Dynamic Resource Caching in the IoT Application Layer for Smart Cities

Xiang Sun, *Student Member, IEEE,* and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—We propose to apply CoAP Publish/Subscribe to cache popular IoT resources in a broker to reduce the energy consumption of servers (which host these popular resources) and the average delay for delivering the IoT resources' contents to the clients. We provide the smart parking application in smart cities as an example to demonstrate the benefit for conducting popular IoT resource caching. However, caching popular IoT resources in the broker may not always be the optimal choice, i.e., the broker may be congested by caching too many IoT resources, and so the average delay for enabling the broker to deliver contents of the IoT resources may be unbearable. Thus, we propose a novel Energy Aware and latency guaranteed dynamic reSourcE caching (EASE) strategy to enable the broker to cache suitable popular resources such that the energy savings from the servers are maximized, while the average delay for publishing the contents of the resources to the corresponding clients is minimized. We demonstrate the performances of EASE via simulations as compared to other two baseline IoT resource caching strategies.**

***Index Terms*—Internet of things, caching, CoAP, CoAP Publish/Subscribe, broker**

I. Introduction

Internet of Things (IoTs) enables sensors to transmit their sensed data and actuators to be controlled so as to facilitate users to understand and change the physical world. Basically, the IoT architecture comprises three layers, i.e., the perception, network, and application layer [1], [2]. Specifically, the perception layer represents the physical IoT devices, which perform different functionalities directly related to the hardware, e.g., temperature sensors capture current surrounding temperature values and the switch automatically turns off the light. The perception layer digitizes and transmits the data to/from the network layer, which provides connectivity among different IoT devices by applying different communications technologies. The application layer provides various functionalities (such as resource discovery and data management) and interfaces to access different hardware resources and provision smart services to customers. Currently, Constrained Application Protocol (CoAP) and CoAP Publish/Subscribe (CoAP Pub/Sub) are the most widely used IoT application layer protocols.

CoAP [3] is originally designed for communications among resource constrained devices. CoAP assumes two logical roles, i.e., client and server. A client is a resource requester, which sends a resource retrieval request to the server; a server is a resource host, which maintains the resource and responds to the resource retrieval request from the client. A resource refers to a physical phenomenon sensed by a server. For instance, a temperature sensor (i.e., a server) is to sense the temperature value of Bob's smart home and the current temperature value is 30^oC. Then, "the temperature value of Bob's smart home" is a resource hosted by the temperature sensor and "30^oC" is the related IoT content. As shown in Fig. 1, CoAP applies a simple request/response model to manipulate resources.

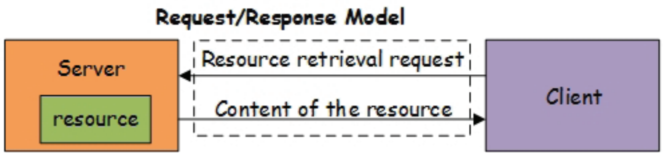


Fig. 1: CoAP communications model.

CoAP Pub/Sub [4] plays three logical roles in communications, i.e., client, server, and broker. As shown in Fig. 2, if a server prefers to cache its hosting resource, it can send a resource caching request to a broker, which refers to an intermediary node that is capable of caching resource; the broker will cache the resource[1] if it receives the request from the server; consequently, the servers can update the contents of their resources cached in the broker, which forwards the contents to the clients upon requests. For instance, the temperature sensor (i.e., the server) in Bob's smart home can request to cache the "temperature value of Bob's smart home" (resource) in a broker (e.g., the home gateway). After the resource has been cached in the broker, the temperature sensor can publish the up-to-date content of the resource (e.g., "30^oC") to the resource cached in the broker, which will store and forward the content to the clients (e.g., Bob's mobile phone) upon requests. Each physical entity with computing, communications, and storage capabilities (such as a cloudlet [5], [6], a fog node [7], etc.) could be a potential broker.

The initial motivation for designing CoAP Pub/Sub is to facilitate clients to retrieve the content of a resource hosted by a sleep-enabled server [4], i.e., the clients are still able to access the resources via the broker when the servers (which host those resources) are sleeping. Applying CoAP Pub/Sub offers other unveiled benefits, i.e., applying CoAP Pub/Sub to

X. Sun and N. Ansari are with Advanced Networking Lab., New Jersey Institute of Technology, Newark, NJ 07102, USA. E-mail:{xs47, nirwan.ansari}@njit.edu.

[1]"A broker is caching a resource" means that the broker is creating a URI of the resource such that any operation on the resource can be performed via the URI (e.g., the server accesses the URI to update the content of the resource in the broker, and the clients access the URI to retrieve the content of the resource in the broker).

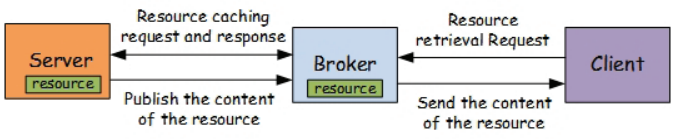

Server
resource
Resource caching request and response
Publish the content of the resource
Broker
resource
Resource retrieval Request
Send the content of the resource
Client

Fig. 2: CoAP Pub/Sub communications model.

cache popular resources in the broker can reduce the energy consumption of the corresponding servers and accelerate the content delivery rate. In this paper, we are focusing on analyzing the pros and cons of caching resources in the broker and designing an optimal resource caching strategy in order to determine which resources are suitable to be cached in the broker. The contributions of the paper are summarized as follows. 1) We propose to apply CoAP Pub/Sub to achieve the IoT resource caching. To the best of our knowledge, this is the first introduction of the resource caching in the IoT application layer. 2) We demonstrate that caching the IoT resources in the broker is not always the best choice to save energy of the servers and reduce the average delay to publish contents of the resources. 3) We propose a novel dynamic resource caching strategy to maximize the energy savings from servers and guarantee the minimum average delay for publishing contents of the resources to the corresponding clients. 4) We demonstrate the performance of the proposed dynamic resource caching strategy via simulations.

The rest of paper is organized as follows. In Section II, we explain the pros and cons for caching resources in the broker, define the popular resource, and design a broker-side resource caching strategy. In Section III, we demonstrate the performance of the proposed resource caching strategy via simulations. In Section IV, we briefly review the related works, followed by conclusion in Section V.

II. Resource caching in IoT application layer

Essentially, "a resource is cached in the broker" implies that CoAP Pub/Sub is applied to enable the broker to deliver the content of the resource to the clients; otherwise, CoAP is applied to enable the server (which hosts the resource) to deliver the content of the resource to the clients. We next provide the smart parking use case to illustrate the pros and cons for caching resources in the broker.

A. Use case

In smart cities, there are many street parking spots located near a stadium. Each street parking spot is equipped with a smart parking meter, which generates the parking spot status indicating whether the parking spot is empty and when it will become available if it is currently occupied. Assume there is a broker located near the stadium and available for all the street parking meters. If there is a football game hosted in the stadium, as shown in Fig. 3, tens of thousands of smart cars would look for the available parking spots near the stadium before the game starts. Consequently, each parking meter would receive a large number of resource retrieval requests from smart cars and need to respond to them accordingly. This would tremendously increase the energy consumption of the parking meter, which may be powered by its battery.

Now, if each parking meter caches its resource in a broker by periodically sending the up-to-date content of the resource to the broker and let the broker respond to the resource retrieval requests from the smart cars, the parking meter can save a huge amount of energy by sending less amount of data. The broker normally has abundant power supply and powerful hardware. In other words, the transmission rate of the broker should be much higher than that of a parking meter, and thus caching the parking spot status resources in the broker can reduce the average delay for delivering contents of the resources to the clients.

Caching the resource in the broker may not always be the optimal solution. Consider the case that the broker caches many resources and needs to handle a huge number of resource retrieval requests from the clients; consequently, the average delay for the broker in delivering contents of the resources may be unbearable. Consider another case when the football game finishes, as shown in Fig. 4; smart cars are no longer interested in the parking spot status resources and some parking spots are still occupied. If these parking meters do not cache their parking spot status resources in the broker, they do not need to transmit any packet since nobody is interested in these resources. Yet, if the parking meters cache their parking spot status resources in the broker, these parking meters need to transmit their up-to-date statuses to the broker. Therefore, without caching the resource in the broker may consume less energy in this scenario. Therefore, caching the resource in the broker by applying CoAP Pub/Sub may not always benefit the servers. It is necessary to determine whether to cache the resources in the broker or not based on different scenarios.

B. The definition of a popular resource

Caching a resource in the broker can save energy of the server iff the resource is popular. Thus, it is important to define a popular resource.

Denote $\mathcal{I}$ as the set of resources in the network, i as the index of resource, l_i as the average content size of resource i, $\bar{\lambda}_i$ as the average arrival rate of resource retrieval request (i.e., the average number of resource retrieval requests per second during a time slot) for resource i, $\bar{\eta}_i$ as the average content delivery rate of resource i (i.e., the number of times that the server delivers the up-to-date content of resource i to the broker per second during a time slot) when the resource is cached in the broker, and ΔT as the duration of a time slot.

As shown in Fig. 5, if resource i is not cached in the broker, the server needs to transmit $l_i\bar{\lambda}_i\Delta T$ amount of data in order to publish contents of resource i to the clients. On the other hand, if resource i is cached in the broker, the server needs to send $l_i\bar{\eta}_i\Delta T$ amount of data to the broker and let the broker publish contents of resource i to the clients. Apparently, if Eq. 1 is satisfied, the server may request to cache resource i in the broker, and vice versa.

$$\epsilon_i l_i \Delta T\left(\bar{\lambda}_i - \bar{\eta}_i\right) > \theta. \tag{1}$$

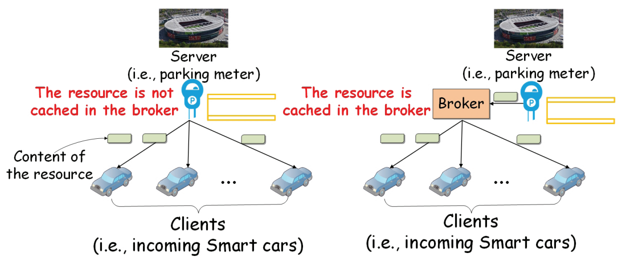

Server
(i.e., parking meter)
The resource is not cached in the broker
Content of the resource
Clients
(i.e., incoming Smart cars)
Server
(i.e., parking meter)
The resource is cached in the broker
Broker
Clients
(i.e., incoming Smart cars)

Fig. 3: The use case of a street parking meter nearby a stadium before a football game starts.

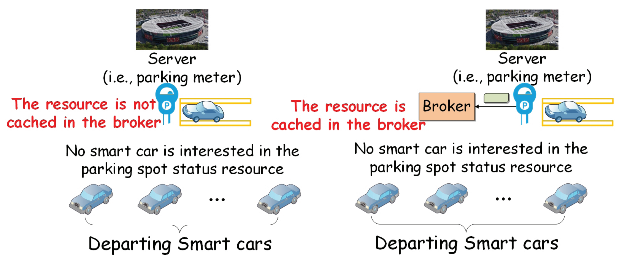

Server
(i.e., parking meter)
The resource is not cached in the broker
No smart car is interested in the parking spot status resource
Departing Smart cars
Server
(i.e., parking meter)
The resource is cached in the broker
Broker
No smart car is interested in the parking spot status resource
Departing Smart cars

Fig. 4: The use case of a street parking meter nearby a stadium after a football game.

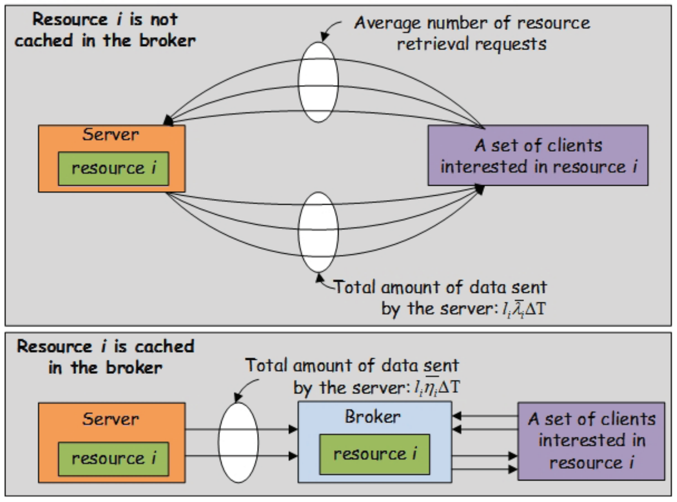

Resource i is not cached in the broker
Average number of resource retrieval requests
Server
resource i
A set of clients interested in resource i
Total amount of data sent by the server: $l_i\bar{\lambda}_i\Delta T$
Resource i is cached in the broker
Total amount of data sent by the server: $l_i\bar{\eta}_i\Delta T$
Server
resource i
Broker
resource i
A set of clients interested in resource i

Fig. 5: The amount of data sent by a server.

Here, ϵ_i is the energy coefficient of the server (which hosts resource i) that maps transmitting one bit of data into energy consumption and θ is a predefined energy threshold. Thus, we define resources which satisfy Eq. 1 as popular resources (i.e., $\mathcal{J} = \left\{i|\epsilon_i l_i \Delta T\left(\bar{\lambda}_i - \bar{\eta}_i\right) > \theta, i \in \mathcal{I}\right\}$, where $\mathcal{J}$ is the set of popular resources in the network) and define resources, which do not satisfy Eq. 1, as unpopular resources. Note that a server would send a resource caching request to the broker if its resource becomes popular.

C. *Resource caching strategy in the broker*

The broker may be congested if it caches all the popular resources such that the broker needs to transmit a huge volume of data for delivering the cached resources' contents. Consequently, the average delay for enabling the broker to deliver a content of a resource may be longer than the average delay for enabling a server itself to deliver a content of its hosting resource. Therefore, it is beneficial to design a dynamic resource caching mechanism for the broker to determine which popular resource should be cached in the broker in order to maximize the total energy savings from servers, while guaranteeing the average delay.

1) Average delay for the broker to deliver the content of the popular resource: The broker should estimate the average delay for publishing a popular resource's content to respond to the corresponding resource retrieval request. Denote j as the index of these popular resources. Let x_j to be the binary variable indicating whether popular resource j should be cached in broker (i.e., $x_j = 1$) or not (i.e., $x_j = 0$); thus, $\mathcal{X} = \{x_j|j \in \mathcal{J}\}$ denotes the popular resource caching strategy adopted by the broker. Meanwhile, we assume that the content size of the popular resources, i.e., $\{l_j|j \in \mathcal{J}\}$, follows a Poisson distribution and $\bar{l}$ is the average content size among all the popular resources. Consequently, the service rate of the broker (the average number of resource retrieval requests handled by the broker per second) also follows a Poisson distribution and $\frac{\bar{l}}{u^b}$ is the average service rate of the broker, where u^b is the average transmission rate of the broker. In addition, assume that the resource retrieval request arrivals for each resource during a time slot exhibits a Poisson distribution and $\bar{\lambda}_j$ is the average arrival rate of resource retrieval request for resource j. Then, we can model the broker's publishing contents of the cached popular resources in response to the resource retrieval requests from the clients as an M/M/1 queueing model, and so the average delay (i.e., the average queueing delay plus the average transmission delay) for the broker's publishing a popular resource's content in response to the resource retrieval request can be expressed[2]:

$$t^b = \frac{1}{\frac{u_b}{\bar{l}} - \sum\limits_{j \in \mathcal{J}} \bar{\lambda}_j x_j}. \tag{2}$$

2) Average delay for the server to deliver the content of the resource: The broker estimates the average delay if the content of the popular resource is delivered by the server. If the average delay to deliver the content of the resource by the server is lower than that by the broker, this popular resource is not suitable to be cached in the broker.

Assume each server hosts one resource and denote u_j^s as the average transmission rate of the server, which hosts popular resource j. Since the content size of popular resource j, i.e., the value of l_j, is normally fixed over time, the average service rate for this server in handling the resource retrieval requests is deterministic, which is $\frac{u_j^s}{l_j}$. Meanwhile, since the resource retrieval request arrival rate for popular resource j exhibits a Poisson distribution with the average arrival rate of $\bar{\lambda}_j$, we model the server in delivering the content of resource j to the

[2]The average delay of a broker is the average queueing delay of a resource's content waiting in the broker's network queue plus the average transmission delay of the broker in sending the content out of its network interface. The propagation delay for transmitting the content to the client over the network is not considered.

clients as an M/D/1 queueing model, and so the average delay for the server in delivering resource j's content to a client, denoted as t_j^s, can be expressed:

$$t_j^s = \frac{l_j}{2u_j^s} \times \frac{\overline{\lambda_j}}{\frac{u_j^s}{l_j} - \overline{\lambda_j}}. \tag{3}$$

3) Problem formulation: We formulate the dynamic resource caching problem (i.e., $\boldsymbol{P0}$) in the broker as follows:

$$\begin{aligned} \boldsymbol{P0}: \quad & \arg\max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) x_j, & (4)\\ s.t. \quad & \forall j \in \boldsymbol{\mathcal{J}},\ \frac{x_j}{\frac{u_b}{\bar{l}} - \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \overline{\lambda_j} x_j} \le t_j^s, & (5)\\ & \sum_{j\in\boldsymbol{\mathcal{J}}} \overline{\lambda_j} x_j < \frac{u_b}{\bar{l}}, & (6)\\ & \forall j \in \boldsymbol{\mathcal{J}},\ x_j \in \{0,1\}. & (7) \end{aligned}$$

$\boldsymbol{P0}$ is to minimize the energy consumption of servers. Constraint (5) is to guarantee $t^b \le t_j^s$ for all the resources cached in the broker. Constraint (6) is to ensure the system is stable, i.e., the average arrival rate should be less than the average service rate in the broker to assure the queue does not overflow. Constraint (7) imposes x_j to be a binary variable.

Theorem 1. $\boldsymbol{P0}$ is NP-hard.

Proof: By combining Constraints (5) and (6), $\boldsymbol{P0}$ can be transformed into:

$$\begin{aligned} \boldsymbol{P1}: \quad & \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}}) = \arg\max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i \Delta T l_i \left(\bar{\lambda}_i - \bar{\eta}_i\right) x_j, \\ & s.t.\ \ \forall j \in \boldsymbol{\mathcal{J}},\ \sum_{j\in\boldsymbol{\mathcal{J}}} \overline{\lambda_j} x_j + \frac{1}{t_j^s} x_j \le \frac{u_b}{\bar{l}}, \quad (8)\\ & \forall j \in \boldsymbol{\mathcal{J}},\ x_j \in \{0,1\}. \end{aligned}$$

Consider the case that Constraint (8) is equivalent to $\sum\limits_{j\in\boldsymbol{\mathcal{J}}}\left(\sum\limits_{j\in\boldsymbol{\mathcal{J}}} \overline{\lambda_j} x_j + \frac{1}{t_j^s} x_j\right) \le \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \frac{u_b}{\bar{l}}$, which is transformed into:

$$\sum_{j\in\boldsymbol{\mathcal{J}}} \left(|\boldsymbol{\mathcal{J}}|\,\overline{\lambda_j} + \frac{1}{t_j^s}\right) x_j \le \frac{|\boldsymbol{\mathcal{J}}|\, u^b}{\bar{l}}. \tag{9}$$

$\boldsymbol{P1}$ can be transformed into:

$$\begin{aligned} \boldsymbol{P2}: \quad & \arg\max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) x_j, \\ s.t. \quad & Constraints\ (7)\ and\ (9). & (10) \end{aligned}$$

Obviously, $\boldsymbol{P2}$ is the 0-1 knapsack problem, where $\epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right)$ is the value of item j, $|\boldsymbol{\mathcal{J}}|\,\overline{\lambda_j} + \frac{1}{t_j^s}$ is the weight of item j, and $\frac{|\boldsymbol{\mathcal{J}}|u^b}{\bar{l}}$ is the weight capacity of the knapsack. Note that the 0-1 knapsack problem is a well known NP-hard problem. Therefore, we conclude that the 0-1 knapsack problem is reducible to the original dynamic resource caching problem (i.e., $\boldsymbol{P0}$), and so $\boldsymbol{P0}$ is NP-hard. ■

4) Energy Aware and latency guaranteed dynamic reSourcE caching (EASE): We propose EASE to solve $\boldsymbol{P1}$[3]. Specifically, we relax Constraint (8) to construct the following Lagrangian problem of $\boldsymbol{P1}$:

$$\begin{aligned} \boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}}) = & \max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \left(\epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) - \bar{\lambda}_j \sum_{j\in\boldsymbol{\mathcal{J}}} \omega_j - \frac{\omega_j}{t_j^s} \right) x_j \\ & + \left(\frac{u_b}{\bar{l}}\right) \sum_{j\in\boldsymbol{\mathcal{J}}} \omega_j, \qquad (11)\\ s.t. \quad & Constraint\ (7), \end{aligned}$$

where $\boldsymbol{\mathcal{W}} = \{\omega_j \ge 0 | j \in \boldsymbol{\mathcal{J}}\}$ are the Lagrangian multipliers. Note that the above Lagrangian problem will have the optimal solution $\boldsymbol{\mathcal{X}}^* = \left\{x_j^* | j \in \boldsymbol{\mathcal{J}}\right\}$, where:

$$x_j^* = \begin{cases} 1, & \epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) - \bar{\lambda}_j \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \omega_j - \frac{\omega_j}{t_j^s} > 0. \\ 0, & \epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) - \bar{\lambda}_j \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \omega_j - \frac{\omega_j}{t_j^s} \le 0. \end{cases} \tag{12}$$

Lemma 1. $\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}})$ provides an upper bound on $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$.

Proof: Assume $\hat{\boldsymbol{\mathcal{X}}} = \{\hat{x}_j | j \in \boldsymbol{\mathcal{J}}\}$ is a feasible solution of $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$. Thus, $\forall j \in \boldsymbol{\mathcal{J}}$, $\sum\limits_{j\in\boldsymbol{\mathcal{J}}} \left(\overline{\lambda_j} \hat{x}_j\right) + \frac{1}{t_j^s}\hat{x}_j - \frac{u_b}{\bar{l}} \le 0$. Since $\forall j \in \boldsymbol{\mathcal{J}}, \omega_j \ge 0$, we can derive:

$$\sum_{j\in\boldsymbol{\mathcal{J}}} \left(\omega_j \left(\sum_{j\in\boldsymbol{\mathcal{J}}} \left(\overline{\lambda_j} \hat{x}_j\right) + \frac{1}{t_j^s} \hat{x}_j - \frac{u_b}{\bar{l}} \right) \right) \le 0,$$

i.e.,

$$\begin{aligned} & \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) \hat{x}_j - \sum_{j\in\boldsymbol{\mathcal{J}}} \left(\omega_j \left(\sum_{j\in\boldsymbol{\mathcal{J}}} \left(\overline{\lambda_j} \hat{x}_j\right) + \frac{1}{t_j^s} \hat{x}_j - \frac{u_b}{\bar{l}} \right) \right) \\ & \ge \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T \left(\bar{\lambda}_i - \bar{\eta}_i\right) \hat{x}_j, \end{aligned}$$

which implies that $\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}}) \ge \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}})$. ■

Since $\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}})$ is an upper bound on $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$, the next step is to select suitable values of $\boldsymbol{\mathcal{W}}$ such that the gap between $\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}})$ and $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$ is as small as possible. We apply the subgradient method to iteratively select the values of $\boldsymbol{\mathcal{W}}$ in order to find the minimum gap. Specifically, we first select the initial values of $\boldsymbol{\mathcal{W}}$, denoted as $\boldsymbol{\mathcal{W}}^{(0)} = \left\{\omega_j^{(0)} \ge 0 | j \in \boldsymbol{\mathcal{J}}\right\}$. Then, $\boldsymbol{\mathcal{W}}$ are updated in each iteration based on the following equation:

$$\begin{aligned} \forall j \in \boldsymbol{\mathcal{J}}, \omega_j^{(k+1)} &= \omega_j^{(k)} - \alpha^{(k)} \frac{\partial \boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}}^{(k)})}{\partial \omega_j^{(k)}} \\ &= \omega_j^{(k)} + \alpha^{(k)} \left(\sum_{j\in\boldsymbol{\mathcal{J}}} \left(\overline{\lambda_j} x_j^{*(k)}\right) + \frac{1}{t_j^s} x_j^{*(k)} - \frac{u_b}{\bar{l}} \right), \end{aligned} \tag{13}$$

where $\omega_j^{(k)}$ is the value of ω_j in the k^{th} iteration; $x_j^{*(k)}$, which is calculated based on Eq. 12, is the optimal solution of the Lagrangian problem in the k^{th} iteration (note that $x_j^{*(k)}$ may not be the feasible solution of $\boldsymbol{P1}$ since $x_j^{*(k)}$ may not satisfy

[3]As mentioned previously, $\boldsymbol{P1}$ is equivalent to $\boldsymbol{P0}$. Thus, we will try to solve $\boldsymbol{P1}$ in the remaining paper.

Constraint(8); thus, we have to map the optimal solution of the Lagrangian problem into the feasible solution of $\boldsymbol{P1}$, i.e., $\hat{x}_j^{(k)} = \boldsymbol{\mathcal{M}}(x_j^{*(k)})$, where $\hat{x}_j^{(k)}$ is the feasible solution of $\boldsymbol{P1}$ in the k^{th} iteration and $\boldsymbol{\mathcal{M}}(\cdot)$ specifies the mapping); $\alpha^{(k)}$ is the step size adopted in the k^{th} iteration [8]:

$$\alpha^{(k)} = \beta \frac{\boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{W}}^{(k)}\right) - \boldsymbol{\mathcal{F}}^{max}}{\sum\limits_{j\in\boldsymbol{\mathcal{J}}}\left(\sum\limits_{j\in\boldsymbol{\mathcal{J}}}\left(\overline{\lambda_j}\hat{x}_j^{(k)}\right) + \frac{1}{t_j^s}\hat{x}_j^{(k)} - \frac{u_b}{l}\right)^2}, \tag{14}$$

where β is a decreasing adaption parameter with $0 < \beta < 2$ and $\boldsymbol{\mathcal{F}}^{max}$ is the maximum objective value for $\boldsymbol{P1}$ found so far, i.e., $\boldsymbol{\mathcal{F}}^{max} = \max\left\{\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(1)}), \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(2)}), ..., \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(k)})\right\}$ (where $\hat{\boldsymbol{\mathcal{X}}}^{(k)} = \left\{\hat{x}_j^{(k)} | j \in \boldsymbol{\mathcal{J}}\right\}$ is the feasible solution calculated in the k^{th} iteration and $\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(k)})$ is the corresponding objective value for $\boldsymbol{P1}$).

The values of $\boldsymbol{\mathcal{W}}$ continue to be updated until the gap between the Lagrangian problem and the maximum objective value for $\boldsymbol{P1}$, i.e., $\boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{W}}^{(k)}\right) - \boldsymbol{\mathcal{F}}^{max}$, does not change over iterations. EASE is summarized in Algorithm 1.

Algorithm 1 The EASE algorithm

```
1:  Calculate $\boldsymbol{\mathcal{T}}^s = \{t_j^s | j \in \boldsymbol{\mathcal{J}}\}$ based on Eq. 3.
2:  Initialize $\boldsymbol{\mathcal{W}} = \mathbf{0}$.
3:  Calculate $\boldsymbol{\mathcal{X}}^* = \{x_j^* | j \in \boldsymbol{\mathcal{J}}\}$ based on Eq. 12.
4:  Calculate $\hat{\boldsymbol{\mathcal{X}}} = \{\hat{x}_j | j \in \boldsymbol{\mathcal{J}}\}$, where $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{X}}^*)$.
5:  Initialize $\boldsymbol{\mathcal{F}}^{max} = \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}})$ and $\boldsymbol{\mathcal{X}} = \hat{\boldsymbol{\mathcal{X}}}$.
6:  Calculate the value of $\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}})$ based on Eq. 11.
7:  while $\boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}}) - \boldsymbol{\mathcal{F}}^{max}$ changes over the iterations do
8:      Update the step size $\alpha$ based on Eq. 14;
9:      Update the values of $\boldsymbol{\mathcal{W}}$ based on Eq. 13;
10:     Update the values of $\boldsymbol{\mathcal{X}}^*$ based on Eq. 12;
11:     Calculate the values of $\hat{\boldsymbol{\mathcal{X}}}$, where $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{X}}^*)$.
12:     if $\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}) > \boldsymbol{\mathcal{F}}^{max}$ then
13:         $\boldsymbol{\mathcal{F}}^{max} = \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}})$ and $\boldsymbol{\mathcal{X}} = \hat{\boldsymbol{\mathcal{X}}}$.
14:     end if
15: end while
16: return $\boldsymbol{\mathcal{X}}$.
```

As mentioned previously, the mapping function $\boldsymbol{\mathcal{M}}(\cdot)$ involved in Algorithm 1 is to convert the optimal solution of the Lagrangian problem (i.e., $\boldsymbol{\mathcal{X}}^*$) into the feasible solution of the primal problem (i.e., $\boldsymbol{P1}$) such that Constraint 8 is satisfied by all $j \in \boldsymbol{\mathcal{J}}$. The basic idea is that the broker iteratively drops a suitable popular resource (i.e., the popular resource is not selected to be cached by the broker) among all the popular resources that are currently cached by the broker (which are calculated by Eq. 12) until Constraint 8 is satisfied by all $j \in \boldsymbol{\mathcal{J}}$. The suitable popular resource, denoted as j', is defined as the popular resource that incurs the minimum average delay (for enabling its server to deliver the content of the resource) among all the currently cached popular resources, i.e.,

$$j' = \arg\max_j \left\{t_j^s | x_j^* = 1, j \in \boldsymbol{\mathcal{J}}\right\}. \tag{15}$$

The mapping function is summarized in Algorithm 2.

Algorithm 2 The mapping function $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{X}}^*)$.

```
1: Obtain the popular resource set $\boldsymbol{\mathcal{J}}' = \{j | x_j^* = 1, j \in \boldsymbol{\mathcal{J}}\}$.
2: Find the suitable resource $j'$ based on Eq.15.
3: while $\forall j \in \boldsymbol{\mathcal{J}}, \sum_{j\in\boldsymbol{\mathcal{J}}} \overline{\lambda_j} x_j^* + \frac{1}{t_j^s} x_j^* \le \frac{u_b}{l}$ cannot be satisfied do
4:     $x_{j'}^* = 0$;
5:     Remove $j'$ from resource set $\boldsymbol{\mathcal{J}}'$;
6:     Find the suitable resource $j'$ based on Eq.15;
7: end while
8: return $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{X}}^*$.
```

III. Evaluations

Consider the scenario with $N = 100$ servers deployed in an area covered by a Base Station (BS), which is attached to a broker. As shown in Fig. 6, each server can communicate with the broker via the BS based on different kinds of communications technologies [9], [10] and clients retrieve the contents of resources (from the broker or the servers) via the BS. Meanwhile, each server hosts one IoT resource and the content size of each resource is generated from a Poisson distribution with mean $\bar{l} = 500\ Kb$. In addition, the average transmission rate of each server is obtained from a Poisson distribution with the average value of $8\ Mbps$. If the resource is cached in the broker, the server should deliver the up-to-date resource content to the broker and we assume that the average content delivery rate is the same among all the resources during a time slot, i.e., $\bar{\eta}_i = 0.01\ update/sec$. Moreover, the average transmission rate of the broker is $u^b = 350\ Mbps$. The energy coefficients of all the servers in the network are the same, i.e., $\epsilon_i = 1\ unit/bit$. The average arrival rate of resource retrieval requests for a resource is randomly selected between 0 and 0.1 $request/sec$ during a time slot, i.e., $\overline{\lambda_i} = U(0, 0.1)$. θ is set to be 0 and the duration of a time slot is 10 $mins$.

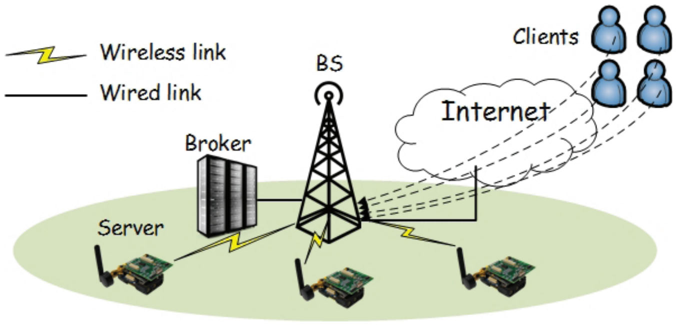


Fig. 6: The simulation setup.

We compare the performance of EASE with other two baseline strategies, i.e., Caching Preferred (CP) and Caching Non-Preferred (CNP). The idea of CP is that each popular resource j is preferred to be cached in the broker until the broker is overflowed[4]. Meanwhile, the intuition of CNP is that each popular resource j is not preferred to be cached in the

[4]The broker is overflowed if the average arrival rate of resource retrieval requests of the broker is no less than the average service rate of the broker, i.e., $\sum_{i\in\boldsymbol{\mathcal{I}}} \overline{\lambda_i} x_i \ge \frac{u_b}{l}$.

broker if its server can still handle the corresponding resource retrieval requests (i.e., $\frac{l_j}{u_j^s} > \overline{\lambda_j}$). Note that if the server cannot handle the resource retrieval requests (i.e., $\frac{l_j}{u_j^s} \leq \overline{\lambda_j}$), this resource has to be cached in the broker in CNP. Note that each server would send the resource caching request to the broker as long as its resource becomes popular in each time slot and the broker determines whether to cache these popular resources in each time slot. The total simulation period is 100 time slots and the Monte Carlo results are generated to measure the performance of the three caching strategies.

A. Overall Performance

As shown in Fig. 7, EASE and CP save a similar amount of energy from servers, which is much more than the one saved by CNP because CNP does not prefer to cache popular resources in the broker, and so servers need to transmit contents of their resources by themselves. Fig. 8 shows the number of popular resources that are cached in the broker. CNP has the fewest number of resources cached in the broker; this is why CNP saves the least energy from servers.

As shown in Fig. 7, although EASE and CP generate similar energy savings, EASE incurs much lower *average overall delay*[5] for publishing a content of a resource as compared to CP. To explain this, we analyze the *average delay among the servers*[6] and the *average delay of the broker*[7] with respect to the three strategies. As shown in Fig. 9, CP incurs much higher average delay of the broker than EASE and CNP because CP only tries to maximize the energy savings by enabling the broker to cache as many popular IoT reosurces as possible. This would result in the broker being congested, thus incurring high average delay of the broker in delivering the content of a resource. Accordingly, high average delay of the broker causes high average overall delay for CP. On the other hand, although CNP incurs the lowest average delay of the broker, it generates the highest average delay among servers as compared to EASE and CP because some servers may be over-loaded to handle many resource retrieval requests, thus suffering from high delay. As a result, high average delay among servers causes high average overall delay for CNP. Therefore, EASE optimizes suitable resources to be cached in the broker in order to save almost the same amount of energy from servers as compared to CP while ensuring the lowest average overall delay.

We further compare the performance of the three strategies by changing the average transmission rate of the broker, i.e., the value of u^b. As shown in Fig. 10, EASE and CP always incur the similar amount of energy savings; meanwhile, the amount of energy savings increases as u^b increases. This is because when u^b increases, the broker caches more popular resources by applying EASE and CP, thus potentially reducing the energy consumption of the servers. Fig. 11 demonstrates that the number of resources cached in the broker increases as u^b increases when EASE and CP are applied. However, as shown in Fig. 11, the amount of energy savings incurred by CNP does not change as u^b varies because CNP does not prefer to cache the resources in the broker even if the broker has a larger capacity to cache more popular resources.

Fig. 12 shows the average overall delay by applying the three strategies. The average overall delay of EASE decreases as u^b increases and EASE always outperforms CP and CNP. Note that the average overall delay of CP is monotonically increasing when $u^b < 440\ Mbps$ and monotonically decreasing when $u^b \geq 440\ Mbps$ because as u^b increases, CP would cache more popular resources in the broker, and so the broker needs to handle more resource retrieval requests from the clients. Thus, as shown in Fig. 13, the average delay of the broker for applying CP is increasing as u^b increases; this increases the average overall delay of CP accordingly. When $u^b \geq 440\ Mbps$, all the popular resources have already been cached in the broker, i.e., the average arrival rate of resource requests of the broker would not increase as u^b increases. Consequently, as shown in Fig. 13, the average delay of the broker for applying CP decreases, thus resulting in the decrease of the average overall delay of CP.

IV. Related works

In-network content caching has been proposed to speed up the content delivery process in Content Delivery Network (CDN) [11]–[13], i.e., contents are cached in network nodes (e.g., gateways and routers) based on some strategies (e.g., the popularity of a content) such that clients can retrieve these contents without explicitly contacting content providers. However, traditional in-network caching methods cannot be tailored for the IoT system owing to the unique features of IoT. First, most of IoT devices, are resource constrained [14], and so the main objective of in-network caching methods in IoT is to minimize the energy consumption of IoT devices rather than to minimize the average content delivery latency in CDN. Second, the contents generated by IoT devices exhibit transient feature [15]–[17], i.e., the old content quickly loses its value, thus requiring proper policies to refresh it. For instance, the content of a parking spot status (i.e., empty or occupied) may frequently change during a hour and needs to be updated over time. This feature is different from the contents cached in CDN, whose popularity remains stable over long timescale [18]. Third, the size of IoT contents may be smaller than the size of contents in CDNs, but the number of IoT contents may be larger than the number of contents in CDNs.

Owing to these unique features of the IoT system, in-network content caching strategies in the IoT network layer should be redesigned. Vural *et al.* [19] proposed to cache IoT contents in the edge routers and argued that retrieving contents from the edge routers may lose freshness (i.e., obtained data may not be up-to-date) but reduce the network traffic as compared to retrieving contents from the original IoT servers.

[5]Average overall delay is the mean of the average delay of all the resources, i.e., $average\ overall\ delay = \sum_{i\in\mathcal{I}} \left(t^b x_i + t_i^s (1-x_i)\right) / |\mathcal{I}|$.

[6]Average delay among the servers indicates the mean of the average delay of all the resources, whose contents are delivered by their servers, i.e., $average\ delay\ among\ servers = \sum_{i\in\mathcal{I}} t_i^s (1-x_i) / \sum_{i\in\mathcal{I}} (1-x_i)$.

[7]The average delay of the broker is the mean of the average delay of all the resources, whose contents are delivered by the broker, i.e., $average\ delay\ of\ the\ broker = \sum_{i\in\mathcal{I}} t^b x_i / \sum_{i\in\mathcal{I}} x_i$.

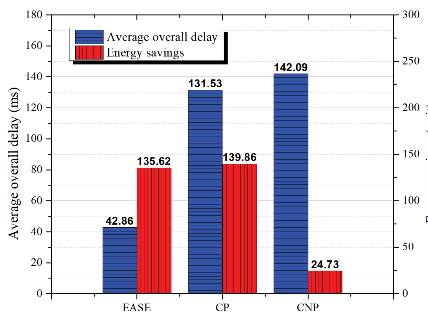


Fig. 7: Average overall delay and the amount of energy savings.

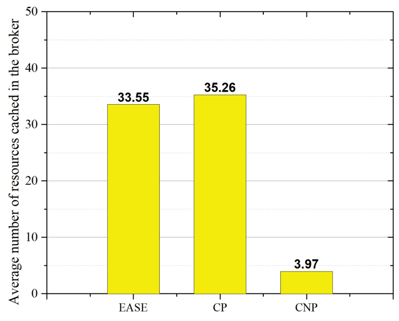


Fig. 8: Average number of resources cached in the broker.

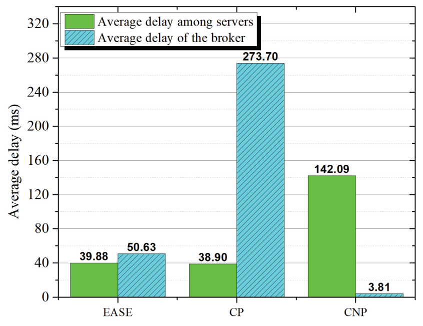


Fig. 9: Average delay among servers and Average delay of the broker.

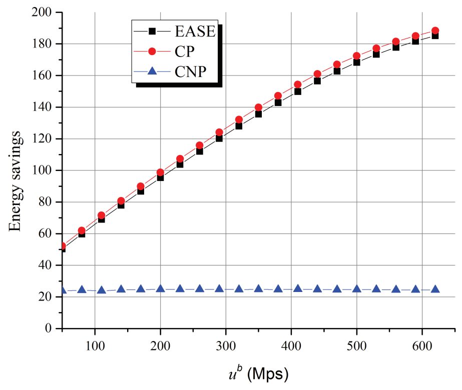


Fig. 10: Amount of energy savings.

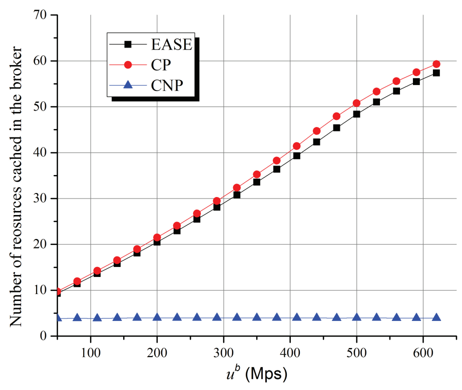


Fig. 11: Number of cached resources.

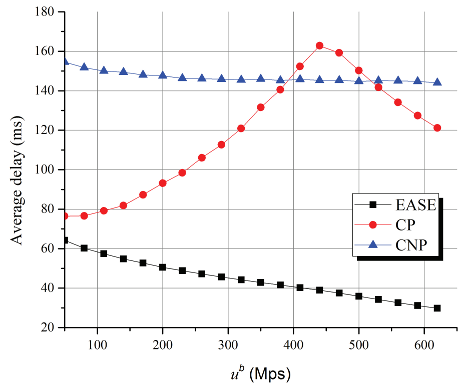


Fig. 12: Average overall delay.

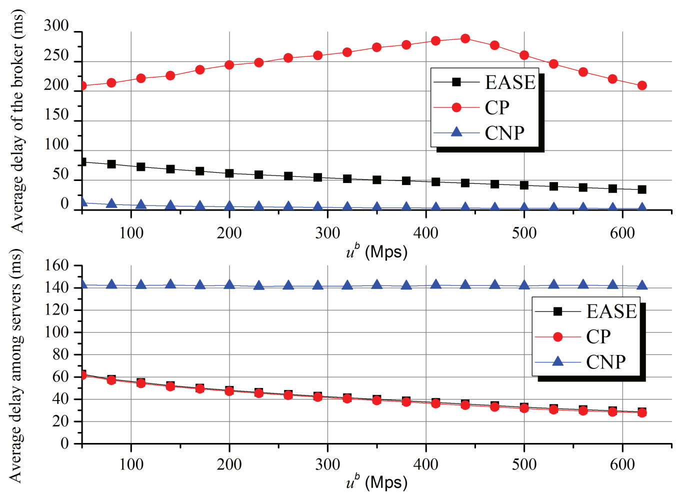


Fig. 13: Average delay among servers and average delay of the broker.

Thus, they dynamically modified the edge routers' content caching probabilities in order to optimize the tradeoff between content freshness and network traffic. Similarly, Hail *et al.* [20] proposed a network layer IoT content caching strategy in the multi-hop wireless network scenario, in which IoT devices are equipped to cache the forwarding contents. They designed a novel distributed probabilistic caching strategy, which is based on the freshness of the content as well as the energy level and the storage capability of the device, to improve the energy efficiency of the IoT devices and reduce the content delivery delay. Niyato *et al.* [21] advocated that contents generated by IoT devices should be cached in local wireless access points, from which contents should always be retrieved by clients. They designed an optimal caching update period for each IoT device to maximize the hit rate in terms of the probability of clients in successfully obtaining the corresponding contents.

Although in-network caching in the IoT network layer can potentially reduce the delay of the clients in retrieving their requested IoT contents and the energy consumption of IoT devices in delivering their IoT contents, it suffers from two drawbacks, i.e., the cache inconsistency problem [22], [23] and the security problem [24]. The cache inconsistency problem refers to the inconsistency between the content cached in network nodes and that generated by the IoT device, and it would result in the content received by the clients not reflecting the current state of the corresponding IoT device. Meanwhile, the security problem implies that network nodes are unable to conduct access control to prevent unauthorized clients from retrieving the cached IoT contents. The cache inconsistency and security problems are attributed to the local caching decisions made by network nodes (i.e., an IoT device is unaware of its content having been cached by the network nodes, and is thus unable to update the caches and add corresponding access control policies in these network nodes).

In order to solve the cache inconsistency and security problems of in-network caching in the IoT network layer, we propose to implement IoT resource caching in the application layer, which is different from IoT content caching in the IoT network layer. First, IoT resources do not have the freshness feature. Clients can retrieve up-to-date contents by accessing the corresponding IoT resources. Second, IoT resources are cached in a broker (i.e., a middleware) rather than network nodes, and the broker is discoverable by clients and IoT devices. Third, IoT devices are aware of where their cached IoT resources are located, and thus IoT devices can update the contents and add access control policies of the IoT resources. Therefore, IoT resource caching in the application layer can basically resolve the cache inconsistency and security problems. To the best of our knowledge, we are the first to propose caching resources in the IoT application layer.

V. Conclusion

In this paper, we have proposed to cache the IoT resources in the application layer by applying CoAP Pub/Sub. We have formulated the problem of dynamic resource caching in the broker to maximize the total energy savings from servers, while guaranteeing the minimum average delay (for publishing the content of the resource to the clients) for each resource. We have designed the EASE algorithm to solve the problem. The performance of EASE has been demonstrated via simulations.

References

[1] Q. Jing, *et al.*, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.

[2] A. Al-Fuqaha, *et al.*, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.,*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] The Constrained Application Protocol (CoAP). [Online]. Available: https://tools.ietf.org/html/rfc7252.

[4] Publish-Subscribe Broker for CoAP. [Online]. Available: https://tools.ietf.org/html/draft-koster-core-coap-pubsub-05.

[5] X. Sun and N. Ansari, "PRIMAL: PRofIt Maximization Avatar pLacement for mobile edge computing," *2016 IEEE Intl. Conf. Commun. (ICC)*, Kuala Lumpur, 2016, pp. 1–6.

[6] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, July 2017.

[7] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[8] B.T. Polyak and A.B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control and Opt.*, vol. 30, no.4, pp. 838–855, 1992

[9] N. Ansari and X. Sun, "Mobile edge computing empowers Internet of Things," *IEICE Trans. on Commun.*, doi: 10.1587/transcom.2017NRI0001, early access.

[10] X. Sun and N. Ansari, "Cloudlet Networks: Empowering Mobile Networks with Computing Capabilities," *IEEE COMSOC MMTC Commun.-Frontiers*, vol. 12, no. 4, pp. 6-11, July 2017.

[11] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Comput. Netw.*, vol. 57, no. 16, pp. 3128–3141, 2013.

[12] X. Yuan, *et al.*, "Enabling secure and efficient video delivery through encrypted in-network caching," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2077–2090, Aug. 2016.

[13] Y. Li, *et al.*, "How Much to Coordinate? Optimizing In-Network Caching in Content-Centric Networks," *IEEE Trans. Netw. and Serv Manag.*, vol. 12, no. 3, pp. 420–434, Sept. 2015.

[14] A. Sehgal, *et al.*, "Management of resource constrained devices in the internet of things," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 144-149, December 2012.

[15] S. Vural, *et al.*, "Caching transient data in Internet content routers," *IEEE/ACM Trans. on Netw.*, vol. 25, no. 2, pp. 1048-1061, Apr. 2017.

[16] M.A. Hail, *et al.*, "On the performance of caching and forwarding in information-centric networking for the IoT," *Intl. Conf. Wired/Wireless Internet Commun.*, Malaga, Spain, May 25–27, 2015, pp. 313-326.

[17] M. Abu-Elkheir, M. Hayajneh, and N. A. Ali, "Data management for the internet of things: Design primitives and solution," *Sensors*, vol. 13, no. 11, pp. 15582–15612, 2013.

[18] N. Golrezaei, *et al.*, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, April 2013.

[19] S. Vural, *et al.*, "In-network caching of internet-of-things data," *2014 IEEE Intl. Conf. Commun. (ICC)*, Sydney, NSW, 2014, pp. 3185–3190.

[20] M.A. Hail, *et al.*, "Caching in named data networking for the wireless Internet of Things," *2015 Intl. Conf. Recent Advances in Internet of Things (RIoT)*, Singapore, 2015, pp. 1–6.

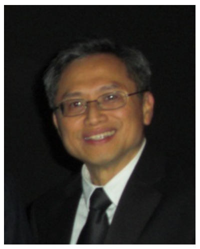
[21] D. Niyato, *et al.*, "A novel caching mechanism for Internet of Things (IoT) sensing service with energy harvesting," *2016 IEEE Intl. Conf. on Commun. (ICC)*, Kuala Lumpur, 2016, pp. 1–6.

[22] A. Lindgren, *et al.*, "Design choices for the IoT in information-centric networks," *2016 13th IEEE Ann. Consumer Commun. & Netw. Conf. (CCNC)*, Las Vegas, NV, 2016, pp. 882–888.

[23] A. Rao, O. Schelen, and A. Lindgren, "Performance implications for IoT over information centric networks," *Proc. Eleventh ACM Wksp. on Challenged Netw.*, New York City, NY, Oct. 03–07, 2016, pp. 57–62.

[24] R. Li, *et al.*, "A Verifiable and Flexible Data Sharing mechanism for Information-Centric IoT," *2017 IEEE Intl. Conf. Commun. (ICC)*, Paris, France, 2017, pp. 1-7.

Xiang Sun [S'13] received a B.E. degree in electronic and information engineering and an M.E. degree in technology of computer applications from Hebei University of Engineering, Hebei, China. He is currently working towards the Ph.D. degree in electrical engineering at NJIT, Newark, New Jersey. His research interests include mobile edge computing, big data networking, green computing and communications, content/resource caching in Internet of Things, and Drone-aided mobile access networks.



Nirwan Ansari [S'78, M'83 ,SM'94, F'09] is Distinguished Professor of Electrical and Computer Engineering at NJIT. He has also been a visiting (chair) professor at several universities.

He recently authored Green Mobile Networks: A Networking Perspective (Wiley-IEEE, 2017) with T. Han, and co-authored two other books. He has also (co-)authored more than 500 technical publications, over 200 in widely cited journals/magazines. He has guest-edited a number of special issues covering various emerging topics in communications and networking. He has served on the editorial/advisory board of over ten journals. His current research focuses on green communications and networking, cloud computing, and various aspects of broadband networks.

He was elected to serve in the IEEE ComSoc Board of Governors as a member-at-large, has chaired ComSoc technical committees, and has been actively organizing numerous IEEE International Conferences/Symposia/Workshops. He has frequently delivered keynote addresses, distinguished lectures, tutorials, and invited talks. Some of his recognitions include several Excellence in Teaching Awards, some best paper awards, the NCE Excellence in Research Award, the IEEE TCGCC Distinguished Technical Achievement Recognition Award, the ComSoc AHSN TC Technical Recognition Award, the NJ Inventors Hall of Fame Inventor of the Year Award, the Thomas Alva Edison Patent Award, Purdue University Outstanding Electrical and Computer Engineer Award, and designation as a COMSOC Distinguished Lecturer. He has also been granted 35 U.S. patents.

He received a Ph.D. from Purdue University in 1988, an MSEE from the University of Michigan in 1983, and a BSEE (summa cum laude with a perfect GPA) from NJIT in 1982.