A Time-Adaptive Heuristic for Cognitive Cloud Offloading in Multi-RAT Enabled Wireless Devices

S. Eman Mahmoodi, *Student Member, IEEE,* and K. P. Subbalakshmi, *Senior Member, IEEE*

***Abstract*—We introduce the concept of cognitive cloud offloading where all viable wireless interfaces of a multiple radio enabled device are used for computation offloading. We propose a time and wireless adaptive heuristic for offloading computationally intensive applications to a remote cloud with goals of reducing the energy consumption on the mobile device, execution time of the application, and efficient use of the multiple radio interfaces available at the device. The proposed algorithms simultaneously determine: (*i*) execution place of each application component (mobile/cloud); (*ii*) amount of the associated data to be sent via each available interface of the multi-RAT device; and (*iii*) scheduling order of the application components. We define a net utility function that trades off mobile device resources (battery, CPU, and memory) with realtime communication costs such as latency and communication energy, subject to constraints that ensure queue stability of radio interfaces. Simulations using real data from an HTC smartphone running multi-component applications with Amazon EC2 as the cloud, and two radios, LTE and WiFi, show that cognitive cloud offloading provides higher net utility in comparison to the best-interface protocol. Scalability of the proposed heuristic is further analyzed using various levels for component dependency graphs and energy-delay trade-off factors.**

***Index Terms*—Cognitive computation offloading, joint scheduling–offloading, mobile cloud computing, multi-RAT offloading, spectrum-aware mobile computing.**

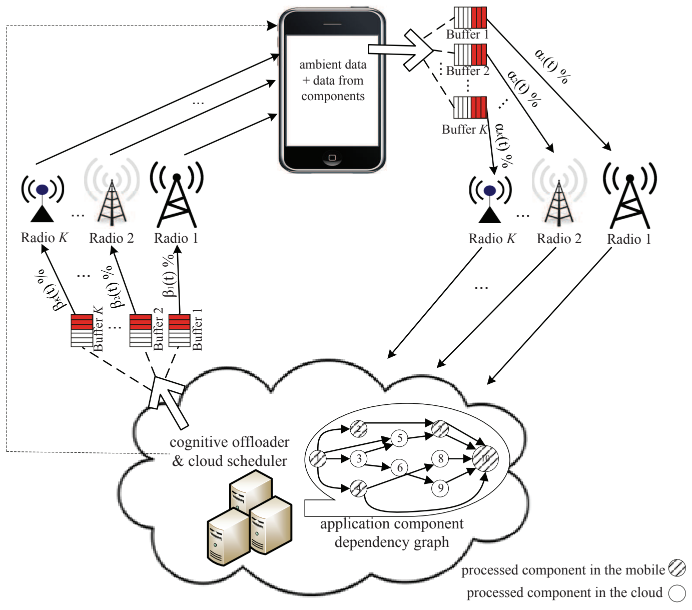

ambient data + data from components
Buffer 1
Buffer 2
Buffer K
$\alpha_1(t)$ %
$\alpha_2(t)$ %
$\alpha_K(t)$ %
Radio K
Radio 2
Radio 1
$\beta_K(t)$ %
$\beta_2(t)$ %
$\beta_1(t)$ %
cognitive offloader & cloud scheduler
application component dependency graph
processed component in the mobile
processed component in the cloud

Fig. 1: Cognitive offloading for multi-RAT enabled wireless devices.

I. Introduction

With the advent of 5G wireless communications, the expectations from the mobile devices have increased. Computationally intensive mobile applications such as augmented reality, speech recognition, 3D interactive gaming, natural language translation, healthcare sensing and analysis are beginning to be supported on the mobile device. This multi-tasking puts significant *effective* constraints on resources like memory and battery power available to each of the applications.

Although offloading computation [2] to a remote resource-rich cloud has existed as a solution to this problem in various forms [3]–[5], offloading will also place a further burden on the already limited *spectrum resources* since data will have to be transferred between the mobile devices and the cloud in order to successfully deliver the application on the mobile device.

Meanwhile multiple radio access technology (multi-RAT) enabled mobile devices are poised to become a mainstay of the future of wireless networking [6] using several approaches like: antenna arrays that can concurrently transmit [7]; multi-path TCP (MPTCP) protocols to simultaneously access multiple networks [8]; and carrier aggregation [9]. *We propose to use this newly emerging technology in our cloud offloading solution. We define the concept of cognitive cloud offloading where the computational offloader not only decides which components of a complex application should be offloaded and which should run locally, but also which radio interfaces must be used in the associated data transfers and what percentage of the data should be communicated through each interface. Cognitive use of all the wireless interfaces at the same time leads to a higher throughput of the network* (see Fig. 1). The term cloud offloading can mean data *flow* offloading [10], [11] or offloading computationally intense tasks to the cloud [12]. In this paper, we refer to the latter. Our first work in this area appeared in [13], where this problem was considered when all the wireless parameters were collected and the optimal one shot solution was derived. In this paper, we move to a more realistic extension of the problem, where: (1) we consider more general dependencies between the components of the application (see Section III for more on component dependency graphs (CDGs)) and (2) propose a time-adaptive algorithm that varies with the *dynamic changes* in the wireless network over time. In this paper, *we propose a heuristic online (time-adaptive) scheme to optimally schedule the application's components for offloading, while simultaneously optimizing the percentage of data to be sent by the mobile and the cloud via each wireless interface*. We develop a comprehensive

S. E. Mahmoodi and K. P. Subbalakshmi are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, 07030 USA e-mails: {smahmood, ksubbala}@stevens.edu.

model for the utility function that trades-off resources (such as energy, memory, and CPU consumption by the mobile device) with the cost of communication required for offloading (such as energy consumed by offloading and the data queue length at the multiple radio interfaces). Our solution can be implemented in two ways: (*i*) a *two stage algorithm* where some of the components are eliminated as unsuitable for offloading at the outset, maximizing the instantaneous utility values at time t_0 (offline stage). The actual components to be offloaded will be selected online using the appropriate scheduling constraints in the second stage; and (*ii*) a *single stage algorithm* where all the components are considered for offloading and the offload decisions are made online, based on some scheduling constraints. The offloading strategies for transmission at the mobile and cloud ends use past wireless interface data, queue status and the current data flow to update the current queue status. We compare the performance of the proposed algorithm to different approaches such as (*i*) no offloading; (*ii*) complete offloading (all components remotely executed); (*iii*) the offline dynamic offloading algorithm proposed in [14] extended to applications with *sequential* dependency graphs; and (*iv*) the approach where offloading takes place only via the best link at that time. Note that since the algorithm in [14] is an *offline* strategy specifically for applications with *sequential* CDGs, we use a special case of the algorithm presented here for comparisons with the work in [14].

The rest of this paper is organized as follows. In Section II, we discuss the related works to this paper. Then, we model the multi-RAT network, express the CDG of sophisticated mobile applications, and formulate the net utility function of the system in Section III. The proposed heuristic strategy for joint cognitive offloading and scheduling is addressed step-by-step in Section IV. In Section V, we present versions of the proposed heuristics as well as the experiments and simulations to evaluate the performance of the proposed strategy. Finally, Section VI presents the conclusion and future work of the paper.

II. Related Work

We now provide an overview of recent cloud offloading mechanisms classified based on the granularity and extent of offloading, application partitioning, offline/online scheduling, and use (or not) of multi-RAT technologies.

Classification based on the granularity and extent of offloading: Computation offloading for mobile networks can be categorized into three groups: (*i*) all or nothing offloading, where the application is either completely offloaded to a remote cloud or completely executed locally [15]; (*ii*) wholesale offloading– where the entire application is eventually offloaded [16], [17]; and (*iii*) those that partition the application into smaller units and make piecewise decisions to either execute the unit locally or to offload it to a remote cloud [13], [18]–[22]. This last category offers the maximum degrees of freedom for this problem [5], [23] and hence, we use this approach in the paper.

Within the partial offloading strategies, some schemes have proposed coarse level partitioning of the applications where the code is pre-partitioned into components [13], [14], [22], [24]. A more fine-grained offloading can be achieved by using method-level partitioning as in MAUI [18]. ThinkAir [19] also provides method-level partitioning but focuses more on scalability issues and parallel execution of offloaded tasks. An Android specific services-based mobile cloud computing middleware called Mobile Augmentation Cloud Services (MACS) [20] allows for seamless offloading of the application to the cloud. The decision for partitioning is cast as an optimization problem using cloud and device parameters, such as CPU load, available memory, remaining device battery power and available spectrum bandwidth. COSMOS [21] is also a fine grained platform where partial computation offloading of sequential tasks is proposed as a service.

Offline/Online scheduling in wireless cloud offloading: Another way to classify the existing offloading techniques is based on whether the decisions to offload components is done at the beginning (one-shot, also known as offline strategy) or whether these decisions are made on the fly (online strategies) [25].

Offline offload strategies that minimize the energy consumed by the mobile device with constraints on the overall application deadline have been considered in [26], while individual deadline constraints for application tasks is monitored in [27]. A partial offline offloading policy for the special case of applications with serial dependency graphs (see Section III for detailed description of CDGs) is proposed in [28] and [29]. A partial offloading strategy using a predictive algorithm for wireless connectivity is used in [30], where a risk control strategy is applied to increase reliability of the prediction. Another work, based on genetic algorithms was proposed in [31]; however, this strategy does not consider multi-RAT enabled devices or the scheduling order of components based on the CDGs.

Online (time-adaptive) cloud offloading for mobile devices requires awareness of instantaneous changes in the rates, delay values, and communication power for all of the radio interfaces. A partial computation offloading for frame-based real-time tasks with response time guarantees from the cloud servers is studied in [32] where the server estimates the response time for remote execution of each task based on total bandwidth server model [33], and the tasks are scheduled for offloading with “earliest deadline first” algorithm. An “everything on the cloud” offloading strategy based on energy and delay trade-off is proposed in eTime [16]. Although, this work does assume a multi-RAT device only the *best single* wireless interface is used for offloading.

Wireless cloud offloading using single/multiple radio interface(s): The third type of classification is based on whether the offloading algorithms are developed for single- or multi-RAT devices, and if multi-RAT devices are used, whether the wireless interfaces are used in a hybrid mode or an On/Off mode.

1) Wireless cloud offloading for single-RAT devices: A multi-*channel* partial offline offloading solution for single-*RAT* enabled mobile devices was proposed in [22]. Note that the multi-RAT scenario is significantly different from the multi-channel single-RAT, in that the parameters of the different *networks* (e.g. WiFi and LTE) supported by multi-RAT devices vary widely in comparison to the parameters of different *channels* of the same radio interface [34]. The extension of [22] (with predetermined call graphs) to joint allocation of transmit power and OFDM constellation size in single and multi-channel cases in single-RAT enabled devices is studied in [35]. Other work on single-RAT wireless offloading include [12], [15], [18]–[21], [26]–[32], [36]. Recently we proposed the concept of joint scheduling–offloading in [36], where the scheduling order of execution for the components as well as where each component must be executed, is jointly determined. This is in contrast to assuming a compiler pre-determined scheduling order and allows the algorithm to pick an optimal order of execution, appropriate for the wireless network conditions. This algorithm was designed for single-RAT mobile devices and is an offline solution.

2) Wireless cloud offloading for multi-RAT devices: Cloud offloading strategies for multi-RAT devices [13], [14], [16] have recently begun to gain interest in the community because of the advances in cognitive radio networking [37] and heterogeneous networks (HetNets) as well as the trend in 4G to 5G evolution. However, most of these works [14], [16] only use one of the wireless interfaces (the one with the best characteristics) for all offloading related data transfers. Hence, these algorithms basically work with an *On-Off* model for the wireless interfaces. We proposed the first *cognitive offloading* strategy for multi-RAT devices, where all viable wireless interfaces are simultaneously used to offload as well as the optimal percentage of data over each wireless interface [13]. While our prior work [13] was a one-shot (offline) offloading strategy, this paper discusses an online (time-adaptive) strategy for cognitive offloading. Moreover, in [13] we optimized over all available interfaces at the mobile transmitter end only, and not at the cloud transmitter end. In this work we present an online (time-adaptive) cognitive mobile offloading strategy for *both transmission and reception* of relevant data. Finally, in [13] we assumed that the applications had either serial CDGs or that we use a pre-determined scheduling order for the components of the application, whereas here, we propose a joint scheduling–offloading strategy for applications with arbitrary CDGs.

To the best of our knowledge, *this is the first work on online (time-adaptive) joint scheduling and cognitive offloading for multi-RAT devices for applications with arbitrary dependency graphs*. The heuristic scheme in this paper fundamentally differs from the recent works in the following aspects: (*i*) this is a true cognitive offloading strategy for multi-RAT devices that uses all available wireless interfaces (to the extent possible) for offloading computation; (*ii*) it is a time-adaptive online offloading strategy while dynamics of wireless networks are taken into account; (*iii*) this is for applications with *general component dependencies* and (*iv*) hence it is the first joint scheduling-cognitive offloading scheme for multi-RAT devices.

III. Network and Application Model

Consider a mobile device with K radio interfaces in a wireless network, running an N-component mobile application

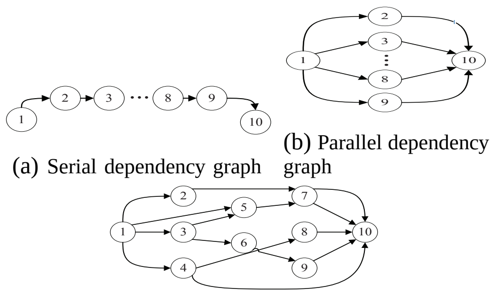
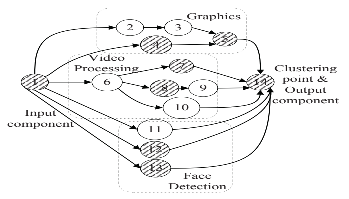

(a) Serial dependency graph (b) Parallel dependency graph (c) General dependency graph

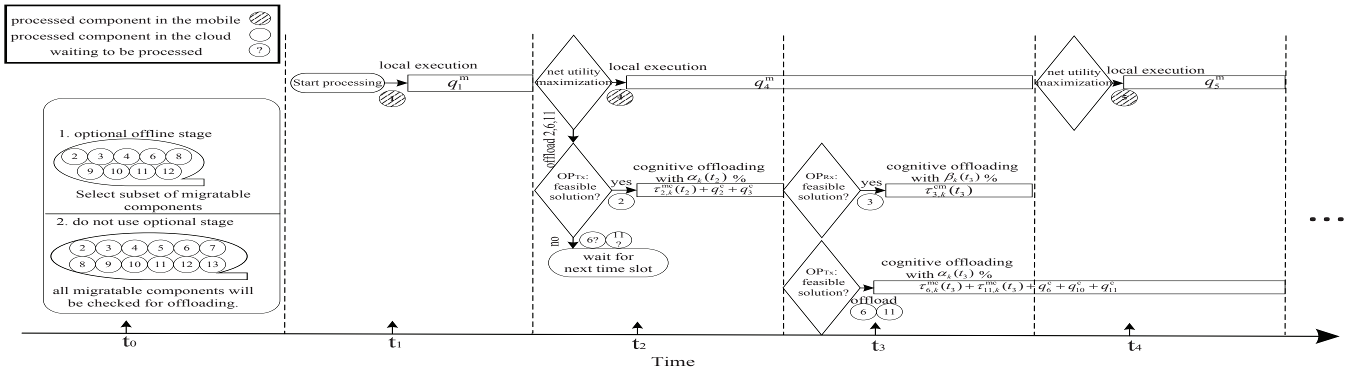
Fig. 2: Examples of component dependency graphs (CDGs) for mobile applications ($N = 10$).

(see Fig. 1). The goal of the algorithm is to find a *time-adaptive* scheduling–offloading policy for all components as well as the optimal wireless resource allocation between the multi-RAT interfaces for data transfers of both the mobile to cloud and the cloud to mobile. In the model in Fig. 1, at time slot t $\forall t$, $\alpha_k(t)$% of the required data for offloading is sent by the mobile device through radio interface k $\forall k$. Similarly, $\beta_k(t)$% of the data is sent by the cloud end using radio interface k $\forall k$. Both $\alpha_k(t)$ and $\beta_k(t)$ are computed to achieve optimum net utility.

Component Dependency Graph (*CDG*): As mentioned earlier, mobile applications can be partitioned into components [20], [38], [39]. Component i is said to be dependent on component j, if data must be sent from j (after j completes execution) to i, in order for i to complete its execution. This dependency is usually depicted as a component dependency graph (CDG). CDGs can be: (*i*) serial (where one component depends only on the output of one other component); (*ii*) parallel (where all components depend on only the first component and the last component depends on the rest); and (*iii*) general (which can be any combination of serial and parallel). Fig. 2 shows examples of each type for $N = 10$. Here the nodes represent components and directed links show dependency between the components. A real example of general CDG for a video navigation application with $N = 14$ components is shown in Fig. 3a. In this video navigation application, graphics library tools are used from the OpenGL mobile Android applications [40], face detection is used from [41], and all of the video processing features are obtained from [42]. Our time-adaptive cognitive cloud offloading strategy will schedule each component to process either in the cloud or in the mobile, while keeping these dependencies in mind, along with other constraints. *The decision to offload or locally execute a component will be made adaptive to current wireless conditions*. Since most applications are user initiated, the first (potentially involving some input from the human user) and last (potentially involving some displayed output) components are typically scheduled on the mobile device. Fig. 3b shows an example of the stages of processing for the application with the CDG shown in Fig. 3a at the first 5 time slots. More details of the algorithm are described in Section IV. We can see that some components (e.g. components 2, 6, and 11 in the cloud and 4 in the mobile for this example) can be scheduled for



(a) CDG of the application.


processed component in the mobile
processed component in the cloud
waiting to be processed
1. optional offline stage
Select subset of migratable components
2. do not use optional stage
all migratable components will be checked for offloading.
Start processing
local execution
net utility maximization
offload 2,6,11
OPTx: feasible solution?
yes
no
wait for next time slot
cognitive offloading
OPRx: feasible solution?
offload
Time

(b) Scheduling strategy.

Fig. 3: Time adaptive scheduling–offloading for an example of 14 component mobile application.

parallel execution.

A smart cognitive cloud offloading algorithm will trade-off the benefits of wireless offloading, namely, energy and time savings (when components can be parallely scheduled in the cloud and the mobile) with the costs of offloading, namely, the energy and delay costs involved in the associated data transfer, while simultaneously deciding on the optimal percentage of the data to send from the mobile and the cloud via each of the available wireless network interfaces. We assume that the energy consumption and the time required to transfer data between components that are executed in the same entity (whether cloud or mobile) are negligible in comparison to when the data must be transferred between entities. Also, we assume that the cloud and the mobile clocks are synchronized [43]. The time-adaptive scheduling–offloading heuristic is managed in the cloud, and the feedback of decisions on offloading the components will be sent to the mobile device (Fig. 1). The mobile device informs the cloud of the corresponding parameters via proper control signaling before cognitive cloud offloading. Please note that the delay caused due to sending data related to the delay, rate, queue size, and communication power is negligible in comparison with the computation offloading costs which may require transferring MegaBytes of data. In the online stage at each time slot, two sub-strategies are developed for cloud offloading: (*i*) mobile to cloud transmission strategy that trades-off the energy consumption by the mobile for transmission, the delay for transferring the required data from mobile to the cloud, and queue stability of the mobile Tx buffer for all the radio interfaces; and (*ii*) cloud to mobile transmission strategy that trades-off the energy consumption by the mobile for reception, the delay for transferring the required data from cloud to the mobile, and queue stability of the cloud Tx buffer for all the radio interfaces, which reflects the connectivity with the mobile receiver.

Net Utility Function: To determine the best strategy for joint scheduling–offloading, we need to define an appropriate net utility function. The notations used for these terms and the other parameters in this paper are defined in Table I. To determine the scheduling–offloading strategy for component i, two decision variables are defined for a time slot t as follows: $I_i(t) = 1$, if component i starts offloading at t, and otherwise it gets 0; $X_i(t) = 1$, if component i starts executing locally at t, and otherwise it gets 0. The net utility is calculated as a weighted sum of the energy, memory, and CPU cycles saved for the mobile device minus the inter-component communication cost arising from executing some components locally and some remotely. This can be written as:

$$\begin{aligned} U(t) =& w_{\text{saved}} E_{\text{saved}}(t) + w_{\text{memory}} M_{\text{saved}}(t) + \\ & w_{\text{CPU}} CPU_{\text{saved}}(t) - w_{\text{com}} C_{\text{com}}(t). \end{aligned} \tag{1}$$

The weights for the individual costs and benefits are chosen such that $w_{\text{saved}} = 1 - w_{\text{com}}$, and $w_{\text{CPU}} = 1 - w_{\text{memory}}$. These weights indicate the relative importance of the specific parameters like memory saved, CPU saved, or the communication costs and battery power. If w_{saved} is higher then it is more desirable to save battery power at the mobile device, and if w_{com} is higher it means that more weight is placed on minimizing the communication cost for cloud offloading. By setting weights, we pre-bias the solution more towards either of these solutions. Sometimes issues including monetary cost of using cloud services could be significant. Therefore, we might want to favor offloading to the cloud server slightly less. By adjusting the weights between "stay in the mobile" or "offload to the cloud", we add an extra control knob that lets us weight one or the other more.

TABLE I: Parameter Definitions.

Parameters	Definitions
$A_k^{\text{mc}}(t)(A_k^{\text{cm}}(t))$	data rate transmitted from the mobile (cloud) to the cloud (mobile) through radio interface k at time slot t.
$B_i^{\text{mc}}(t)$ $(B_i^{\text{cm}}(t))$	arrival data rate at the mobile (cloud), including the ambient traffic as well as the data generated by offloaded component i (arrival data in time slot t).
$c_i(t)$	indicator function that takes on a value of 1, if the component i has started execution in the cloud at any time between 1 and t.
$\hat{C}_{\text{com}}(t_0)$	lower bound approximation of the communication cost for component dependencies in the offline stage.
code_i	code size to launch component i.
$E_{\text{com}}^{\text{Tx}}(t)$ $(E_{\text{com}}^{\text{Rx}}(t))$	Energy consumed for the mobile transmission (reception) due to cloud offloading.
$I_i(t)$	offloading indicator: 1 if the mobile starts to offload component i at time slot t.
K	number of wireless radio interfaces.
$l_{ji}(t)$	the time slots to process the preceding component j, and transfer the output data from component j to i by t.
M_i	RAM memory consumed by the mobile device to launch component i.
$m_i(t)$	indicator function that takes on a value of 1, if the component i has started local execution at any time slot between 1 and t.
N	number of components in the application.
P_i^{m}	power consumed by the mobile device when it is actively processing component i.
$P_k^{\text{Tx}}(t)$ $(P_k^{\text{Rx}}(t))$	transmit (received) power consumed by the mobile device through radio interface k at time slot t.
$Q_k(t)$ $(S_k(t))$	the transmission queue of data from the mobile (cloud) side for wireless interface k at time slot t.
q_i^{m} (q_i^{c})	number of time slots to process component i in the mobile (cloud).
T	number of time slots to complete processing the application.
$T_{\text{th}}^{\text{Tx}}$ $(T_{\text{th}}^{\text{Rx}})$	threshold number of time slots for transmission from mobile (cloud) to cloud (mobile).
$U(t)$	net utility function at time t.
V_{mc} (V_{cm})	control parameter in mobile (cloud) transmission for Lyapunov optimization.
w_x	weight factor of function x.
x	span of each time slot.
$X_i(t)$	local execution indicator: 1 if the mobile starts to execute component i locally at time slot t.
$z_{ij}(t)$	indicator for communication requirement: 1 if component i is executed in the mobile and j is offloaded to the cloud by time slot t.
$\alpha_k(t)$	percentage of allocated uplink (mobile to cloud) rate using radio interface k for communication at t.
$\beta_k(t)$	percentage of allocated downlink (cloud to mobile) rate using radio interface k for communication at t.
ε	mapping factor to relate code size and the CPU instructions [44].
γ	weight factor (to adjust the wait time for offloading).
μ_{ij}	dependency indicator: 1 if component i must be processed before j and 0 otherwise.
$\Omega_{\text{mc}}(t)$ $(\Omega_{\text{cm}}(t))$	the objective function for mobile (cloud) transmission strategy at time t.
$\tau_{i,k}^{\text{mc}}(t)$ $(\tau_{i,k}^{\text{cm}}(t))$	delay (in number of time slots) to transmit the output data from component i in the mobile (cloud) to the cloud (mobile) at interface k starting by t.

At any given time t, the total energy saved by executing the components in the cloud can be computed as the energy cost for running it locally ($P_i^{\text{m}} q_i^{\text{m}}$), which is given by: $E_{\text{saved}}(t) = \sum_{i=1}^{N} c_i(t) P_i^{\text{m}} q_i^{\text{m}}$, where $c_i(t) = \sum_{s=1}^{t} I_i(s)$ and $s = 1$ corresponds to the first time slot, when component i begins to execute. Likewise, $m_i(t) = \sum_{s=1}^{t} X_i(s)$ where s runs from the first time slot to the time slot corresponding to the current time t.

The memory saved in the mobile device by offloading the components to the cloud can be expressed as: $M_{\text{saved}}(t) = \sum_{i=1}^{N} c_i(t) M_i$, where M_i is the memory consumed by the mobile device to launch component i. The objective function for CPU cycles saved is given by: $CPU_{\text{saved}}(t) = \sum_{i=1}^{N} c_i(t)(\varepsilon(\text{code}_i))$, where code_i is the size of the code for instructions that is used for executing component i and ε is the mapping between code size and the CPU instructions. The communication cost at time slot t ($C_{\text{com}}(t)$) will be discussed in the next section.

IV. Proposed Cognitive Offloading & Scheduling Heuristic

In this section, we propose a heuristic to find an online cognitive scheduling–offloading strategy for the computations of mobile applications. The objective of the strategy is to specify the components that are selected for computation offloading, the time that each component should be scheduled for execution either locally or remotely, and the radio interface allocation for offloading at each time slot for both the mobile and cloud data transmission. Decision variables include: offloading indicator ($I_i(t)$)/ local execution indicator ($X_i(t)$) for component i at time slot t; and the percentage of allocated uplink ($\alpha_k(t)$) and downlink ($\beta_k(t)$) rates using radio interface k at time slot t. The complete algorithm is depicted in Algorithm 1. A detailed description of the algorithm follows:

A. Optional Offline Stage

As mentioned in Section I, the offloading problem can be formulated as a two-stage or single-stage algorithm. This section discusses the first stage of the two-stage algorithm. In this stage, we eliminate some of the components as unsuitable for offloading at the outset, maximizing the instantaneous utility values at time t_0 (e.g. components 5 and 7 in Fig. 3b). Thus, this stage provides a suboptimal solution for the heuristic algorithm. This stage can be omitted, and all the components can be considered for potential offloading in case a single-stage version of the algorithm is preferred. Note that the single-stage algorithm adds more time complexity to the online stage of the algorithm as compared to the two-stage algorithm, but is closer to the optimal solution.

In the offline stage, we identify the components that contribute the most to an increase in the net utility if scheduled in the mobile device and then eliminate them from being considered for offloading. We first obtain an approximate value for the optimal solution based on the information available at time t_0. To do this, we maximize the instantaneous net utility given by Eqn (1) using an approximation for the energy cost of offloading corresponding to time t_0 ($C_{\text{com}}(t_0)$) assuming that the interface with the lowest communication power levels is used for data transfer. Mathematically,

$$\hat{C}_{\mathrm{com}}(t_0) = w_{\mathrm{com}}\{\sum_{i=1}^{N}\sum_{j=1}^{N}\mu_{ij}m_i(t_0)c_j(t_0)\min_k(P_k^{\mathrm{Tx}}(t_0)\tau_{\mathrm{th}}^{\mathrm{mc}}) + \mu_{ij}c_i(t_0)m_j(t_0)\min_k(P_k^{\mathrm{Rx}}(t_0)\tau_{\mathrm{th}}^{\mathrm{cm}})\}, \tag{2}$$

where μ_{ij} represents the dependency indicator (1 if component i must be processed before j, and 0 otherwise), and $\tau_{\mathrm{th}}^{\mathrm{mc}}$ and $\tau_{\mathrm{th}}^{\mathrm{cm}}$ are the maximum transmission durations (in number of time slots) at the mobile and cloud ends for each component, respectively. To obtain the approximation given by Eqn (2), we assume that the Tx and Rx power levels ($P_k^{\mathrm{Tx}}(t_0)$ and $P_k^{\mathrm{Rx}}(t_0)$) are fixed when computing these values for the offline stage. By selecting the wireless interface with the lowest Tx and Rx energy levels, we obtain the minimum energy consumed for communication over the wireless interfaces ($\forall k = 1, 2, \ldots, K$) in Eqn (2) with the initial information in the offline stage (i.e., in the transmission, we have: $\min_k(P_k^{\mathrm{Tx}}(t_0)\tau_{\mathrm{th}}^{\mathrm{mc}})$). The optimization problem in the offline stage can be written as:

$$\mathrm{OP}_{\mathrm{off}} : \max_{\mathbf{c}} w_{\mathrm{saved}}E_{\mathrm{saved}}(t_0) + w_{\mathrm{memory}}M_{\mathrm{saved}}(t_0) + w_{\mathrm{CPU}}CPU_{\mathrm{saved}}(t_0) - \hat{C}_{\mathrm{com}}(t_0), \tag{3}$$

s.t.

$$\sum_{i=1}^{N} m_i(t_0)q_i^{\mathrm{m}} + \sum_{i=1}^{N} c_i(t_0)q_i^{\mathrm{c}} + \sum_{i=1}^{N}\sum_{j=1}^{N}\mu_{ij}(m_i(t_0)c_j(t_0)\tau_{\mathrm{th}}^{\mathrm{mc}} + c_i(t_0)m_j(t_0)\tau_{\mathrm{th}}^{\mathrm{cm}}) \le T, \tag{4}$$

where $\mathbf{c}$ is the offload indicator vector ($\mathbf{c}$=[$c_1(t_0)\ c_2(t_0)\ \ldots\ c_N(t_0)$]), and T is the number of time slots to complete processing the application. Since the energy consumed by local execution ($P_i^{\mathrm{m}}q_i^{\mathrm{m}}$), local memory consumption (M_i), and local CPU consumption ($\varepsilon(\mathrm{code}_i)$) are constant parameters, calculating the lower bound of the communication cost in Eqn (2) at initial time slot t_0 by $\hat{C}_{\mathrm{com}}(t_0)$ gives the upper bound of the net utility approximation considering all the potential components for offloading in the online stage.

By solving this optimization problem, $c_i^*(t_0)$ $\forall i$ is obtained which specifies if the component i must be offloaded ($c_i(t_0) = 1$) or not. If $c_i(t_0) = 1$ (equivalently, $m_i(t_0) = 1 - c_i(t_0) = 0$), then component i will be processed in the online stage (components $2, 3, 4, 6, 8, 9, 10, 11$, and 12 in Fig. 3b). Otherwise, it will be scheduled for local execution based on the precedence constraints dictated by the CDG for the application.

In $\mathrm{OP}_{\mathrm{off}}$, we have the terms $m_i(t_0)c_j(t_0)$ $\forall i, j$ in the cost function ($\hat{C}_{\mathrm{com}}(t_0)$), which makes the optimization problem nonlinear. To convert this to a linear optimization problem, we replace the terms $m_i(t_0)c_j(t_0)$, $\forall i, j$, with a new variable $z_{ij}(t_0)$ and add new constraints to make the new optimization problem equivalent to the original one [45]. These constraints are as follows: $z_{ij}(t_0) \le m_i(t_0), z_{ij}(t_0) \ge 0, z_{ij}(t_0) \le c_j(t_0), z_{ij}(t_0) \ge c_j(t_0) - (1 - m_i(t_0))$, where $z_{ij}(t_0)$ is the indicator specified at time t_0. This indicator is one if component i will be executed in the mobile device and component j will be executed in the cloud; otherwise, it is 0. The following subsections describe the online stage of the proposed heuristic.

B. Online Stage

In the online stage of the algorithm (starting from t_1 in Fig. 3b), the following two precedence constraints must be checked to see if a component is eligible for execution at the current time slot t: First, each component must be processed only once, either in the mobile or in the cloud. This constraint is mathematically written as:

$$m_i(t-1) + c_i(t-1) < 1, \qquad \forall i. \tag{5}$$

This equation shows that component i has not started execution (either locally or remotely) by time slot t. Second, for execution of component i at time slot t, we require that all the components on which component i depends, say $j \prec i$, should have completed execution before starting the offload process to component i, or local/cloud execution of component i. Therefore, the precedence constraints are:

$$\begin{aligned} &m_i(t - l_{ji}(t-1)) + c_i(t - l_{ji}(t-1)) \le \\ &m_j(t-1) + c_j(t-1), \\ &\forall j \prec i, \quad m_j(t-1) + c_j(t-1) = 1, \\ &t = l_{ji}(t-1) + 1 \ldots T, \end{aligned} \tag{6}$$

where $l_{ji}(t)$ is the number of time slots to process the preceding component j, either locally or remotely, and transfer the output data from component j to i by time slot t. Note that this time duration $l_{ji}(t)$ is a function of t, because the time taken to execute a component and to communicate relevant data are time dependent (because of varying mobile device resource availability and wireless data rates). The duration $l_{ji}(t)$ is expressed as: $l_{ji}(t) = m_j(t)q_j^{\mathrm{m}} + c_j(t)q_j^{\mathrm{c}} + \sum_{s=1}^{t}\sum_{k=1}^{K}\left(z_{ji}\alpha_k(s)\tau_{j,k}^{\mathrm{mc}}(s) + z_{ij}\beta_k(s)\tau_{j,k}^{\mathrm{cm}}(s)\right)$, where $\alpha_k(s)$ and $\beta_k(s)$ are the percentages of allocated rates for the mobile to the cloud and the cloud to the mobile, respectively, using radio interface k for offloading at time slot s. The first two terms on the RHS of $l_{ji}(t)$ are the execution time slots for component j in the mobile device and cloud, respectively, weighted by the respective indicator functions ($m_j(t)$ and $c_j(t)$). The third term represents the relevant data-offload time between components j and i. If $z_{ji}\alpha_k(s)$ is non-zero, then $\alpha_k(s)\tau_{j,k}^{\mathrm{mc}}(s)$ represents the time slots to transmit the allocated output data of component j in the mobile device using radio interface k, to the cloud where component i will be executed at time slot s. If $z_{ij}\beta_k(s)$ is non-zero, then $\beta_k(s)\tau_{j,k}^{\mathrm{cm}}(s)$ represents the time slots to send the part of the output data from component j in the cloud via radio interface k to the mobile device where component i will be executed at time slot s.

If these two constraints are satisfied for component i, then it is safe to execute it. Otherwise, component i is not ready for execution at this current time slot, and $X_i(t)$ and $I_i(t)$ are set to 0. Also note that it is possible to calculate $l_{ji}(t-1)$, because we have access to all the decision variables, such as

$\alpha_k(s), \beta_k(s)$, $X_i(s)$, and $I_i(s)$, for the previous time slots for $s \in \{1, 2, \ldots, t-1\}$.

1) Mobile Transmission Strategy: Once a component has been identified for offloading, then radio allocation for the transmission from the mobile to cloud must be computed. Since our cognitive cloud offloader works with multiple wireless interfaces at the same time, the stability of the data transmission buffers should be monitored to ensure no buffer overflows. Mathematically, this can be written as follows: $\overline{Q} = \lim_{T\to\infty} \sup \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\{|Q_k(t)|\} < \infty$, where $Q_k(t)$ is the transmission queue of wireless interface k at time slot t from the mobile side. We cast the above problem as a Lyapunov optimization [46] problem. The Lyapunov function is defined as $L(\boldsymbol{Q(t)}) = \frac{1}{2}\sum_{k=1}^{K} Q_k^2(t)$ where $\boldsymbol{Q(t)}$=$[Q_1(t)\ \ Q_2(t)\ \ \ldots\ \ Q_K(t)]$. While the queue of mobile transmission (which includes all data that must be transferred from the mobile device to the cloud) is updated with time, the Lyapunov drift will be $\Delta_{\mathrm{mc}}(\boldsymbol{Q(t)}) \triangleq \mathbb{E}\{L(\boldsymbol{Q(t+1)}) - L(\boldsymbol{Q(t)})|\boldsymbol{Q(t)}\}$. The Lyapunov drift is opportunistically minimized, taking into account the cost of the energy consumed for mobile transmission: $\Delta_{\mathrm{mc}}(\boldsymbol{Q(t)}) + V_{\mathrm{mc}}\mathbb{E}\{E_{\mathrm{com}}^{\mathrm{Tx}}(t)|\boldsymbol{Q(t)}\}$ [47], where V_{mc} is the control parameter for the queuing of the mobile transmission, considering the balance between the Lyapunov drift and the cost of energy consumed for transmission ($E_{\mathrm{com}}^{\mathrm{Tx}}(t)$), and $E_{\mathrm{com}}^{\mathrm{Tx}}(t) = \sum_{k=1}^{K} P_k^{\mathrm{Tx}}(t)\sum_{i=1}^{N} I_i(t)\alpha_k(t)\tau_{i,k}^{\mathrm{mc}}(t)$, where all the components i, prepared for offloading at time slot t have $I_i(t)$ set to one.

Lemma 1: As proved in [46], the upper bound of the Lyapunov drift is obtained by:

$$\begin{aligned}\Delta_{\mathrm{mc}}(\boldsymbol{Q(t)}) + V_{\mathrm{mc}}\mathbb{E}[E_{\mathrm{com}}^{\mathrm{Tx}}(t)|\boldsymbol{Q(t)}] \le \frac{(A_{\max}^{\mathrm{mc}})^2}{2} + \\ V_{\mathrm{mc}}\mathbb{E}\{E_{\mathrm{com}}^{\mathrm{Tx}}(t)|\boldsymbol{Q(t)}\} + \\ \sum_{k=1}^{K}\mathbb{E}\{Q_k(t)(\sum_{i=1}^{N} B_i^{\mathrm{mc}}(t) - A_k^{\mathrm{mc}}(t))|\boldsymbol{Q(t)}\}.\end{aligned} \tag{7}$$

where $A_k^{\mathrm{mc}}(t)$ represents data rate transmitted from the mobile device to the cloud through radio interface k at time slot t, $B_i^{\mathrm{mc}}(t)$ is the arrival data rate in the mobile transmission buffer at time slot t. In the paper, we use the parameter $A_k^{\mathrm{mc}}(t)$ to indicate the bandwidth allocated to the mobile device over wireless interface k. This time-adaptive parameter decreases if the demand from other mobile devices increases. Therefore, dynamic changes in $A_k^{\mathrm{mc}}(t)$ reflect the effect of ambient traffic produced from other mobile devices. Also, $B_i^{\mathrm{mc}}(t)$ will include both the data that the application needs to transfer due to offload operations of component i as well as other ambient data that the mobile generates and which is unrelated to the offloading. Also, $A_{\max}^{\mathrm{mc}}$ is the maximum transmitted data rate.

Following the Lyapunov optimization framework, the upper bound of the objective function in Eqn (7) must be minimized. This can be done by opportunistically minimizing the expectation [47, pp13]. Therefore, we use a lemma in [47, pp13] to simplify the RHS of Eqn (7) as:

$$\mathrm{OP_{Tx}} : \min_{\boldsymbol{\alpha}} \Omega_{\mathrm{mc}}(\mathrm{t}) = \mathrm{V_{mc}E_{com}^{Tx}} - \sum_{k=1}^{K}\big(Q_k(t)A_k^{\mathrm{mc}}(t)\big), \tag{8}$$

s.t.

$$\sum_{k=1}^{K}\alpha_k(t)\sum_{i=1}^{N} I_i(t)\tau_{i,k}^{\mathrm{mc}}(t) \le T_{\mathrm{th}}^{\mathrm{Tx}}, \tag{9}$$

$$\sum_{k=1}^{K}\alpha_k(t) = 1, \alpha_k(t) \ge 0, \forall k, \tag{10}$$

where $\boldsymbol{\alpha} = [\alpha_1(t)\ \alpha_2(t)\ \ldots \alpha_K(t)]$. In Eqn (8), introducing the parameter V_{mc} is a standard way to convert a constrained optimization problem to a single-equation unconstrained problem [16]. Here, V_{mc} is interpreted as a parameter that trades-off communication costs with decreasing the uplink communication energy ($E_{\mathrm{com}}^{\mathrm{Tx}}$) and satisfying the queue stability constraint for all the radio interfaces. The RHS of Eqn (8) is the representative of averaged aggregated queue length ($\overline{Q}$). Decreasing V_{mc} emphasizes on decreasing the averaged aggregated queue length, while increasing it enforces possible minimization of the transmission energy. Constraints (9) and (10) respectively ensure that the transmission time lies below a certain threshold, $T_{\mathrm{th}}^{\mathrm{Tx}}$, and that the required data is transferred through multi-RATs, but the summation of weights for radio interface allocation should be one.

The performance bounds of the transmission strategy based on the Lyapunov optimization [46] for the transmission queue stability and energy consumed for transmission, respectively, are expressed as: $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\overline{Q}_i(t) \le \frac{\frac{(A_{\max}^{\mathrm{mc}})^2}{2} + V_{\mathrm{mc}}E_{\mathrm{com}}^{*\mathrm{Tx}}}{\varepsilon_{\max}}, \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\overline{E}_{\mathrm{com}}^{\mathrm{Tx}}(t) \le \frac{(A_{\max}^{\mathrm{mc}})^2}{2V_{\mathrm{mc}}} + E_{\mathrm{com}}^{*\mathrm{Tx}}$, where $\overline{a}$ represents the mean value for parameter a, and $E_{\mathrm{com}}^{*\mathrm{Tx}}$ is the optimal value of $E_{\mathrm{com}}^{\mathrm{Tx}}$ obtained from solving the optimization problem in Eqn (8).

The offloading strategy (from the mobile to the cloud) in the online stage is as follows. For every component i, qualified for processing from the previous step, if the optimization problem $\mathrm{OP_{Tx}}$ has a solution for the variable parameter set $\boldsymbol{\alpha}$ in the feasible region, then component i is offloaded starting at time slot t ($I_i(t) = 1$) via K wireless interfaces at the optimal percentage values $\alpha_k^*(t)\ \forall k$ (e.g. component 2 at time t_2, component 6 and 11 at t_3 in the example of Fig. 3b). If the optimization problem does not have a feasible solution, then there are two options: (*i*) wait for the next time slot (e.g. components 6 and 11 wait at time t_2 in Fig. 3b); (*ii*) execute the component locally. The difference between the current time slot t and the time slot t_i^{req} that requested for processing component i should be much lower than the local execution time. This constraint is given by $|t - t_i^{\mathrm{req}}| < \gamma q_i^{\mathrm{m}}$, where γ is the weight factor. If the wait time does not exceed the local execution time for component i, then $I_i(t)$ is set to 0 and the component i is set aside to await its turn for execution. However, if $|t - t_i^{\mathrm{req}}| \ge \gamma q_i^{\mathrm{m}}$, the component is flagged for local execution in the next time slot, and will not be considered for offloading again ($c_i(T) = 0$).

In addition to updating the rates and latency values for each wireless interface in the time slots, the transmission queue for the next time slot for radio interface k ($\forall k$) needs to be updated as follows:

$$Q_k(t+1) = \max[Q_k(t) - A_k^{\mathrm{mc}}(t), 0] + \alpha_k(t)\sum_{i=1}^{N} B_i^{\mathrm{mc}}(t), \tag{11}$$

where the first term on the RHS of Eqn (11) represents the data remaining in the queue for interface k, and the second term represents the data arrival at radio interface k in time slot t. As shown in Eqn (11), queue is a function of percentage of allocated uplink. Also, note that if component i originally scheduled for remote execution is not offloaded at time slot t due to not finding a feasible solution for $\mathrm{OP_{Tx}}$ (meaning that energy and time constraints of offloading are not satisfied), then the delay values for transmission of the output data from component i in the mobile to the cloud via wireless interface k will be updated to: $\tau_{i,k}^{\mathrm{mc}}(t+1)+1$. This means that, after each time slot, if the scheduled component for remote execution is not offloaded, the delay cost will be updated by the delay value at the next time slot, plus one.

2) Cloud Transmission Strategy: Just as in the case of the mobile transmission, we also optimize the cloud transmission strategy taking into account delays and energy consumed by the mobile device for receiving this data. We optimally choose the percentage of data that needs to be allocated to each wireless interface to send the necessary information from the cloud to the mobile.

To ensure that no cloud Tx buffer overflows, the time-averaged summation of buffer occupancies must remain finite: $\overline{S} = \lim_{T\to\infty}\sup\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\{|S_k(t)|\} < \infty$, where $S_k(t)$ is the transmission queue via the cloud for wireless interface k at time slot t. The Lyapunov function for the cloud transmission strategy, which also reflects the receiver strategy for the mobile, can be written as $L(\boldsymbol{S(t)}) = \frac{1}{2}\sum_{k=1}^{K} S_k^2(t)$, where $\boldsymbol{S(t)} = [S_1(t)S_2(t)\dots S_K(t)]$. The Lyapunov drift in the data transfer from the cloud to the mobile is expressed as $\Delta_{\mathrm{cm}}(\boldsymbol{S(t)}) \triangleq \mathbb{E}\{L(\boldsymbol{S(t+1)}) - L(\boldsymbol{S(t)})|\boldsymbol{S(t)}\}$. This Lyapunov drift is opportunistically minimized, considering the penalty of energy consumed for downlink mobile reception as $\Delta_{\mathrm{cm}}(\boldsymbol{S(t)}) + V_{\mathrm{cm}}\mathbb{E}\{E_{\mathrm{com}}^{\mathrm{Rx}}(t)|\boldsymbol{S(t)}\}$, where V_{cm} is the control parameter in data transfer from the cloud to the mobile, while the trade-off between the Lyapunov drift of the cloud transmission queue and the penalty of energy consumed for mobile reception ($E_{\mathrm{com}}^{\mathrm{Rx}}(t)$) is applied, and $E_{\mathrm{com}}^{\mathrm{Rx}}(t) = \sum_{k=1}^{K} P_k^{\mathrm{Rx}}(t)\sum_{i=1}^{N}\beta_k(t)\tau_{i,k}^{\mathrm{cm}}(t)$, Following Lemma 1, the upper bound of the objective function for the Lyapunov drift of data transfer from the cloud to the mobile is obtained by:

$$\begin{aligned}\Delta_{\mathrm{cm}}(\boldsymbol{S(t)}) + V_{\mathrm{cm}}\mathbb{E}[E_{\mathrm{com}}^{\mathrm{Rx}}(t)|\boldsymbol{S(t)}] \le \frac{(A_{\max}^{\mathrm{cm}})^2}{2} + \\ V_{\mathrm{cm}}\mathbb{E}\{E_{\mathrm{com}}^{\mathrm{Rx}}(t)|\boldsymbol{S(t)}\} + \\ \sum_{k=1}^{K}\mathbb{E}\{S_k(t)(\sum_{i=1}^{N} B_i^{\mathrm{cm}}(t) - A_k^{\mathrm{cm}}(t))|\boldsymbol{S(t)}\},\end{aligned} \tag{12}$$

where $A_k^{\mathrm{cm}}(t)$ represents the data rate transmitted from the cloud to the mobile device through radio interface k at time slot t, $B_i^{\mathrm{cm}}(t)$ is the arrival data rate in the cloud transmission buffer at time slot t. $B_i^{\mathrm{cm}}(t)$ will include both the data that the cloud transfers due to offload operations of component i as well as other ambient data. Also, $A_{\max}^{\mathrm{cm}}$ is the maximum data rate from the cloud. After simplifying the upper bound on the RHS of Eqn (12) and using the concept of opportunistically minimizing the expectation [47, pp13], the objective function of the optimization problem for the cloud transmission strategy, which reflects the mobile reception status, is obtained. The optimal strategy can be written as the solution to the following optimization problem considering the delay constraint from the cloud to the mobile:

$$\mathrm{OP_{Rx}} : \min_{\boldsymbol{\beta}} \Omega_{\mathrm{cm}}(\mathrm{t}) = \mathrm{V_{cm}}\mathrm{E_{com}^{Rx}} - \sum_{k=1}^{K}\big(S_k(t)A_k^{\mathrm{cm}}(t)\big), \tag{13}$$

s.t.

$$\sum_{k=1}^{K}\beta_k(t)\sum_{i=1}^{N}\tau_{i,k}^{\mathrm{cm}}(t) \le T_{\mathrm{th}}^{\mathrm{Rx}}, \tag{14}$$

$$\sum_{k=1}^{K}\beta_k(t) = 1, \beta_k(t) \ge 0, \forall k, \tag{15}$$

where $\boldsymbol{\beta} = [\beta_1(t) \quad \beta_2(t) \quad \dots\beta_K(t)]$. In Eqn (13), V_{cm} is interpreted as a parameter that trades-off communication costs with decreasing the downlink communication energy ($E_{\mathrm{com}}^{\mathrm{Rx}}$) and satisfying the queue stability constraint for all the radio interfaces ($\overline{S}$). Constraints (14) and (15), respectively, ensure that the latency from the cloud to the mobile lies below a certain threshold, $T_{\mathrm{th}}^{\mathrm{Rx}}$, and that the required data is optimally transmitted from the cloud via multiple interfaces and that the weights sum to unity. Also, the performance bounds for the cloud transmission queue and energy consumed by the mobile receiver, respectively, are given as: $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\overline{S}_i(t) \le \frac{\frac{(A_{\max}^{\mathrm{cm}})^2}{2}+V_{\mathrm{cm}}E_{\mathrm{com}}^{*\mathrm{Rx}}}{\varepsilon_{\max}}$, $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\overline{E}_{\mathrm{com}}^{\mathrm{Rx}}(t) \le \frac{(A_{\max}^{\mathrm{cm}})^2}{2V_{\mathrm{cm}}} + E_{\mathrm{com}}^{*\mathrm{Rx}}$.

The offloading strategy (cloud transmission) in the online stage is as follows. For every component i from which data must be received at the mobile device, if the optimization problem $\mathrm{OP_{Rx}}$ has a solution for the variable parameter set $\boldsymbol{\beta}$ in the feasible region, then component i is transmitted by the cloud, at time slot, t, via K wireless interfaces at the optimal percentage values $\beta_k^*(t)\ \forall\ k$ (e.g. component 3 at time slots t_3 in the example of Fig. 3b). If the optimization problem $\mathrm{OP_{Rx}}$ does not have a feasible solution, then transmission from the cloud is scheduled for the next time slot. Note that if component i scheduled for remote execution is not transmitted by the cloud to mobile at time slot t, then the delay values of the output data from cloud to mobile for component i via wireless interface k will be updated to: $\tau_{i,k}^{\mathrm{cm}}(t+1)+1$. At the end of current time slot t, the data queues in Tx buffer for the cloud at the next time slot for radio interface k ($\forall k$) is updated as follows:

$$S_k(t+1) = \max[S_k(t) - A_k^{\mathrm{cm}}(t), 0] + \beta_k(t)\sum_{i=1}^{N} B_i^{\mathrm{cm}}(t). \tag{16}$$

Note that $\mathrm{OP_{off}}$, $\mathrm{OP_{Tx}}$, and $\mathrm{OP_{Rx}}$ are solved using the simplex algorithm [48] in practice.

Algorithm 1 Cognitive Offloading and Scheduling Heuristic.

```
1: **Offline stage** at $t_0$ (optional) (output: c=[$c_1(t_0)$ $c_2(t_0)$ ... $c_N(t_0)$]):
2: Solve $\mathrm{OP_{off}}$ to calculate which components can be offloaded
3: otherwise all components are analyzed for offloading (except 1, $N$)
4: **Online stage** (output: $X_i(t)$, $I_i(t)$, $\alpha_k(t)$, $\beta_k(t)$):
5: **repeat** $t \to t+1$
6:     Check scheduling constraints given by Eqns (5) & (6)
7:     For offloadable components:
8:     Solve $\mathrm{OP_{Tx}}$ (output: $\alpha_k(t)$)
9:     **if** $\mathrm{OP_{Tx}}$ has feasible solution **then**
10:        offload related components with the corresponding values for $\alpha_k(t)$ $\forall k$ obtained by $\mathrm{OP_{Tx}}$
11:    **else if** $|t - t_i^{\mathrm{req}}| < \gamma q_i^{\mathrm{m}}$ **then**
12:        wait for the next time slot to check offloading
13:    **else if** $|t - t_i^{\mathrm{req}}| \geq \gamma q_i^{\mathrm{m}}$ **then**
14:        component $i$ will be scheduled for local execution
15:    **end if**
16:    Solve $\mathrm{OP_{Rx}}$ (output: $\beta_k(t)$)
17:    **if** $\mathrm{OP_{Rx}}$ has feasible solutions **then**
18:        Send output data from the cloud with corresponding values for $\beta_k(t)$ $\forall k$ obtained by $\mathrm{OP_{Rx}}$
19:    **else**
20:        Do not send data from cloud and wait for the next slot
21:    **end if**
22:    For the components scheduled for local execution:
23:    **if** local execution constraint given by Eqn (17) is satisfied **then**
24:        Execute component $i$ locally at time slot $t$
25:    **end if**
26:    Update $Q_k(t+1)$ $\forall k$ using Eqn (11)
27:    Update $S_k(t+1)$ $\forall k$ using Eqn (16)
28:    Add delay ($\tau_{i,k}^{\mathrm{mc}}(t+1)$ $\forall k$) for waiting components in the mobile
29:    Add delay ($\tau_{i,k}^{\mathrm{cm}}(t+1)$ $\forall k$) for waiting components in the cloud
30: **until** $t = T$.
```

3) Local Execution: As mentioned earlier, some components are selected for local execution (e.g. components 1, 4 and 5 at time slots t_1, t_2, and t_4, respectively, in the example of Fig. 3b). Although the cloud can execute several components in parallel, we assume that the mobile device processes components serially. In order to schedule component i on the mobile device, at the current time slot, t, we need to ensure that no other application's component is currently running on the mobile. This is expressed as:

$$\sum_{i=1}^{N} \sum_{s=t-1-q_i^{\mathrm{m}}}^{t-1} X_i(s) < 1. \tag{17}$$

V. Performance Analysis

In this section, we first introduce the evaluation setup for the performance analysis of the proposed cognitive offloading and scheduling heuristic. Then the methods for comparisons are introduced. Finally, results of the simulation results are provided and discussed.

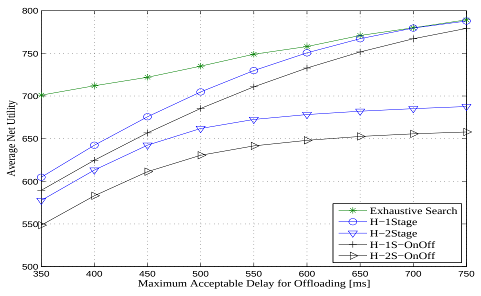
A. Real Data Measurements and Simulation Setup

All our experiments were run on an Android HTC Vivid smartphone. This device is equipped with a 1.2 GHz dual-core processor. Although our theory is developed for a general, K, number of radio interfaces, our experiments were conducted using two wireless radio interfaces for cloud offloading: WiFi and LTE ($K = 2$). The Amazon Elastic Compute Cloud (Amazon EC2) was used for cloud computing. In all the simulations, except the simulations in Figs 6 and 7, a 14-component video navigation mobile application ($N = 14$) was used, in which 4 components are used for graphic features [40], 3 are used for face detection [41], 6 are used for video processing [42], and the last component is used for clustering the video points and showing the output results. Fig. 3a shows the CDG for this real application. In the simulations of Fig. 7, synthetic applications with arbitrary CDGs were used in order to test the proposed scheme for large number of components as well as different types of CDG structures [49]. Also in Fig. 6, the face recognition application in [50] was used. The average wireless service rates and latencies were obtained by using the TCPdump tool. Please note that If monetary constraints are important to the end user, then the weights assigned to the individual interfaces can be considered to reflect this. In some cases the solution can be weighted accordingly. Example of the second kind of scenario includes business clients who often travel with their subscription LTE connections wherever possible. In tactical and emergency response situation, the most important consideration is to get the job done, rather than minimizing the monetary costs. Our proposed solution can work for all these use cases. Also, we added a Poisson distributed background arrival data in the mobile device at each time slot [16] to simulate ambient traffic not related to this particular application. The average transmission and reception power levels of the mobile device for WiFi service are respectively 257.83 and 123.74 mW, and for LTE service are respectively 356.1 and 197.1 mW. The active and idle power levels of the phone are 644.9 and 22 mW, respectively. Running the multi-component is based on the following process: Component 1 starts the application in the mobile device as the starter of the application. It sends data to the components 2, 4, 6, 11, 12, and 13. Components 2, 3, 4, and 5 compute graphic features in a series-parallel dependency graph. Component 6, that is the camera view, distributes jobs for video processing including feature extraction and classification, mixed processing, camera control, and image manipulations. In parallel, object detection and recognition is done via components 11, 12, and 13. Finally the last component clusters the results and shows the output. The run times for each component in the mobile and cloud are obtained as averages of 10 independent runs. We obtained the average runtime measurements of each component in both of the mobile device and the cloud. The local execution time of the 14 components (for the 14-component application) were measured as $[30\ 340\ 345\ 125\ 30\ 80\ 70\ 30\ 185\ 125\ 650\ 571\ 904\ 56]$ ms. The power measurements were obtained by using CurrentWidget: Battery monitor application [42]. Since processing of the components in the mobile device is performed serially, the local execution time for the application is simply the sum of the processing times of the individual components (3541 ms). We set the duration for each time slot (x) as 35 ms; that is, the online stage ran every 35 ms. If x is too long compared to wireless network parameter variations, then the solutions will not be sensitive to changes in wireless parameters, and if x is too short, then the optimization runs several times thereby costing more to run the optimization. The ideal value of x would strike a comprise between these two factors. We applied uniform distribution for the weights (e.g. $w_{\text{saved}} = w_{\text{com}}$=0.5 and also $w_{\text{memory}} = w_{\text{CPU}}$=0.5). The graphs obtained in this section are averages of 1000 independent runs.

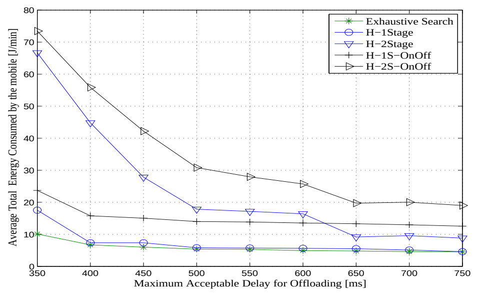
B. *Versions of the Proposed Heuristics*

We compare our proposed work to several other scenarios: (1) no offload (mobile-only) execution, (2) all offload (cloud-only) execution, (3) the dynamic offloading algorithm (DOA) in [14], and several variants of offload strategies that we propose below:

- **Exhaustive Search:** In this scheme, the proposed optimization problem is solved using brute force – by evaluating all possibilities and picking the best value. This gives an upper bound of performance for all algorithms. There is no offline stage to select the preferred components for offloading, and all components can potentially be offloaded in the online stage (except components 1 and N). Also the optimal offloading strategy in the heuristic for multi-RATs is obtained by searching exhaustively over all possible transmission strategies to guarantee the maximum net utility rather than using OP_{Tx}, OP_{Rx} for multi-RATs (Subsections *1* and *2* in Section IV). Note that this exhaustive search also includes the local execution of components. This method is computationally prohibitive for large and complex applications but serves as a benchmarking tool. This is provided here only to give an idea of the performance of the heuristics vis-a-vis the optimal solution. The output from this exhaustive search gives the gap between optimal solution (the upper bound) and the proposed heuristic schemes.
- **Proposed Heuristic with No Offline Stage (H-1Stage):** This strategy essentially eliminates the offline stage and proceeds with the rest of the heuristic algorithm where all components are eligible for offloading.
- **Two-Stage Heuristic Algorithm (H-2Stage):** This is the proposed algorithm in which the preprocessing stage (described in Section IV-A) is used to eliminate some of the components from being considered for offloading. While this reduces the time complexity because fewer components are processed in the heuristic, it may eliminate some eligible components from being considered for offloading in the online stage, thereby sacrificing the net utility. The assignment of preferred components for offloading is performed using the optimizing strategy (OP_{off}) mentioned in Section IV. Note that since the offline optimization problem (OP_{off}) does not use instantaneous values of the parameters, this scheme will be suboptimal in comparison to H-1Stage. However, there are fewer components for offloading, so system complexity is lower and the algorithm is faster.
- **Proposed Single Stage Heuristic under On-Off model for the Wireless Interfaces (H-1S-OnOff):** In this scheme, we use the proposed single stage heuristic, where all components are considered for offloading at the online stage using only the wireless interface with the best characteristics at that time slot. Although [14] and [16] also use On-Off model for the wireless interface, they are different from this proposed variant. In [16], the entire application is offloaded and only the cloud strategy is optimized; moreover, it is not a joint scheduling-offloading scheme in that, it does not determine the scheduling order for the components but rather uses a pre-determined, serial execution order for the components, and [14] is not an online strategy.
- **Proposed Two-Stage Heuristics with ON/OFF Wireless Interfaces (H-2S-OnOff):** In this scheme, we use the two-stage algorithm for offloading, but we only use the wireless interface with the best characteristics for data transfers of both the mobile to cloud and the cloud to mobile.



(a)

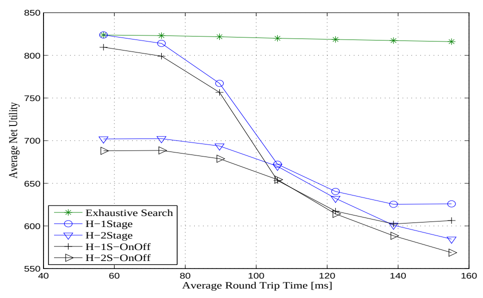


(b)

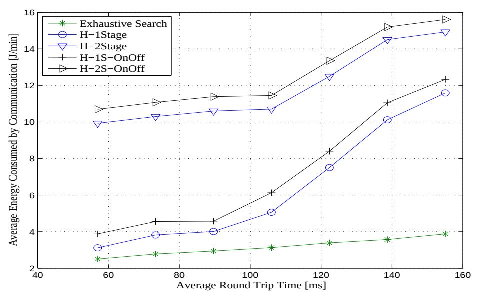
Fig. 4: Average (a) net utility & (b) mobile energy consumption versus maximum acceptable delay for computation offloading.

C. *Results and Discussion*

We first study the average net utility (Eqn (1)) as a function of maximum acceptable delay for offloading (T_{th}) for the



(a)



(b)

Fig. 5: Average (a) net utility & (b) mobile energy consumption versus average round trip time (execution deadline of the application= 1330 ms, time threshold for offloading= 550 ms).

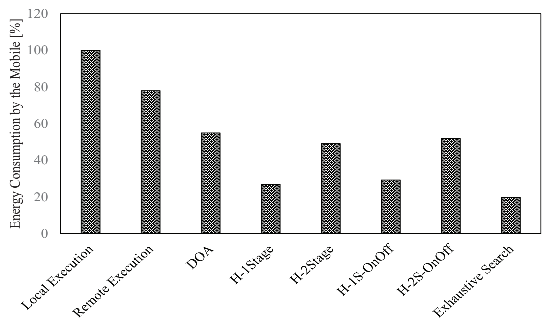


Fig. 6: Total energy consumed by the mobile device for the proposed and classical schemes, normalized to the energy consumed by local execution (using the face recognition application in [50]).

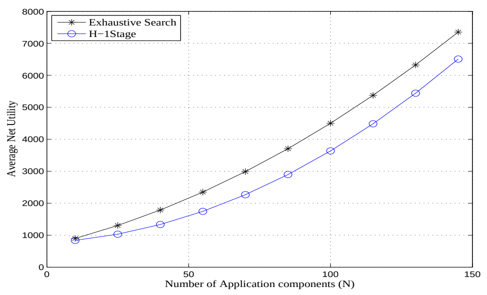
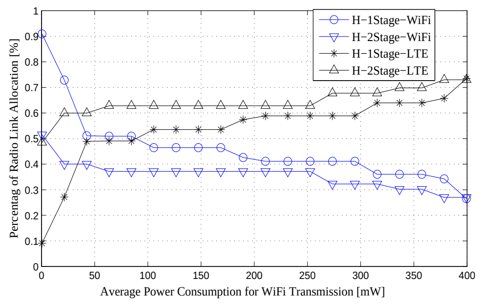


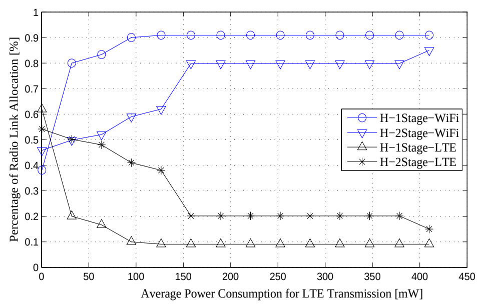
Fig. 7: Average net utility versus number of application's components [10 25 40 55 70 85 100 115 130 145] where CDGs are based on a random graph of Fan-in/Fan-out (execution deadline of the application= 1330 ms).

five schemes (see Fig. 4a). We set the execution deadline of the application ($x \times T$) to 1330ms (much lower than application runtime which is 3541 ms), and the transmission time threshold $T_{\text{th}}^{\text{Tx}}$ and the reception time threshold $T_{\text{th}}^{\text{Rx}}$ respectively to 55% and 45% of the maximum acceptable delay for offloading (T_{th}). It is observed that while the maximum acceptable delay of wireless interfaces increases, more components are scheduled to be offloaded, more resources are saved, and the mobile device achieves higher net utility. The plots in this figure also show that we achieve the highest net utility using the proposed multi-RAT optimized single stage heuristics. Note also, that the proposed single stage heuristics that operates in an ON-OFF mode performs better than the two-stage heuristics. As the maximum acceptable delay for offloading increases, the proposed heuristics converges to the optimal solution obtained by the exhaustive search solution. In Fig. 4b, we plot the time average of the total energy consumed by the mobile device versus the maximum acceptable delay for offloading. The total energy is the sum of mobile active energy, energy consumed during communication, and the idle energy of the mobile device. The trend observed here is similar to that which was observed in the net utility: as the acceptable offload delay increases, the energy consumed by the mobile device also decreases. We can also see that for higher acceptable delays, H-2Stage consumes less energy compared to H-1S-OnOff. This shows that using cognitive networking in this range consumes less energy even when the suboptimal offline stage in H-2Stage is used.

In Fig. 5a, the average net utility (Eqn (1)) is illustrated as a function of the average round trip time (RTT). This RTT is calculated on mean values of delays (in units of time slot) over WiFi ($\tau_{i,1}^{\text{mc}}(t), \tau_{i,1}^{\text{cm}}(t), \forall i, t$) and LTE ($\tau_{i,2}^{\text{mc}}(t), \tau_{i,2}^{\text{cm}}(t), \forall i, t$) interfaces. The execution deadline of the application ($x \times T$) and the maximum acceptable delay for offloading ($x \times T_{th}$) are set to 1330 ms and 550 ms, respectively. It is observed that while delay increases, the energy and time costs for cloud offloading increase, and therefore the average net utility decreases in all five schemes. We can see that again the proposed scheme with cognitive networking outperforms the schemes with ON-OFF link strategy. Note that although H-2Stage uses offline stage to decrease the system complexity, it gives higher net utility in comparison to H-1S-OnOff in the upper ranges of latencies. In this figure, we see that the decrease rate of net utility in exhaustive search scheme is lower in comparison with the heuristic schemes. Fig. 5b shows the average energy consumed for offloading versus the average RTT for the five schemes. We can see that while latency increases, more energy is consumed for communication. In Fig. 6, we compare the total energy consumption of the five proposed schemes with three other schemes: (*i*) complete local execution; (*ii*) complete remote execution; and (*iii*) the dynamic offloading algorithm (DOA) proposed in [14], which uses ON-OFF multi-RAT strategy. A face recognition application with 10 sequential components was used [50]. WiFi and 3G interfaces were used for this bar graph, the wireless network parameters in [51] are used such that exactly the same parameters used for the simulation of DOA in [14] were used for all the other seven schemes. This comparison is normalized to the scheme with local execution of all the components. We see

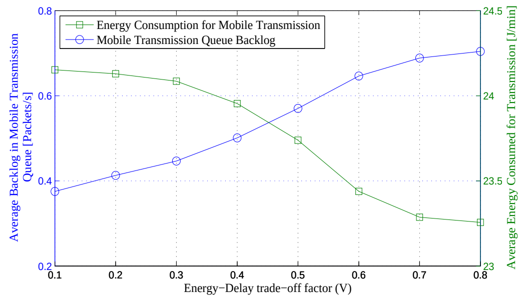


(a)

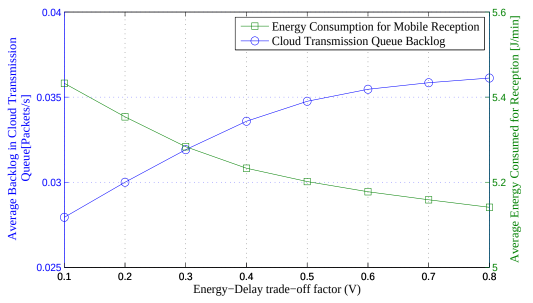


(b)

Fig. 8: Percentage of radio interface allocation versus time average power consumption for (a) WiFi & (b) LTE transmission by the mobile device.



(a)



(b)

Fig. 9: The impact of energy-delay trade-off factor on the average values of (a) mobile & (b) cloud transmission queue backlog and energy consumed for (a) uplink & (b) downlink offloaded data (the maximum acceptable delay for offloading= 550 ms, execution deadline of the application= 1330 ms).

that H-1Stage consumes 73%, 51%, 28%, and 3% less energy in comparison to the schemes using local execution, remote execution, DOA, and H-1S-OnOff, respectively, and consumes 8% more energy in comparison to the global optimum obtained using exhaustive search. In comparison of the schemes that use offline stage (H-2Stage, H-1S-OnOff, and DOA), H-2Stage consumes 2.5% and 6% less energy in comparison to H-1S-OnOff and DOA, respectively. H-2Stage outperforms H-2S-OnOff because it takes advantage of cognitive networking. Although the initial offline solution is applied in all the three schemes, the strategy can be modified in the online stage for H-2Stage and H-2S-OnOff. Therefore, using either of these schemes consumes less energy than using DOA. Overall, it is clear from this bar graph that the mobile device consumes lower energy by considering online dynamics of wireless networks in the offload strategy (heuristic strategies) in comparison with classical schemes (local execution, remote execution, and DOA).

We now investigate the scalability of our approach w.r.t the number of components in the application. Fig. 7 plots the average net utility as a function of the number of application components. We conducted this experiment for random applications with CDGs obtained based on a random Fan-in/Fan-out graph with in-degree and out-degree of one [49]. The maximum acceptable delay for offloading and RTT are set to 550 ms and 100 ms, respectively. We see that while the number of components increases, the complexity of the application (higher execution times and more component dependencies) increases so that higher net utility is saved. Figs 8a and 8b show the percentages of radio interface allocation for both WiFi and LTE versus the time-average transmit power of WiFi and LTE interfaces, respectively. Execution deadline of the application, the maximum acceptable delay for offloading, and RTT are set to 1330 ms, 550 ms and 100 ms, respectively. In Fig 8a, results show that when the WiFi transmit power increases, the percentage of WiFi allocation decreases and percentage of LTE interface allocation increases. We can see that in the very low ranges of WiFi transmit power, much higher percentages of WiFi are allocated for offloading in the H-1Stage in comparison to those of H-2Stage. On the other hand, in upper ranges of WiFi transmit power, we observe that the performance of both schemes are close to each other while the components decided for offloading are almost the same. In Fig 8b, we observe that when average LTE transmit power increases, the percentage of interface allocation for LTE decreases and percentage of interface allocation for WiFi increases. In Figs 9a and 9b, queue backlogs of mobile transmission buffers and cloud transmission buffers are presented respectively as functions of the trade-off control factors for energy and delay ($V = V_{\text{ul}} = V_{\text{cm}}$) where the H-1Stage scheme is applied. Application deadline, the maximum acceptable delay, and RTT are set to 1330 ms, 550 ms and 100 ms, respectively. In the same plots, the energy values consumed by the mobile device to transmit and receive offloaded data are presented. Looking at Eqns (8) and (13), we can see that Lyapunov control parameter, V, reflects the weight of transmission (reception) energy by the mobile device in comparison to the aggregated queue backlog for transmission to (from) the cloud through all radio interfaces. We observe that when the V increases, the queue backlog increases and less energy is consumed by the mobile device

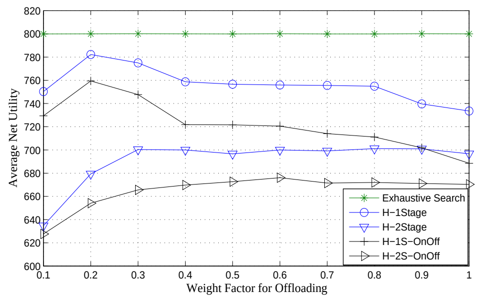


Fig. 10: Average net utility versus weight factor (to adjust the wait time for offloading), γ (the maximum acceptable delay for offloading$=550$ ms, execution deadline of the application$=1330$ ms).

for offloading. The intersection points in these plots show the value of V that achieves minimum transmission (reception) energy while sending (receiving) the highest amount of data for offloading. Finally, Fig. 10 shows the time-averaged net utility as a function of the weight factor to adjust the wait time for offloading (γ) in the five schemes. Execution deadline of the application, the maximum acceptable delay, and RTT are set to 1330 ms, 550 ms, and 100 ms, respectively. When the weight factor increases, the time to wait for offloading the components increases. The γ value that give the maximum net utility for the H-1Stage, H-1S-OnOff, H-2Stage and H-2S-OnOff are 0.2, 0.2, 0.3, and 0.6, respectively. However, we observe that the net utility changes by less than 1.25% in the exhaustive search scheme with γ. Please note that the exhaustive search, obtained by brute force technique, is independent of γ.

VI. Conclusion & Future Work

In this paper, we proposed a new concept of cognitive cloud offloading and introduced online heuristic strategies for computation offloading of multi-component applications in multi-RAT enabled mobile devices. The changes in wireless parameters at all available interfaces (e.g. rate, delay, power) are taken into account while determining the scheduling order of the multiple components of the application with arbitrary component dependency graphs. To obtain the best strategies for uplink and downlink offloading, the percentages of data to be sent by the mobile and the cloud via each wireless interface are updated at each time slot. We presented a comprehensive model for the net utility. Satisfying the constraints for schedule order of applications with arbitrary CDGs, the strategy for either offloading or locally executing the components is provided in each time slot using the updated knowledge of wireless parameters. Note that queue stabilities of the mobile and cloud transmission buffers for aggregated interfaces are guaranteed in the offloading strategy. We analyzed performance of the heuristic strategy with real data measurements using a video navigation application on an HTC Vivid smartphone. Several versions of the heuristic strategy were compared with the globally optimal scheme (obtained by exhaustive search), a recent strategy in dynamic computation offloading [14], and the classical strategies where all the components are executed locally and remotely. We observed that our proposed strategy (H-1Stage) consumes 28%, 51%, and 73% less energy in comparison to DOA, remote execution scenario, and local execution scenario, respectively. It consumes only 8% more energy than the optimal solution (obtained via exhaustive search). The output from this exhaustive search gives the gap between optimal solution and the proposed heuristic schemes while an upper bound of performance for all algorithms is obtained. This work will be extended in the future to obtain the overall optimal joint cognitive offloading and scheduling problem, where all constraints including precedence of components, offloading costs, dynamic queuing, overall execution will be optimally traded off.

References

[1] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, “Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications,” *IEEE Wireless Communications*, vol. 20, no. 3, pp. 14–22, June 2013.

[2] N. Vallina-Rodriguez and J. Crowcroft, “Energy management techniques in modern mobile handsets,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 179–198, First quarter 2013.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “Clonecloud: elastic execution between mobile device and cloud,” in *Proceedings of the sixth conference on Computer systems*, 2011, pp. 301–314.

[4] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, “Adaptive offloading for pervasive computing,” *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 66–73, Third quarter 2004.

[5] P. Demestichas, A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong, and J. Yao, “5g on the horizon: Key challenges for the radio-access network,” *IEEE Vehicular Technology Magazine*, vol. 8, no. 3, pp. 47–53, September 2013.

[6] O. Barak and A. Touboul, “Point to point link and communication method,” Sep. 22 2009, uS Patent 7,593,729. [Online]. Available: https://www.google.com/patents/US7593729

[7] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, “Improving energy efficiency of mptcp for mobile devices,” *arXiv preprint arXiv:1406.4463*, 2014.

[8] K. Johansson, J. Bergman, D. Gerstenberger, M. Blomgren, and A. Wallén, “Multi-carrier HSPA evolution,” in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*. IEEE, 2009, pp. 1–5.

[9] S. Merlin, N. Vaidya, and M. Zorzi, “Resource allocation in multi-radio multi-channel multi-hop wireless networks,” in *The IEEE Conference on Computer Communications (INFOCOM)*, April 2008, pp. 1283–1291.

[10] V. Bhandari and N. H.Vaidya, “Scheduling in multi-channel wireless networks,” in *Distributed Computing and Networking*, ser. Lecture Notes in Computer Science, K. Kant, S. Pemmaraju, K. Sivalingam, and J. Wu, Eds. Springer Berlin Heidelberg, 2010, vol. 5935, pp. 6–17.

[11] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, “When mobile terminals meet the cloud: computation offloading as the bridge,” *IEEE Network*, vol. 27, no. 5, pp. 28–33, September 2013.

[12] S. E. Mahmoodi, K. P. Subbalakshmi, and V. Sagar, “Cloud offloading for multi-radio enabled mobile devices,” in *IEEE International Communication Conference (ICC)*, June 2015, pp. 1–6.

[13] D. Huang, P. Wang, and D. Niyato, “A dynamic offloading algorithm for mobile computing,” *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, June 2012.

[14] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, “Energy-optimal mobile cloud computing under stochastic wireless channel,” *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, September 2013.

[15] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, “eTime: Energy-efficient transmission between cloud and mobile devices,” in *IEEE Proceedings of INFOCOM*, April 2013, pp. 195–199.

[16] Y. Lin, E. Chu, Y. Lai, and T. Huang, “Time-and-Energy-Aware computation offloading in handheld devices to coprocessors and clouds,” *IEEE Systems Journal*, vol. 9, no. 2, pp. 393–405, June 2015.

[17] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “MAUI: Making smartphones last longer with code offload,” in *International Conference on Mobile Systems, Applications, and Services*. ACM, 2010, pp. 49–62.

[18] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *IEEE Proceedings of INFOCOM*, 2012, pp. 945–953.

[19] D. Kovachev, T. Yu, and R. Klamma, "Adaptive computation offloading from mobile devices into the cloud," in *IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2012, pp. 784–791.

[20] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "COSMOS: Computation offloading as a service for mobile devices," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '14. New York, NY, USA: ACM, 2014, pp. 287–296.

[21] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Computation offloading for mobile cloud computing based on wide cross-layer optimization," in *Future Network and Mobile Summit (FutureNetworkSummit)*, July 2013, pp. 1–10.

[22] W. Gao, Y. Li, H. Lu, T. Wang, and C. Liu, "On exploiting dynamic execution patterns for workload offloading in mobile cloud applications," in *IEEE 22nd International Conference on Network Protocols (ICNP)*, October 2014, pp. 1–12.

[23] H. Wu, Q. Wang, and K. Wolter, "Trade-off between performance improvement and energy saving in mobile cloud offloading systems," in *IEEE International Conference on Communications Workshops (ICC)*, June 2013, pp. 728–732.

[24] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein, "Scheduling to minimize average completion time: Off-line and on-line approximation algorithms," 1996.

[25] M. Nir, A. Matrawy, and M. St-Hilaire, "An energy optimizing scheduler for mobile cloud computing environments," in *IEEE Conference on Computer Communications Workshops (INFOCOM)*, April 2014, pp. 404–409.

[26] P. Balakrishnan and C.-K. Tham, "Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing," in *IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC)*, December 2013, pp. 34–41.

[27] W. Zhang, Y. Wen, and D. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 81–93, January 2015.

[28] ——, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *IEEE Proceedings of INFOCOM*, April 2013, pp. 190–194.

[29] C. Shi, P. Pandurangan, K. Ni, J. Yang, M. Ammar, M. Naik, and E. Zegura, "IC-Cloud: Computation offloading to an intermittently-connected cloud," *SCS Technical Report in Georgia Institute of Technology*, 2013. [Online]. Available: http://hdl.handle.net/1853/45985

[30] S. Deng, L. Huang, J. Taheri, and A. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, Early access 2015.

[31] A. S. M. Toma and J.-J. Chen, "Computation offloading for frame-based real-time tasks under given server response time guarantees," *Leibniz Transactions on Embedded Systems*, vol. 1, no. 2, pp. 1–21, November 2014.

[32] M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," *REAL-TIME SYSTEMS*, vol. 10, pp. 179–210, 1996.

[33] K. Ramachandran, E. Belding, K. Almeroth, and M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–12.

[34] S. S. Paolo Di Lorenzo, Sergio Barbarossa, "Joint optimization of radio resources and code partitioning in mobile cloud computing," *IEEE Transactions on Mobile Computing*, Under second round of review, arXiv preprint arXiv:1307.3835 2015.

[35] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," in *IEEE Transactions on Cloud Computing*, accepted, http://personal.stevens.edu/ smahmood/MahUmaSub15TCCSI.pdf 2016.

[36] M. Zorzi, A. Zanella, A. Testolin, M. D. F. D. Grazia, and M. Zorzi, "Cognition-based networks: A new perspective on network optimization using learning and distributed intelligence," *IEEE Access*, vol. 3, pp. 1512–1530, August 2015.

[37] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. Irwin, and R. Chandramouli, "Studying energy trade-offs in offloading computation/compilation in java-enabled mobile devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 9, pp. 795–809, September 2004.

[38] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[39] [Online]. Available: http://www.opengl.org/.

[40] [Online]. Available: http://www.developer.com/ws/android/programming/face-detection-with-android-apis.html.

[41] [Online]. Available: http://opencv.org/.

[42] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proceedings of the 12th Conference on Hot Topics in Operating Systems*, ser. HotOS'09, 2009, pp. 8–8.

[43] J. K. Ousterhout, "Scripting: Higher level programming for the 21st century," *IEEE Computer magazine*, March 1998.

[44] P. Rubin. [Online]. Available: http://orinanobworld.blogspot.de/2010/10/binary-variables-and-quadratic-terms.html.

[45] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[46] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, April 2006.

[47] D. G. Luenberger, *Introduction to linear and nonlinear programming*. Addison-Wesley Reading, MA, 1973, vol. 28.

[48] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J.-M. Vincent, and F. Wagner, "Random graph generation for scheduling simulations," *Proceedings of the International ICST Conference on Simulation Tools and Techniques*, vol. 10, no. 60, pp. 60:1–60:10, 2010.

[49] [Online]. Available: http://darnok.org/programming/face-recognition/.

[50] [Online]. Available: http://www.3gpp.org/ftp/tsg-ran/wg4-radio/.

S. Eman Mahmoodi is currently pursuing his PhD degree at the Department of Electrical and Computer Engineering, Stevens Institute of Technology. He received the BS and MS degree in Electrical Engineering from Iran University of Science and Technology, respectively in 2009 and 2012. He has been working on Mobile Cloud Computing, Optimization and Applied Modeling, Cognitive Networks, and Wireless Communications. Mahmoodi is a Stevens Innovation and Entrepreneurship Doctoral Fellow.

K. P. (Suba) Subbalakshmi is a Professor at Stevens Institute of Technology, and will serve as a Jefferson Science Fellow in 2016. Her research interests span: Cognitive radio networks, Cognitive Mobile Cloud Computing, Social Media Analytics and Wireless security. She is a Founding Associate Editor of the IEEE Transactions on Cognitive Communications and Networking and an Associate Editor of the IEEE Transactions on Vehicular Technology. She is the Founding Chair of the Security Special Interest Group of the IEEE Technical Committee on Cognitive Networks. She is also a recipient of the NJIHOF Innovator award.