: proposed decentralized NE seeking algorithm.
- **Heuristic**: use 20% of the maximum idle resources from MEDs, as long as the utility is positive.
- **Average**: use 50% of the maximum idle resources from MEDs, as long as the utility is positive.

Sensitivity analysis

Γ: scaling factor to scale down/up γ regarding base value generated in the default setting.

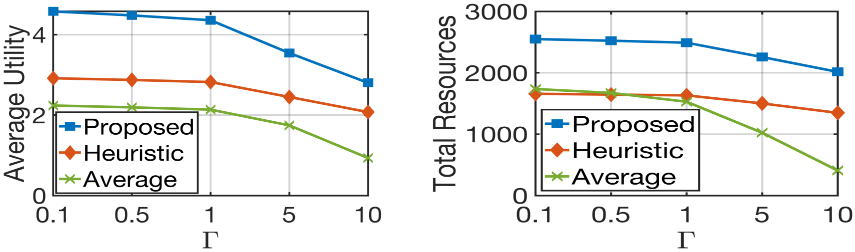


Figure: Impacts of γ

Sensitivity analysis

Γ: scaling factor to scale down/up γ regarding base value generated in the default setting.

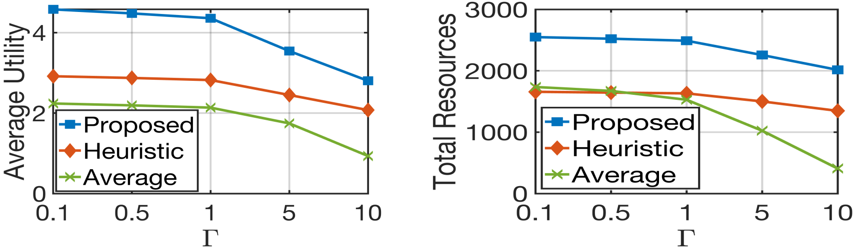


Figure: Impacts of γ

- Decreasing γ leads to a higher utility: the unit reward increases, motivating more MEDs to contribute their resources to the system.

Sensitivity analysis

Γ: scaling factor to scale down/up γ regarding base value generated in the default setting.

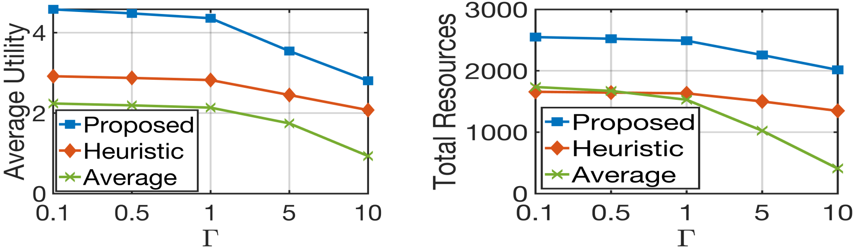


Figure: Impacts of γ

- Decreasing γ leads to a higher utility: the unit reward increases, motivating more MEDs to contribute their resources to the system.
- Increasing γ leads to a lower utility: discourages MEDs to reduce their contributions, avoiding over-provisioning cost

Sensitivity analysis

Λ: scaling factor to scale down/up the maximum unit price ($\bar{P}$) regarding base value generated in the default setting.

Sensitivity analysis

Λ: scaling factor to scale down/up the maximum unit price ($\bar{P}$) regarding base value generated in the default setting.

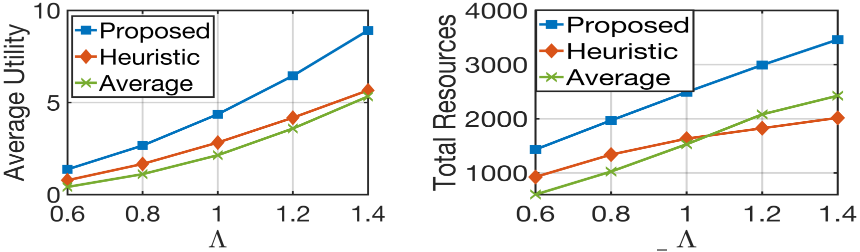


Figure: Impacts of maximum unit price $\bar{P}$

Sensitivity analysis

Λ: scaling factor to scale down/up the maximum unit price ($\bar{P}$) regarding base value generated in the default setting.

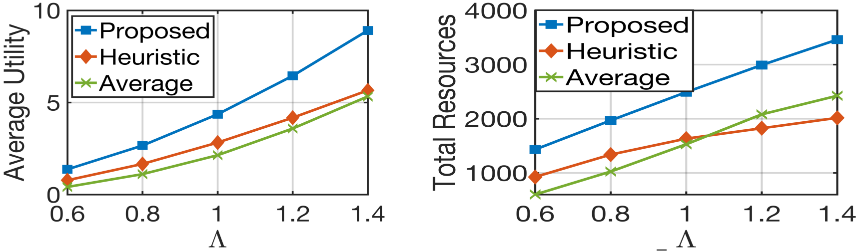


Figure: Impacts of maximum unit price $\bar{P}$

- Increasing $\bar{P}$: incentivize the MED to supply more resources to the platform, resulting in a higher average utility for the MEDs.

Sensitivity analysis

Λ: scaling factor to scale down/up the maximum unit price ($\bar{P}$) regarding base value generated in the default setting.

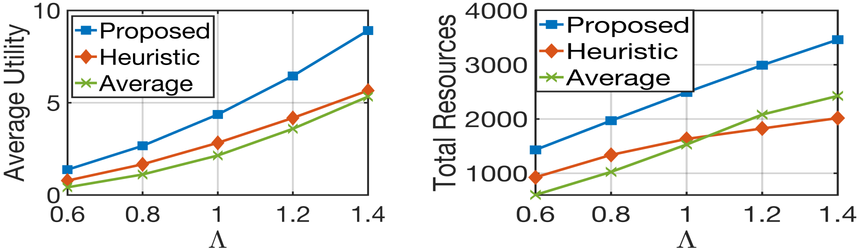


Figure: Impacts of maximum unit price $\bar{P}$

- Increasing $\bar{P}$: incentivize the MED to supply more resources to the platform, resulting in a higher average utility for the MEDs.
- Increasing $\bar{P}$: access a larger resource pool, leading to operational scalability, improved user experience.

Sensitivity analysis

Ψ: scaling factor to scale down/up the quadratic cost parameter (Q_i) regarding base value generated in the default setting.

Sensitivity analysis

Ψ: scaling factor to scale down/up the quadratic cost parameter (Q_i) regarding base value generated in the default setting.

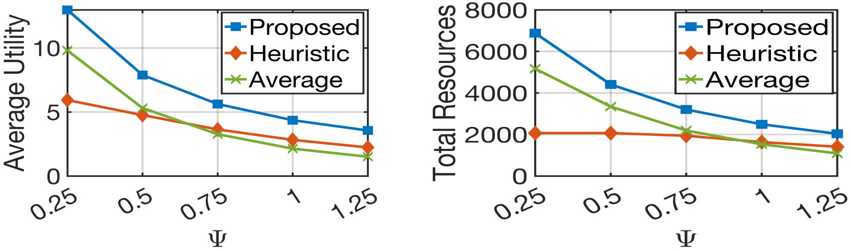


Figure: Impacts of quadratic cost Q_i, for all $i \in \mathcal{I}$

Sensitivity analysis

Ψ: scaling factor to scale down/up the quadratic cost parameter (Q_i) regarding base value generated in the default setting.

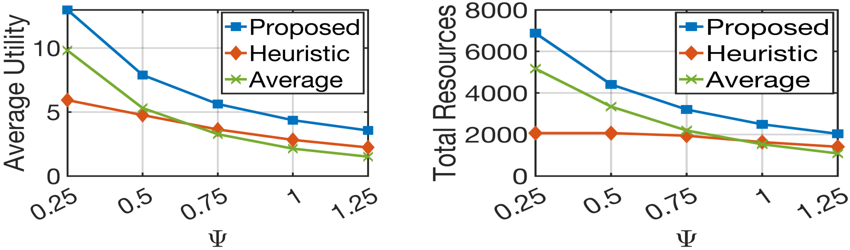


Figure: Impacts of quadratic cost Q_i, for all $i \in \mathcal{I}$

- Increasing Q_i: MED may reduce the amount of outsourcing resources, leading to a decrease in the amount of idle resources available for outsourcing,

Sensitivity analysis

Ψ: scaling factor to scale down/up the quadratic cost parameter (Q_i) regarding base value generated in the default setting.

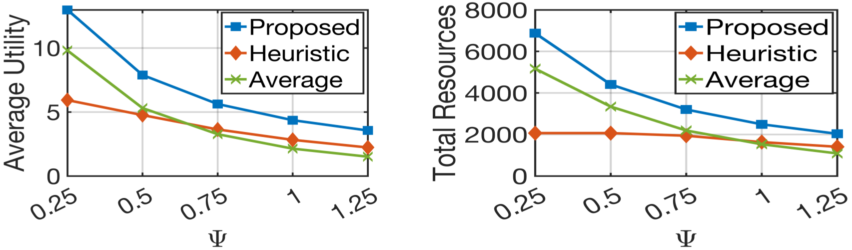


Figure: Impacts of quadratic cost Q_i, for all $i \in \mathcal{I}$

- Increasing Q_i: MED may reduce the amount of outsourcing resources, leading to a decrease in the amount of idle resources available for outsourcing,
- Quadratic cost coefficient leads to more rapid decrease in the marginal utility of outsourcing additional resources

Sensitivity analysis

δ: scaling factor to scale down/up the linear cost parameter (h_i) regarding base value generated in the default setting.

Sensitivity analysis

δ: scaling factor to scale down/up the linear cost parameter (h_i) regarding base value generated in the default setting.

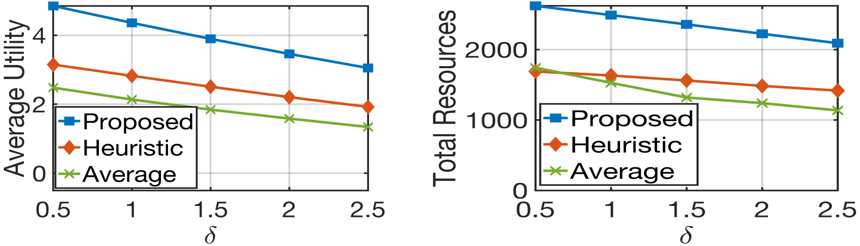


Figure: Impacts of linear cost h_i, for all $i \in \mathcal{I}$

Sensitivity analysis

δ: scaling factor to scale down/up the linear cost parameter (h_i) regarding base value generated in the default setting.

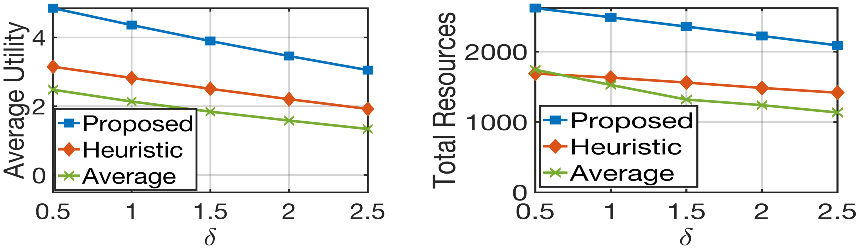


Figure: Impacts of linear cost h_i, for all $i \in \mathcal{I}$

- Quadratic cost coefficient leads to more rapid decrease in the marginal utility of outsourcing additional resources

Numerical results

N	**DCrowdCache**		**DCrowdCache-m** ($\beta = 0.5$)	
	# Iterations	Run time (s)	# Iterations	Run time (s)
2^8	2973	0.035	1474	0.018
2^9	5791	0.099	2876	0.053
2^{10}	10812	0.437	5407	0.239
2^{11}	19951	1.519	9978	0.858
2^{12}	37541	8.482	18734	4.464

Table: Average performance over 1000 simulations.

Conclusions and future work

Conclusions:

- Introduced a novel privacy-preserving framework to tackle the decentralized mobile edge content caching and sharing problem, where MEDs can share their cached content with neighbors.
- Operated over a time-varying communication graph, ensuring that users' privacy is preserved throughout the process.
- Examined the convergence of the algorithm over a time-varying undirected communication network.

Conclusions and future work

Conclusions:

- Introduced a novel privacy-preserving framework to tackle the decentralized mobile edge content caching and sharing problem, where MEDs can share their cached content with neighbors.
- Operated over a time-varying communication graph, ensuring that users' privacy is preserved throughout the process.
- Examined the convergence of the algorithm over a time-varying undirected communication network.

Future work:

- Consider more complicated local objective function and its impact in context of this problem
- Explore more complex generalized game models containing local constraints and coupling constraints among agents' decisions.

THANK YOU!

Notations

- $z_i^k = (z_{i1}^k, \ldots, z_{im}^k)' \in \mathbb{R}^n$: agent i's estimate of the joint action x
- $\{\pi_k\}$: sequence of stochastic vectors satisfying $\pi_{k+1}' W_k = \pi_k'$ (see Lemma 2 in [2])
- $\hat{z}_{\pi_k}^k = \sum_{i=1}^m [\pi_k]_i z_i^k$: weighted-average of the estimates
- x^*: an NE point of the game
- We define the matrices:

$$\mathbf{z}^k = \begin{bmatrix} (z_1^k)' \\ (z_2^k)' \\ \vdots \\ (z_m^k)' \end{bmatrix} = \begin{bmatrix} x_1^k & z_{12}^k & \ldots & z_{1m}^k \\ z_{21}^k & x_2^k & \ldots & z_{2m}^k \\ \vdots & \vdots & \ddots & \vdots \\ z_{m1}^k & z_{m2}^k & \ldots & x_m^k \end{bmatrix},$$

$$\hat{\mathbf{z}}^k = \mathbf{1}_m (\hat{z}_{\pi_k}^k)', \quad \mathbf{x}^* = \mathbf{1}_m (x^*)'$$

[2] Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs.