Signals and Communication Technology

Seyed Eman Mahmoodi
Koduvayur Subbalakshmi
R. N. Uma

Spectrum-Aware Mobile Computing

Convergence of Cloud Computing and Cognitive Networking

Springer

Signals and Communication Technology

More information about this series at http://www.springer.com/series/4748

Seyed Eman Mahmoodi • Koduvayur Subbalakshmi
R. N. Uma

Spectrum-Aware Mobile Computing

Convergence of Cloud Computing and Cognitive Networking

Springer

Seyed Eman Mahmoodi
Department of Research and Innovation
Interactions Corporation
New York, NY, USA

R. N. Uma
Department of Mathematics and Physics
North Carolina Central University
Durham, NC, USA

Koduvayur Subbalakshmi
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ, USA

ISSN 1860-4862 ISSN 1860-4870 (electronic)
Signals and Communication Technology
ISBN 978-3-030-02410-9 ISBN 978-3-030-02411-6 (eBook)
https://doi.org/10.1007/978-3-030-02411-6

Library of Congress Control Number: 2018959459

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The recent advances in smartphone technologies have culminated in a surge in smart applications and use cases for these applications across the spectrum from government to gaming industry. This has placed a significant burden on the computation and other related resources (like memory) on the mobile devices. In order to serve nextgen application in this resource-constrained environment, it has become necessary to judiciously offload some of the necessary computation to the edge, fog, and cloud resources, optimally trading off all relevant parameters.

This book presents a new solution called spectrum-aware cognitive mobile computing, which uses dynamic spectrum access and management concepts from wireless networking to offer overall optimized computation offloading and scheduling solutions that achieve optimal trade-offs between the mobile device and wireless resources. This solution uses the latest developments in radio access technologies to achieve this goal.

This book shows that it is essential to consider mobile offloading holistically, from end to end, and use the power of multi-radio access technologies that have been recently developed. The future of mobile computing lies in a holistic approach to spectrum management and cloud offloading. Technologies covered in this book have applications to mobile computing, edge computing, fog computing, vehicular communications, mobile healthcare, and mobile application developments such as augmented reality and virtual reality.

This book can serve as a supplementary reading resource for graduate students, researchers, and practitioners interested in mobile computing and offloading.

New York, NY, USA — Seyed Eman Mahmoodi
Hoboken, NJ, USA — Koduvayur Subbalakshmi
Durham, NC, USA — R. N. Uma

Contents

About the Authors

Seyed Eman Mahmoodi (https://sites.google.com/site/emanmahmoodi1/) is currently a senior inventive scientist at the Research and Innovation Department of Interactions Corporation in New York, NY. His research interests include artificial intelligence and machine learning for speech and language technology specifically in multi-modal dialog systems; and optimization of mobile cloud computing systems with the applications in cognitive networks, mobile edge computing (MEC), and Internet of Things (IoT). Mahmoodi received his PhD degree at the Department of Electrical and Computer Engineering, Stevens Institute of Technology in 2017. He is the recipient of the inventor award from the New Jersey Inventors Hall of Fame (NJIHoF) in 2016. He was also awarded a 5-year Innovation and Entrepreneurship Doctoral Fellowship from the Stevens, and he received the outstanding doctoral dissertation award from the ECE Department in 2017.

Koduvayur Subbalakshmi (http://www.kpsuba.com) is the Founding Director of the Stevens Institute for Artificial Intelligence and a Professor in the Department of Electrical and Computer Engineering at Stevens Institute of Technology. She is also the Co-founder of two technology start-up companies.

Her research interests are in artificial intelligence and machine learning and applications, cognitive radio networking and security, cognitive mobile and edge computing, social media and Internet data mining, analytics and security.

Suba was named a Jefferson Science Fellow in 2016 by the National Academy of Sciences, Engineering and Medicine. As a JSF, she served as a senior science and technology adviser to the US Department of State and worked on technology policy issues in information and communications technologies like IoT and 5G communications as well as artificial intelligence and ML. She is a Founding Associate Editor of the *IEEE Transactions on Cognitive Communications and Networking*. She is the Founding Chair of the Special Interest Group on Security in IEEE COMSOC's Technical Committee on Cognitive Networks. She is a recipient of the New Jersey Inventors Hall of Fame Innovator Award. She has given several tutorials, keynote addresses, and participated in several panel discussions in IEEE and other international conferences and events. Her research is supported by NSF, NIJ, AFRL, US ISSO, industry, and other DoD agencies.

R. N. Uma is a Professor of Computer Science in the Department of Mathematics and Physics at NC Central University, Durham, NC, USA. Her research interests include data science, scheduling, and resource allocation with applications to cloud computing, robotics, wireless sensor networks, multimedia networking, and large logistics problems. Her research has spanned from purely theoretical to experimental and simulations. She received her BSc degree in Mathematics from the University of Madras, Chennai, India, her ME degree in Computer Science from the Indian Institute of Science, Bangalore, India, and her PhD degree in Computer Science from NYU Tandon School of Engineering, New York. She is a senior member of IEEE.

List of Figures

List of Tables

Chapter 1
Introduction

Advances in mobile networks, web technologies, and popularity of smartphones have led to the unprecedented growth of wireless data traffic. In the year 2012 alone mobile web traffic increased by 70% and is expected to grow up to 13 times by 2017. Computationally heavy applications which also produce large data, like real-time visual information reporting, video-intensive games, and computer vision-based applications, are now available on resource-constrained mobile devices [11]. For example, user-generated photographs and videos as well as geo-location enabled applications are being developed to increase the efficiency and speed of relief efforts in natural and man-made disasters. After the Boston bombings, Reddit had a section dedicated to helping the investigation by soliciting videos and images and USTREAM saw more than 130 K people listening to the Boston police scanner simultaneously at one point on their mobile devices.

This uptick in sophisticated mobile applications has not only increased the computational demand on the end device itself, but has also created an increasing load on the carrier's core networks. Add this to the projected growth in mobile customers due to the advent of fifth generation (5G) mobile communications with its promises of lower battery consumption, reduced traffic latency (unlike LTE), enhanced reliability (like software-defined services) as well as perfect coverage and the ability to support bandwidth hungry applications such as HD video streaming, and the pressure on the core networks will be enormous. Further demands on the core networks and the radio spectrum, in general, will be placed as the Internet of Things (IoT) matures. Gartner Inc. estimates that a total of about 8.4B "things" will be connected worldwide by the end of 2017.[1] To compound matters even further, the current mobile development trends indicate a shift towards cheaper phones with fully functional OSes and applications. This implies that mobile devices

[1] http://www.gartner.com/newsroom/id/3598917.

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_1

will be expected to "do more with less" in terms of delivering highly sophisticated applications under more stringent battery power constraints as well as runtime deadlines.

One of the solutions to deal with the resource crunch at the mobile device is to offload heavier computation to resource rich clouds [16]. Services like the Amazon web services (AWS)[2] have made the power of cloud computing accessible to the average user. The need for smarter computations with low latency has also fed the growth of mobile edge computing (MEC), which brings the power of resource rich computation closer to the user [43].

This book makes a distinction between computation offloading [10, 12, 21, 32] and data offloading [6, 13, 25, 26]. Computation offloading refers to actually completing part of the computation on a remote resource rich server and transferring the data related to these computations between entities as needed.

Although offloading to a remote cloud can address the resource crunch at the end device, it can also add to the aforementioned burden on the wireless backbone [24]. In order to develop workable solutions, it is essential to consider both problems simultaneously. The network level problem can be addressed by using multiple radio interface technology (multi-RAT) which is now becoming a mainstay of 5G wireless systems. The heterogeneous network (HetNets) paradigm, enabling multi-RATs is expected to become a mainstay of future wireless networks. Simultaneous access to multiple RATs can be implemented at the transport layer, network layer, or PHY/MAC layer of wireless devices. The growth of mobile virtual network operators (MVNO) will also facilitate such multi-RAT opportunistic spectrum access. Google's recent deal with Sprint and T-Mobile is an example in this direction; LTE-WiFi aggregation is another. Under the circumstances, spectrum aware mobile computation offloading will fare better than solutions that merely focus on traditional performance metrics like energy, bandwidth (within a single network).

1.1 Factors Affecting Computation Offloading

Several factors affect computation offloading, the most obvious one being the local resources in the mobile device like battery power, CPU cycles, and memory usage.

Although intuitively, offloading all of the computations to a resource rich cloud, a distributed cloud or a smart access point, may sound like the most cost-effective solution in terms of conserving resources, several studies have shown that partial offloading is better in terms of a net utility function that not only minimizes local resource consumption such as energy, memory, and CPU cycles, but trades-off inter-component communication costs arising from executing some components locally and some remotely [34].

[2] http://aws.amazon.com/workspaces/.

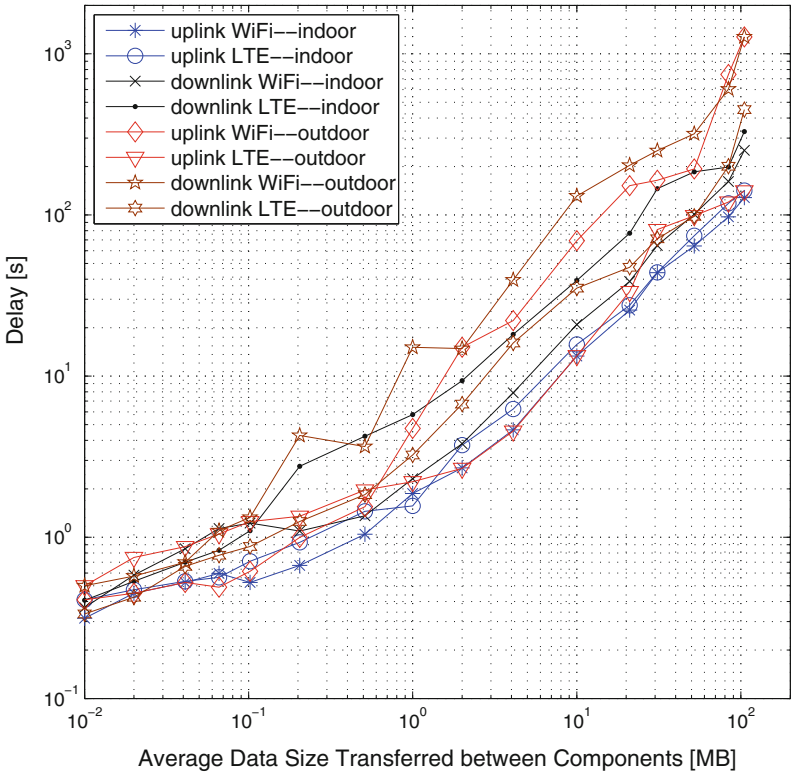


Fig. 1.1 Variation in Uplink/Downlink delay of WiFi and LTE interfaces in indoor and outdoor wireless environments versus average size of data transferred between HTC smartphone and the NSFCloud server in order to run the mobile video navigation application

The second factor affecting cloud offloading is the delay and difference in delay between the available RATs. Several experiments were conducted to measure the end-to-end quality and availability of T-Mobile LTE in Hoboken, New Jersey, and Stevens campus WiFi network to measure delay parameters in indoor and outdoor environments. The experimental setup consisted of a client HTC Vivid smartphone with a 1.2 GHz dual-core processor accessing NSFCloud[3] as a remote server. Experiments were conducted by running a video navigation multi-component application (14 components) on the smartphone in a client configuration. The NSFCloud server was set up in a server configuration for all the experiments. Uplink and downlink traffic were captured at the client side using the Android SFTP tool[4] for experiments.

Figure 1.1 shows the variations in the observed uplink and downlink delay values in indoor and outdoor environments for WiFi and LTE wireless radios for a range

[3] www.chameleoncloud.org/nsf-cloud-workshop/.

[4] http://www.lysesoft.com/.

of data size from 10 KB to 105 MB of data. As can be seen from Fig. 1.1, delay in outdoor wireless can be smaller than delay in indoor wireless environment for some ranges although indoor wireless delays are lower on an average. Such high random delays are unacceptable for computation offloading.

The third factor that will affect offloading decisions is the amount of data that must be transferred to and from the mobile devices, between the elements of the computations at runtime. Consider a video navigation application with several components. These will typically include: video capture at the camera, preview processing, graphical tasks, color detection, feature extraction, Canny edge detection, face recognition, and clustering.

Clearly the data input and output to each of these components can be significant. For example, output data size of four different application components showed the following values for output data: 12 MBs for text to speech recognition with runtime of 120 s; 104 MBs for augmented reality (AR) with runtime of 25 s; 309 MBs for 3D flipping with runtime of 40 s; and 598 MBs for face detection with runtime of 60 s. Examples of such applications include 3D interactive mobile video gaming [5], augmented reality like SpyGlass project with the provision of a visualization framework for wireless sensor networks (WSNs) [4, 17], and the applications related to disaster preparedness and recovery supported by federal agencies including FEMA.[5]

Since cost of offloading different data rates will be different and will also affect the total application runtime because of communication delays, data rates also must be factored when computing the optimal offloading policy.

Therefore, in order to meet the constraints on the mobile devices, the wireless spectrum/backbone, and the application demands, it is essential to think about mobile cloud offloading holistically, in a way that includes all of these factors. **We coin the phrase "spectrum aware mobile computational offloading" to describe this category of ideas.**

In this book we will first discuss the existing mobile cloud computing paradigms and then present a vision for the future that will effectively use the power of 5G cognitive radio networking to assist in mobile cloud computing.

1.2 Organization of the Book

The rest of this book is organized as follows. Chapter 2 discusses the related work on cloud offloading mechanisms and sets the new paradigm of spectrum aware mobile computing in context of this evolution.

In Chap. 3, we look at applications with arbitrary dependency graphs and discuss joint scheduling–offloading schemes for single radio enabled mobile devices that optimally maximize a net utility function. The net utility function trades-off the

[5] Federal Emergency Management Agency.

energy saved at the resource-constrained device with the time and energy costs involved in offloading while meeting the precedence constraints and execution deadline of the application. This approach targets joint scheduling–offloading for mobile applications. By optimizing the scheduling of the individual components along with cloud offloading decisions, taking into account the wireless parameters, allows for an overall better solution compared to optimizing only the offloading decisions using a pre-determined compiler-generated schedule order of execution for the individual components. Besides, using the general dependency graphs (without imposing a sequential ordering for processing) and an optimal joint scheduling–offloading scheme can potentially allow for parallel scheduling of components in the mobile and cloud at the same time, thus reducing time to completion for the application.

While computation offloading to a resource strong cloud seems like the natural solution to the resource crunch at the mobile device level, it is essential to take into account the associated data transfer that must take place between the components that are executed in the cloud and their counterparts in the mobile device. Given the already increasing demands on the wireless backbone caused by the promise of 5G networking, this means that computation offloading must be viewed in the context of the already increasing mobile traffic. Hence it would be prudent to *optimally* use *all* of the radio interfaces (like WiFi, 3G, HSPA, and LTE), as appropriate, that are available in the multi-radio equipped mobile devices of today.

In Chap. 4, we address a solution that optimally decides which components of an application to offload and which to execute locally, while simultaneously optimizing the percentage of data (associated with this offloading) to be sent via each radio interface. Given recent advances in technologies that enable bandwidth aggregation in wireless devices [18, 20] this solution is implementable in practice. Other works that fall under general umbrella of the radio-aware computation offloading include [19], where the best of the available wireless interfaces is chosen (only one of the wireless interfaces) for data transfer, rather than a solution that considers using all of the radio interfaces simultaneously. In [2] a cloud offloading scheduling mechanism is proposed for queue stability, but this work only deals with multi-channel systems, not multi-radio networks. Etime [45] is an "everything on the cloud" offloading strategy, which adapts to the condition of the wireless link, but this work does not consider multiple interfaces. In this chapter, we address a comprehensive model for the energy consumed by the mobile device, including energy expended in communicating relevant data between the cloud and the device. The computation offloading problem is set up as a joint optimization to minimize the energy consumed on the device while at the same time maximizing the radio resources available to the device, under two constraints: (1) the total runtime deadline of the application and (2) the maximum flow rate constraint on the radio resources. Since this optimization problem is non-linear and hence computationally intense, we also propose an iterative algorithm that converges to a local optimum.

Chapter 5 addresses a general approach for optimal cognitive scheduling and cloud offloading using multiple radios. In this chapter, we move to a more realistic extension of the problem, in three ways: (1) we consider natural scheduling order for

more general dependencies between the components of the application (see Sect. 2.2 for more on component dependency graphs), (2) a cognitive cloud offloader is used where all multiple radio interfaces are used for cloud offloading, (3) we note a time adaptive approach that varies with the changes in the wireless network conditions over time.

In Chap. 6, more practical heuristic time adaptive schemes are introduced to schedule the components for offloading, while simultaneously optimizing the percentage of data to be sent by the mobile and the cloud via each wireless interface. A comprehensive model for the utility function is described that trades-off resources saved by remote execution (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). The solution can be implemented in two ways: (1) 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 (2) 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 end use past wireless interface data, queue status, and the current data flow to update the current queue status.

Chapter 7 discusses the performance of all schemes described in this book. The performance of the algorithms is compared with different approaches including (1) local execution (no offloading); (2) complete offloading (all components remotely executed); (3) the non-time adaptive dynamic offloading algorithm proposed in [19] extended to applications with sequential dependency graphs; and (4) the approach where offloading takes place only via the best link at each instant of time.

Finally, Chap. 8 discusses the future of cognitive scheduling and cloud offloading in 5G networks.

Chapter 2
Classification of Mobile Cloud Offloading

Work on mobile computing fall within two broad categories: (1) those that offer platforms for implementing mobile cloud offloading solutions and (2) those that concern themselves only with devising optimal scheduling methods that will provide the best trade-offs of some of the resources involved. Offloading mechanisms can also be seen from four perspectives: (1) extent of offloading; (2) application element interdependency awareness; (3) wireless network awareness; (4) use (or not) of multiple radio access technologies [27]. A bird's eye view of the classification of mobile computation offloading based on these four mechanisms is shown in Fig. 2.1.

These categories are explained in detail in the following subsections.

2.1 Extent of Offloading

Computational offloading methods can be classified broadly into three categories: (1) those that either offload the entire application or execute it entirely locally (all-or-nothing offloading) [50]; (2) those that offload everything to the cloud (wholesale offloading) [8, 16, 42, 45], and (3) those that split the application into smaller parts and make a decision to keep or offload each part [2].

In wholesale offloading, data migration can be implemented using distributed application processing. For wholesale offloading, where the entire application is offloaded, the main problem becomes identifying good network opportunities to complete the offloading. For example, *eTime* explores the energy-delay trade-off in scheduling the required data transmissions for offloading. If a greedy approach is adopted, where the data transmissions are deferred until perfect network conditions are favorable (to ensure aggressive energy savings), the queue backlog of all applications will increase unboundedly [14], consequently leading to large unacceptable delays and poor user experiences. eTime proposes an optimization framework to deal with this trade-off.

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_2

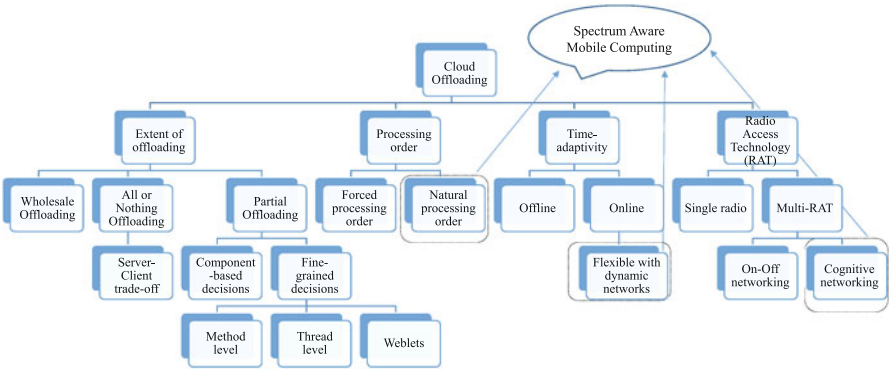


Fig. 2.1 Classification of mobile cloud offloading

In all-or-nothing offloading, the quality of wireless connectivity is the determining factor. In [50], the decision of whether to offload the application or not is based on wireless connectivity, using a threshold. In particular, for a given application profile $A(L, T)$, where L is the data size for the entire application and T is the runtime requirement, the minimum computation energy for mobile execution and the minimum transmission energy for cloud execution are computed. The optimal application execution policy is to choose whichever costs less energy for the mobile device.

Partial offloading is more efficient in limited spectrum resource wireless networks in terms of delay, communication energy, and data queue backlog [2, 10, 21, 22, 35, 44, 51]. Within the partial offloading strategies, some approaches have proposed coarse level partitioning of the applications where the code is pre-partitioned into components [2, 19, 22, 44]. An Android specific services-based mobile cloud computing middleware called mobile augmentation cloud services (MACS) [22] 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 battery power on devices, and bandwidth of the network(s). The approaches can be either static, where the offloading strategy is decided before executing the application, or dynamic where the offloading strategy is made during the application execution, thus adapting to the changing wireless rates. A more fine-grained offloading can be achieved by using method-level partitioning as is done in ThinkAir [21]. ThinkAir also provides method-level, semi-automatic offloading of code, but focuses more on scalability issues and parallel execution of offloaded tasks. CloneCloud [8] provides a flexible code partitioner that enables unmodified mobile applications running in an application-level virtual machine to seamlessly offload parts of their execution from mobile devices to device clones. This allows for an automated, fine-grained offloading mechanism at the thread level. Remote processing by weblets is another type in this category. Here the execution destination for the weblet is determined dynamically at execution time.

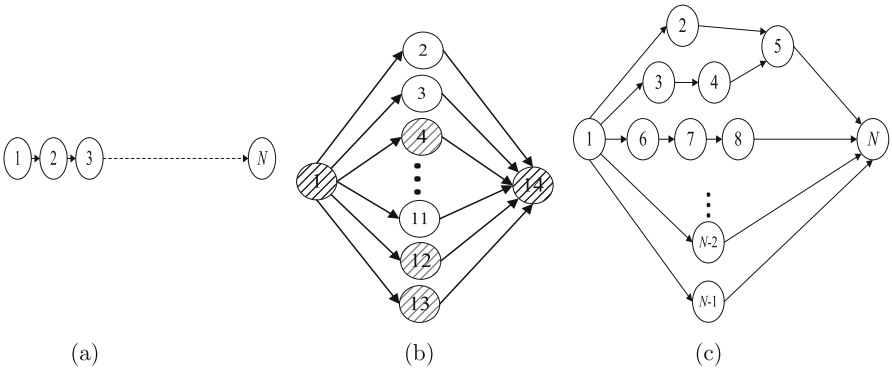


Fig. 2.2 Topologies of component dependency graphs. (**a**) Sequential. (**b**) Parallel. (**c**) General

2.2 Processing Order

As mentioned in the previous section, applications can be split in different ways. One of these is to split the application into components [2, 19, 22, 44, 52]. These components interact with each other in terms of order of execution as well as necessary data transfer between these components. This dependency is captured by the so-called component dependency graph (CDG). Figure 2.2 shows different topologies for CDGs of an N-component application. Component i is dependent on component j, if the output data from j is required to execute i. Component dependencies could be: (1) sequential; (2) parallel where all components depend on only the first component, and the last component depends on all the rest; or (3) series-parallel (general dependency).

The majority of scheduling mechanisms are based on either a pre-determined, sequential schedule order of the tasks. Example of wholesale offloading of components in a compiler generated CDG can be found in [1]. However, a wireless aware scheduling order, that is cognizant of the structure of the component dependency graph, provides more degrees of freedom in the solution and consequently can do significantly better in terms of reducing overall cost metrics [34, 36]. A scheduling strategy for partially offloading the sequence of fine-grained tasks with serial CDG is also proposed in [52].

2.3 Time Adaptivity

It is also possible to classify offloading mechanisms according to whether offload decisions are made in a time-adaptive manner (time-adaptive strategies), thereby tracking the current wireless conditions or as a single shot solution computed just before the application must be executed (non-time adaptive). A partial non-time

adaptive (offline) offloading policy for component level offloading, for the special case when the order of execution of components is pre-determined, appears in [52].

Time-adaptive computation offloading allows the offloader to make decisions based on instantaneous variations of 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 [47] where the server estimates the response time for remote execution of each task based on total bandwidth server model, and the tasks are scheduled for offloading with "earliest deadline first" algorithm.

2.4 Radio Access Technology (RAT)

Early mobile devices were equipped with only one radio interface and hence offloading was restricted to that one interface. Current mobile devices come equipped with multiple of radio access technology (RAT) interfaces, like WiFi and cellular. One of the key developments in wireless networking is the introduction of multiple radio access technologies (multi-RAT) [25, 29, 31, 46], which enable mobile devices to access multiple radio networks (LTE, WiFi) simultaneously to increase network throughput. Incorporating this technology should increase the performance of the offloading protocol. We can classify the existing offload mechanisms as (1) single-RAT offloaders [21, 22], where the schedulers assume that there is only one RAT in the mobile device; (2) On/Off multi-RAT offloader, where although there are multiple RATs, the scheduler only picks one of the interfaces, the "best" interface, for offloading based on current conditions of the individual interfaces [19], and (3) cognitive [54] multi-RAT offloaders, where all available RAT interfaces are used with appropriately allocated percentage of data traffic through each of these radio interfaces to optimize all relevant parameters such as energy consumption, delay, and execution time [34].

Several methods have been developed for optimal offloading over single-RAT devices. In [2], a model for the partial offline offloading in single-*RAT* enabled mobile devices is studied that meets queue stability constraints in multi-*channel* scenario. Note, however, that this method schedules offloading over multiple *channels* within the same band and not over multiple wireless *networks* (or multi-RATs). As pointed out by examples in Chap. 1, wireless networks (e.g., WiFi and LTE) differ significantly in their characteristics from one another and therefore the techniques associated with multi-RAT scheduling are different than those used in multi-channel scheduling.

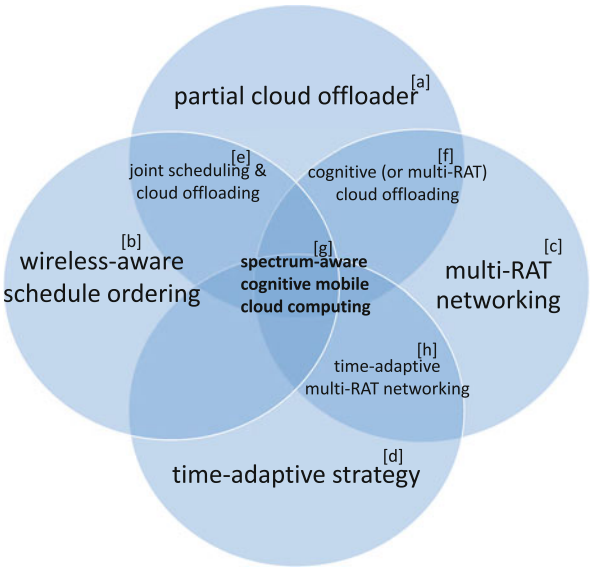


Fig. 2.3 Venn diagram depicting various categories of spectrum aware mobile cloud computing schemes. Here (**a**) refers to works presented in [2, 10, 21, 22, 44, 51]; (**b**) refers to [28]; (**c**) refers to [6]; (**d**) refers to [12]; (**e**) refers to [36]; (**f**) refers to [19, 35]; (**g**) refers to [34, 37], and (**h**) refers to [45]

The relationship between the aforementioned perspectives for cloud offloading strategies is schematically shown in Fig. 2.3. In this Venn diagram, each bubble shows a perspective of computational cloud offloading. Specific schemes are illustrated mapped with the corresponding bubbles while there might be approaches that address more than one perspective in cloud offloading (like the categories in [e,f,g,h]). Finally the works in spectrum-aware cognitive mobile cloud computing cover all the four perspectives.

Chapter 3
Joint Scheduling and Cloud Offloading Using Single Radio

We now explore the effects of doing away with a compiler pre-determined scheduling order on the overall optimality of the offloading solutions. To this end, this chapter introduces the concept of wireless aware *joint* scheduling and computation offloading (JSCO) for multi-component applications. The problem is to find an optimal decision on which components need to be offloaded and in what scheduling order. We do not presume a sequential pre-determined order of scheduling for these components. Hence the method discussed in this chapter falls under the category [e] shown in Venn diagram of Fig. 2.3. The JSCO approach allows for more degrees of freedom in the solution by moving away from a compiler pre-determined scheduling order for the components towards a more wireless aware scheduling order. For some component dependency graph structure, this method can shorten execution times by parallel processing appropriate components in the mobile and cloud. A net utility is defined that trades-off the energy saved by the mobile, subject to constraints on the communication delay, overall application execution time, and component precedence ordering. The linear optimization problem is solved using real data measurements obtained from running multi-component applications on an HTC smartphone and the Amazon EC2, using WiFi for cloud offloading. The performance is further analyzed using various component dependency graph topologies and sizes.

3.1 Scheduling Model for Mobile Cloud Offloading

When consideringmobile cloud offloading model, the mobile device has access to a cloud server for computation offloading, and the cloud server is equipped with parallel processing capabilities. We have the following assumptions: (1) the multi-component mobile application is also installed on the cloud server; and (2) mobile broadband connectivity does not change during the application processing time (T)

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_3

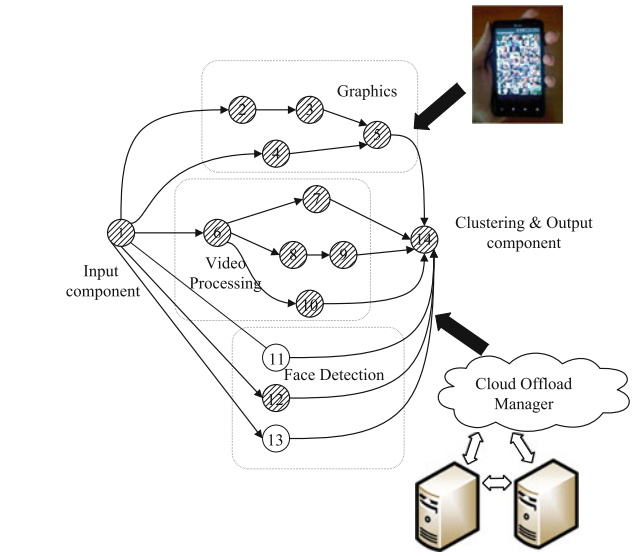


Fig. 3.1 Scheduling model for cloud offloading in a 14-component mobile application with a general CDG. (**a**) CDG of the application. (**b**) Scheduling–offloading model

while the wireless interface may provide different rate and delay values. Single-RAT only optimizations are sufficient where the application processing times are not very large. Examples of such solutions for different cases can be found in [2, 10, 19, 21, 22, 39]. Following these assumptions, a mobile cloud offloading model example of a 14-component application is shown in Fig. 3.1b.

3.2 Multi-Component Application Example for Scheduling–Offloading Model

Consider a video navigation application involving graphics (http://www.opengl.org/, March 2014), face detection (http://www.developer.com/ws/android/programming/face-detection-with-android-apis.html, July 2014), camera preview, and video processing (http://opencv.org/, April 2014), running on an HTC Vivid smartphone. Figure 3.1a shows the dependency graph of this 14-component application. The unidirectional edge from component i to j shows that the output data from component i is required as input by component j, and d_{ij} represents the required data size for transferring from i to j. This dependency could be either sequential (like the dependencies between components 1-2-3-5-14) or parallel (like the conditions of component dependencies between 1-11-14, 1-12-14, and 1-13-14). As per definitions of CDGs in Chap. 2, this is a series-parallel CDG of the most general CDG (like Fig. 2.2c).

In Fig. 3.1b, an example of joint scheduling–offloading of the components based on time, place of processing, and dependency among the components is illustrated. If a component is scheduled for offloading to the cloud, the energy consumption for processing will be saved by remote execution. In addition, the time for processing the component decreases significantly by remote execution (compare the time taken to process components by the mobile device and the cloud in Fig. 3.1b). Moreover, some components can be processed in parallel by the cloud (components 2, 6 and components 3, 10). However, the cost of cloud offloading should also be considered in the scheduling–offloading decisions: (1) the costs of delay and energy consumed by offloading as a function of data size for transferring (e.g., component 11 has very large data for transferring so it takes a longer time for communication); and (2) the cost of the idle state as the mobile waits to receive the required output data from the cloud (between components 4 and 7).

Thus a *smart* scheduling strategy for mobile offloading based on *energy-time* trade-off is required (Table 3.1).

Table 3.1 Parameter definitions for the JSCO problem

Parameters	Definitions
N	Number of components in the application
T	Number of time periods to complete processing the application
t	Time index for period (t-1,t]
m_j	Mobile execution indicator for component j
c_j	Cloud execution indicator for component j
x_{pjt}	A binary indicator which equals to 1 if component j completes processing at time t on processing system p and otherwise equals to 0
μ_{ij}	Dependency indicator: 1 if component i must be processed before j and 0 otherwise
z_{ij}	Component transferring indicator, which equals to $m_i c_j$
d_{ij}	Size of data required by component j from component i
q_j^{m} (q_j^{c})	Time to process component j in the mobile (cloud)
τ_{ij}^{mc}	Time required to transmit data from component i executing in the mobile to component j executing in the cloud
τ_{ij}^{cm}	Time required to receive data from component i executing in the cloud to component j executing in the mobile
v_k	Time to process component k either on mobile or cloud
E_{com}	The total energy consumed by the mobile device for communication
P_{ac}	Active power of the mobile while processing a component
P_{Tx} (P_{Rx})	Power consumption of the mobile to transmit (receive) required data
R_{u} (R_{d})	Average uplink (downlink) rate of the wireless radio interface

3.3 Optimal Joint Scheduling and Computation Offloading Scheme (JSCO)

In this section, the formulation of the optimization problem is presented as an integer linear program. For each time period $(t-1, t]$ denoted by t, decision variable, x_{pjt}, indicates whether component j completes processing at time t on the mobile ($p = 0$) or on the cloud ($p = 1$). This decision variable captures the multi-objective requirement of mobile communication applications to provide "*anywhere, anything, anytime*" service. The processing indicators in the mobile and cloud are respectively given by

$$m_j = \sum_{t=1}^{T} x_{0jt} \qquad \forall j \tag{3.1}$$

$$c_j = \sum_{t=1}^{T} x_{1jt} \qquad \forall j. \tag{3.2}$$

Also τ_{ij}^{cm} denotes the time to transfer data from component i to j when $i \prec j$, and j is processed on the mobile and i is processed on the cloud. τ_{ij}^{cm} includes the product $m_j c_i$ where i is processed on the cloud and j is processed on the mobile. In order to make the optimization problem linear, this quadratic term of decision variables is replaced by a new variable z_{ji} where z_{ji} must satisfy the following four constraints (Rubin, http://orinanobworld.blogspot.de/2010/10/binary-variables-and-quadratic-terms.html, Sept 2014):

$$z_{ji} \leq m_j \qquad \forall j, i \tag{3.3}$$

$$z_{ji} \geq 0 \qquad \forall j, i \tag{3.4}$$

$$z_{ji} \leq c_i \qquad \forall j, i \tag{3.5}$$

$$z_{ji} \geq c_i - (1 - m_j) \qquad \forall j, i. \tag{3.6}$$

Thus, the quadratic term of two decision variables is converted to a new decision variable so that the optimization problem still remains linear. Similarly, τ_{ij}^{mc} denotes the time to transfer data from i to j when i is processed on the mobile device and j is processed on the cloud and includes $m_i c_j$ which is denoted by the variable z_{ij}. Now the times for transferring from mobile to cloud and cloud to mobile are respectively given as

$$\tau_{ij}^{\text{cm}} = \mu_{ij} z_{ji} \frac{d_{ij}}{R_{\text{d}}} \qquad \forall i, j \tag{3.7}$$

$$\tau_{ij}^{\text{mc}} = \mu_{ij} z_{ij} \frac{d_{ij}}{R_{\text{u}}} \qquad \forall i, j. \tag{3.8}$$

Note that τ_{ij}^{cm}, τ_{ij}^{mc} will be zero if $i = j$, or if i does not precede j, or if i and j are both processed on the cloud, or both processed on the mobile device. In addition, the energy consumed for communication due to cloud offloading the components is modeled by

$$E_{\text{com}} = P_{\text{Tx}} \sum_{i=1}^{N} \sum_{j=1}^{N} \tau_{ij}^{\text{mc}} + P_{\text{Rx}} \sum_{i=1}^{N} \sum_{j=1}^{N} \tau_{ij}^{\text{cm}}. \tag{3.9}$$

The objective function in the optimization problem over decision variables (x_{pjt}, z_{ij}, $p \in \{0, 1\}$, $i, j = 1, \ldots, N$, $t = 1, \ldots, T$) for the mobile cloud offloading scheme is mathematically formulated as

$$\max \left\{ \sum_{j=1}^{N} P_{\text{ac}} c_j q_j^{\text{m}} - E_{\text{com}} \right\}. \tag{3.10}$$

Equation (3.10) shows the maximization of the energy saved through remote execution. This energy saved is essentially the energy cost if the offloaded components had been executed locally minus the cost of communication energy.

Besides constraints in (3.3)–(3.8) the following constraints should be satisfied in the optimization problem with the objective function given by Eq. (3.10):

Runtime Deadline Constraint The multi-component application has a time deadline (T), which should be satisfied. This constraint is given by

$$0 < \sum_{t=1}^{T} t x_{0Nt} \leq T \qquad \forall t, \tag{3.11}$$

where $\sum_{t=1}^{T} t \cdot x_{0Nt}$ denotes the completion time of processing the last component (N) on the mobile ($p = 0$). This time should be equal or less than the runtime deadline of the application.

Each Component be Processed Only Once Each component is processed either in the mobile or cloud, which can be written as

$$m_j + c_j = 1 \qquad \forall j. \tag{3.12}$$

Precedence Constraint This constraint shows that component k is required to begin processing no earlier than the completion time of component j where $j \prec k$. The constraint is expressed as

$$\begin{aligned} &\sum_{p=0}^{1} \sum_{s=1}^{t+\nu_k+\tau_{jk}^{\mathrm{cm}}+\tau_{jk}^{\mathrm{mc}}} x_{pks} \leq \sum_{p=0}^{1} \sum_{s=1}^{t} x_{pjs}, \\ &\text{if } j \prec k, t = \nu_j, \ldots, T - \nu_k - \tau_{jk}^{\mathrm{cm}} - \tau_{jk}^{\mathrm{mc}}, \end{aligned} \tag{3.13}$$

where ν_k is the time to process component k either on the mobile or cloud, and is given by

$$\nu_k = m_k q_k^{\mathrm{m}} + c_k q_k^{\mathrm{c}}. \tag{3.14}$$

Based on Eq. (3.12), ν_k will include either the cloud processing time for component k or the mobile processing time for component k, but not both. Here in constraint (3.13), in order for k to be completed after the time t plus the time for possible data transferring from j to k ($\tau_{jk}^{\mathrm{cm}} + \tau_{jk}^{\mathrm{mc}}$), plus the time for processing component k (ν_k), component j must be completed by time t, $\forall t$.

Serial Computation at the Mobile Device The processed components in the mobile are required to be executed in serial. Thus, for each time interval $[t-1, t)$ we can have at most one component for processing in the mobile, which can be written as

$$\sum_{j=1}^{N} \sum_{s=t}^{\min\{t+\nu_j-1,T\}} x_{0js} \leq 1 \qquad \forall t. \tag{3.15}$$

Completion Deadline Each component k must be completed only after the completion of each of its precedent components like j, plus the time to process component k itself, and the time to transfer required data to the execution site of k if j is not on that same site. This constraint is given by

$$\begin{aligned} &\sum_{p=0}^{1} \sum_{t=1}^{T} t x_{pjt} + \tau_{jk}^{\text{cm}} + \tau_{jk}^{\text{mc}} + \nu_k \\ &\leq \sum_{p=0}^{1} \sum_{t=1}^{T} t x_{pkt}, \text{ if } j \prec k, k = 1, \ldots, N. \end{aligned} \tag{3.16}$$

Also, decision variables should be $0-1$,

$$x_{pjt} \in \{0, 1\} \qquad p \in \{0, 1\}, \forall j, t, \tag{3.17}$$

and get zero values while the coordinated component has not been processed yet, which is written as

$$x_{pjt} = 0 \qquad p \in \{0, 1\}, \forall j, t = 1, \ldots, \nu_j - 1. \tag{3.18}$$

3.4 Comparison with the State of the Art

The linear optimization problem is solved using real data measurements obtained from running multi-component applications on an HTC smartphone and the Amazon EC2, using WiFi for cloud offloading. The performance is further analyzed using various component dependency graph topologies and sizes. Results show that the energy saved increases with longer application runtime deadline, higher wireless rates, and smaller offload data sizes. Note that the evaluation setup and the other simulations related to this chapter are elaborated in Sect. 7.2.

JSCO is compared to (1) no offload (local) execution where all the components are executed locally; (2) all offload (remote) execution where all the components are offloaded to the cloud; (3) the dynamic offloading algorithm (DOA) in [19],

which uses an energy efficient partial offloading strategy; (4) HELVM algorithm from [40], which provides runtime offloading services; and (5) a heuristic algorithm that is the revised HEFT [48] for joint scheduling (RHJS) tasks on multiple cores used in [28]. In the simulations for this subsection, a face recognition application with ten sequential components was utilized (http://darnok.org/programming/face-recognition/). The wireless network parameters in http://www.3gpp.org/ftp/tsg-ran/wg4-radio/ are used such that exactly the same parameters used for the simulation of DOA in [19] were used for all the other schemes.

In Fig. 3.2, the total energy consumption of the JSCO scheme is compared with the five schemes. This comparison is normalized to the scheme with local execution of all the components. It is observed that JSCO consumes 54%, 37%, 16%, 30%, and 11% less energy in comparison to the schemes using local execution, remote execution, DOA, HELVM, and RHJS, respectively.

Figure 3.3 shows the time to run the application (http://darnok.org/programming/face-recognition/) for the six schemes. This comparison is also normalized to the scheme with local execution of all the components. By using the optimal JSCO scheme, the application will be executed 25%, 49%, 32%, 19%, and 5% faster in comparison to the schemes using local execution, remote execution, DOA, HELVM, and RHJS, respectively. Thus, JSCO is a joint energy and time efficient scheme in comparison to the other five schemes.

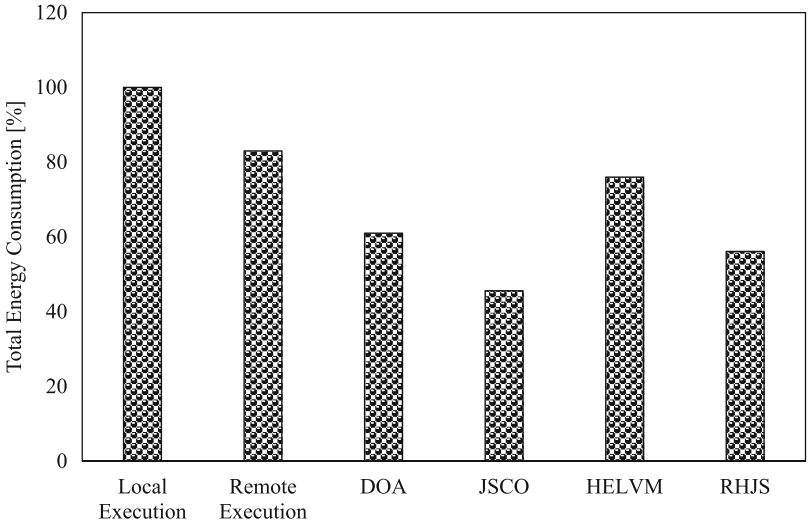


Fig. 3.2 Total energy consumed by the mobile device for different schemes, normalized to the energy consumed by local execution (using the face recognition application in http://darnok.org/programming/face-recognition/)

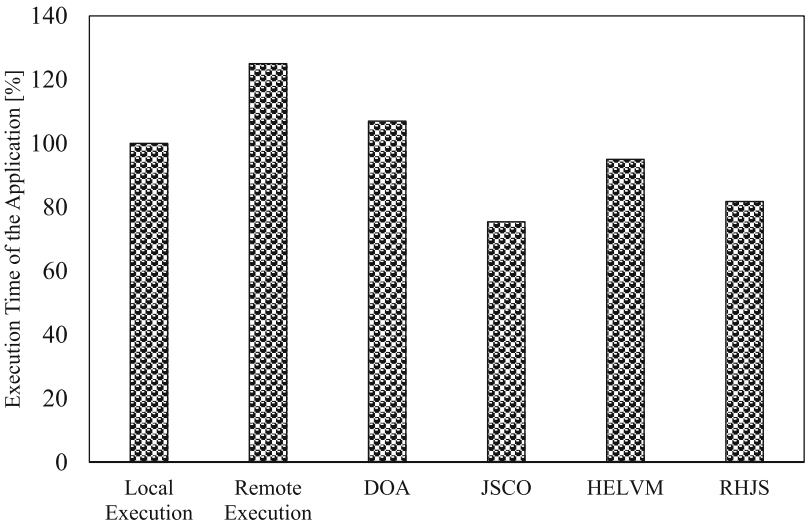


Fig. 3.3 Total execution time of the application for different schemes, normalized to the execution time by local execution (using the face recognition application in http://darnok.org/programming/face-recognition/)

Chapter 4
Cognitive Cloud Offloading Using Multiple Radios

There are two ways in which multiple RATs can be used in offloading: one is to use the best interface at any given time for all computation offloading related data transfers and the other is to use *all* viable interfaces. We refer to the latter as "cognitive cloud offloading." In the Venn diagram shown in Fig. 2.3, both ways of using multiple RATs are covered by category [f]. We now explore the effects of using *cognitive cloud* offloading using multiple RATs simultaneously (wherever possible) on the overall cloud offload optimization. When all available RATs are used for data transfers and there are differences in the qualities of these RATs, the solution must now accommodate for this fact by determining the appropriate amount of data that can be transferred via each of these RAT interfaces.

We define the term "cognitive cloud offloading" in multi-RAT enabled mobile devices as mobile computation transfers to a cloud with simultaneous access to all available radio interfaces of the device. In some applications, there is no capability for adapting the scheduled order of compilers based on wireless parameters, and cloud offloading must be considered based on the forced (sequential) processing order of task components. However, there are still ways to improve the system efficiency of the cloud offloader. This chapter explores this concept where there is no way to keep the flexibility in the scheduling order, but there is flexibility to use all the multiple RATs simultaneously. Simulations illustrate that this cognitive cloud offloader is more efficient than a system which uses the best radio interface. Also in the next chapters, this cognitive cloud offloader is used for the systems where there is the flexibility of using wireless aware scheduling order of component tasks.

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_4

4.1 System Model

Consider a mobile device with K radio interfaces, running computationally intense applications with N components (see Fig. 4.1, with an example where $K = 2$ and $N = 6$).

Any given component may require data from the other components to complete execution. In this example, the optimal offloading strategy stipulates that Components 1, 2, 4, and 6 be executed in the mobile device, and Components 3 and 5 be offloaded to the cloud. In Fig. 4.1, Component 3 requires d_{23} units of data from Component 2 to complete execution. In this example, 60% ($\alpha_{2,1} = 0.6$) of this data is sent through the radio interface 1 (WiFi, say), and 40% ($\alpha_{2,2} = 0.4$) through interface 2 (LTE) to give us the most performance efficient offloading strategy. Once Components 3 and 5 have finished execution, the data needed by Component 6 from Component 5 (d_{56}) must be sent to the mobile device via one of the radio interfaces (for example, in Fig. 4.1 is WiFi). We assume that only one radio interface is used for data reception $\left(\sum_{k=1}^{K} \beta_{i,k} = 1\right)$, leaving the optimization of radio resource allocation for the downlink as future work. Also, we assume that the energy consumption and the time required to transfer data within components that are executing in the same entity (whether cloud or mobile) is negligible in comparison to when the data must be transferred between entities. We also assume that the components of the application are executed in a pre-determined manner

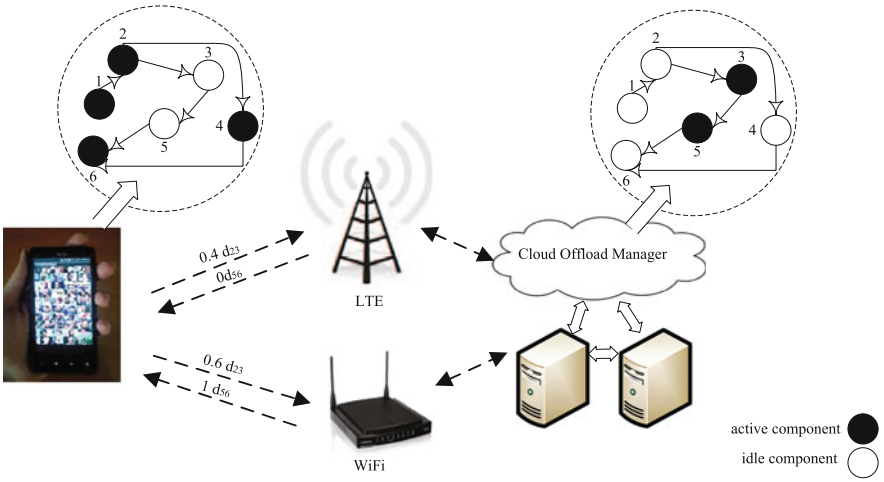


Fig. 4.1 An example of application offloading to the cloud. In this figure, the dots represent components of the application. There are six components in this application. Components 1, 2, 4, and 6 run on the device, whereas Components 3 and 5 are executed in the cloud. Two radio links are available to the mobile device for offloading components to the cloud and the diagram shows the ratio of data that is sent via each radio interface. The term active (idle) components refers to the components that are executed (or not) in that particular entity, mobile device, or the cloud

[10, 19]. These solutions are useful when the compilers pre-determined scheduling order cannot be overwritten.

4.1.1 Mathematical Modeling of the Parameters

The parameters needed to set up the optimization are described in Table 4.1. The energy consumed by the mobile device is modeled in running application component i, as $E_i = E_i^{(\mathrm{m})} + E_i^{(\mathrm{c})} + E_i^{(\mathrm{com})}$, where $E_i^{(\mathrm{m})}$, $E_i^{(\mathrm{c})}$, and $E_i^{(\mathrm{com})}$ are all defined in Table 4.1. The energy consumed to execute component i locally, in the mobile device, is expressed as $E_i^{(\mathrm{m})} = (1 - I_i) P_{\mathrm{ac}}^{(\mathrm{m})}(i) q_i^{(\mathrm{m})}$. If the component is executed remotely, then the mobile will only spend the idle power for the duration of this execution. Hence, the energy consumed by the mobile when component i

Table 4.1 Parameter definitions for cognitive cloud offloading problem

Parameters	Definitions
N	Number of components in the application
K	Number of radio interfaces in the system model
$P_{\mathrm{ac}}^{(\mathrm{m})}(i)$	Power consumed by the mobile device when it is actively processing component i
$P_{\mathrm{id}}^{(\mathrm{m})}$	Power consumed by the mobile in the idle mode
$P_k^{(\mathrm{Tx})}$ ($P_k^{(\mathrm{Rx})}$)	Transmit (received) power consumed by the mobile device at radio interface k
$q_i^{(\mathrm{m})}$ ($q_i^{(\mathrm{c})}$)	Time to process component i in mobile (cloud)
$\tau_{ij,k}^{(\mathrm{mc})}$ ($\tau_{ij,k}^{(\mathrm{cm})}$)	Time to transfer data required by component j to mobile (cloud) from component i in the cloud (mobile), using radio interface k
$T_i^{(\mathrm{com})}$	Time to transfer necessary data between the cloud and mobile, to execute component i
μ_{ij}	Component dependency indicator: 1 if component i must be processed before j, 0 otherwise
I_i	Processing place indicator: 1 if component i is processed on cloud, 0 if processed on mobile
$\alpha_{i,k}$	Percentage of data upload using radio interface k, for execution of component i in the cloud
$\beta_{i,k}$	Radio receiving indicator: 1 if transferred data of component i is received at radio k, 0 otherwise
d_{ij}	Data size required by component j from i
$R_k^{(\mathrm{d})}$ ($R_k^{(\mathrm{u})}$)	Downlink (Uplink) service rate for radio k
r_k	Demand rate for radio interface k
E_i	Total energy consumed by the mobile device to run component i
$E_i^{(\mathrm{m})}$ ($E_i^{(\mathrm{c})}$)	Energy consumed by the mobile device to run component i in the mobile (cloud)
$E_i^{(\mathrm{com})}$	Energy consumed by the mobile for data transfer of component i between cloud and mobile

is being remotely executed is given by $E_i^{(c)} = I_i P_{id} q_i^{(c)}$. $E_i^{(com)}$ comes into play when either the component immediately preceding the component i, or immediately succeeding component i is executed in the other entity. $E_i^{(com)}$ can be written as $E_i^{(com)} = \sum_{j=1}^{N} \sum_{k=1}^{K} (\mu_{ij}\varepsilon_{ij,k} + \mu_{ji}\varepsilon_{ji,k})$, where $\varepsilon_{ij,k}$ (or $\varepsilon_{ji,k}$) is the energy consumed in transferring data from component i (j) to component j (i) using radio interface k, when component i (j) is executed immediately before component j (i). They can be written as follows:

$$\varepsilon_{ij,k} = I_i(1 - I_j)\beta_{j,k} P_{id} \tau_{ij,k}^{(cm)} + (1 - I_i) I_j \alpha_{i,k} P_k^{(Tx)} \tau_{ij,k}^{(mc)}, \tag{4.1}$$

$$\varepsilon_{ji,k} = I_i(1 - I_j)\alpha_{j,k} P_{id} \tau_{ji,k}^{(mc)} + (1 - I_i) I_j \beta_{i,k} P_k^{(Rx)} \tau_{ji,k}^{(cm)}. \tag{4.2}$$

The first terms on the RHS of Eqs. (4.1) and (4.2) represent the idle powers consumed when the relevant component is being executed in the cloud, and second terms represent the energy consumed in transmitting or receiving the relevant data. The time needed to transfer data in the downlink communication (cloud to mobile) and uplink communication (mobile to cloud) is given by $\tau_{ij,k}^{(cm)} = \frac{d_{ij}}{R_k^{(d)}}$ and $\tau_{ij,k}^{(mc)} = \frac{d_{ji}}{R_k^{(u)}}$ respectively, where $R_k^{(d)}$ and $R_k^{(u)}$ are the downlink and uplink rates, respectively, on radio interface k. d_{ij} is the size of the data that must be transferred from component i to j.

4.2 Optimization Problem Formulation

In this section, the formulation of cognitive cloud offloading problem is presented as an integer linear programming (LP) problem. The goal is to minimize the total energy consumed by the mobile user in executing a given application under total execution time constraints. The solution to this problem must determine which components should be executed where (in the device or cloud) and what percentage of data should be allocated to each radio link for necessary uplink data transfer. The decision variables are: (1) I_i, which is the processing place indicator and gets 1 if component i is processed on cloud and 0 if processed on mobile; and (2) α_{ij} which is the component dependency indicator and gets 1 if component i must be processed before j, and 0 otherwise. This optimization is subject to the following constraints: deadline on the execution time of the application; flow rate control on each radio link used for computation offloading; and the total value of data percentage allocated to the radio interfaces for each offloaded component.

The optimization problem is mathematically formulated as

$$\min_{\boldsymbol{\alpha}, \boldsymbol{I}} E \triangleq \sum_{i=1}^{N} E_i, \tag{4.3}$$

where the constraint on the total application execution time is given by

$$\sum_{i=1}^{N} T_i \leq T_{\text{req}}, \tag{4.4}$$

where T_{req} is the execution time deadline of the application, and $T_i = T_i^{(\text{m})} + T_i^{(\text{c})} + T_i^{(\text{com})}, \forall i$. $T_i^{(\text{m})}$ represents the time taken for component i to execute in the mobile device, and is given by $T_i^{(\text{m})} = (1 - I_i) q_i^{(\text{m})}$. Similarly, $T_i^{(\text{c})} = I_i q_i^{(\text{c})}$ is the time taken to execute component i in the cloud. $T_i^{(\text{com})}$ is the time taken to complete the necessary data transfer for execution of component i, and is given by

$$\begin{aligned} T_i^{(\text{com})} = &\sum_{j=1}^{M} \sum_{k=1}^{K} I_i (1 - I_j)(\alpha_{ji} \nu_{j,k} \tau_{ji,k}^{(\text{mc})} + \alpha_{ij} \gamma_{j,k} \tau_{ji,k}^{(\text{cm})}) \\ &+ (1 - I_i) I_j (\alpha_{ij} \nu_{i,k} \tau_{ij,k}^{(\text{mc})} + \alpha_{ji} \gamma_{i,k} \tau_{ij,k}^{(\text{cm})}). \end{aligned} \tag{4.5}$$

This constraint allows us to take into consideration the potential time delays in sending and receiving the data related to each component via radio links ($T_i^{(\text{com})}, \forall i$) and trading it off optimally for energy consumption on the device.

In order for the system to be stable, the transmit data rate on the radio interfaces must be less than the service rate of each radio interface. This is represented by the second constraint:

$$\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \mu_{ij} (1 - I_i) I_j \alpha_{i,k} r_k < R_k^{(\text{u})}, \forall k. \tag{4.6}$$

The final constraint ensures that for each component, the total data allocations to the radio interfaces sum up to the total data that needs to be transferred, and is expressed as

$$\sum_{\substack{j=1 \\ j \neq i}}^{N} \mu_{ij} \sum_{k=1}^{K} \alpha_{i,k} \leq 1, \qquad \forall i. \tag{4.7}$$

4.2.1 Offloading Solution

The objective function of the optimization problem is represented in Eq. (4.3) with the constraints in Eqs. (4.4), (4.6), and (4.7). The objective function and the constraint in Eq. (4.6) involve product terms of two non-negative variables, thereby

forming a non-linear convex function. Thus, the problem can be solved using MIP (mixed integer programming) using Lagrangian multipliers: $\kappa, \zeta_k, \phi_i, \forall i, k$. The Lagrangian, $L = L(\boldsymbol{\alpha}, \boldsymbol{I}, \kappa, \boldsymbol{\zeta}, \boldsymbol{\phi})$, is expressed as

$$\begin{aligned} L \; &= \sum_{i=1}^{N} E_i(\alpha_{i,k}, I_i) + \kappa \sum_{i=1}^{N} (T_i(I_i, \alpha_{i,k}) - T_{\text{req}}) \\ &+ \sum_{k=1}^{K} \zeta_k \left(\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \mu_{ij}((1 - I_i) I_j \alpha_{i,k} r_k - R_k^{(\text{u})}) \right) \\ &+ \sum_{i=1}^{N} \phi_i \left(\sum_{\substack{j=1 \\ j \neq i}}^{N} \mu_{ij} \sum_{k=1}^{K} \alpha_{i,k} - 1 \right). \end{aligned} \tag{4.8}$$

Minimizing L will involve finding the best set of values for the parameters $\alpha_{i,k}$, and $I_i, \forall i, k$. To obtain the best offloading policy (values of I_i), L_i is written as a function of I_i and a constant term (c_1) that does not depend on I_i. That is, $L_i = \Delta_i I_i + c_1$, where

$$\Delta_i = \Lambda_i + \sum_{\substack{j=1 \\ j \neq i}}^{N} (1 - I_j) \Gamma_{i,j}^{(\text{c})} - \sum_{\substack{j=1 \\ j \neq i}}^{N} I_j \Gamma_{i,j}^{(\text{m})}, \tag{4.9}$$

and Λ_i is independent of $\alpha_{i,k}$, and can be written as

$$\Lambda_i = P_{\text{id}} q_i^{(\text{c})} - P_{\text{ac}}^{(\text{m})}(i) q_i^{(\text{m})} + \kappa (q_i^{(\text{c})} - q_i^{(\text{m})}), \tag{4.10}$$

and

$$\Gamma_{i,j}^{(\text{c})} = (P_{\text{id}} + \kappa) \sum_{k=1}^{K} (\mu_{ji} \alpha_{j,k} \tau_{ji,k}^{(\text{mc})} + \mu_{ij} \beta_{j,k} \tau_{ji,k}^{(\text{cm})}), \tag{4.11}$$

and

$$\begin{aligned} \Gamma_{i,j}^{(\text{m})} \; &= \sum_{k=1}^{K} (\alpha_{ij} \nu_{i,k} P_k^{(\text{Tx})} \tau_{ij,k}^{(\text{mc})} + \alpha_{ji} \gamma_{i,k} P_k^{(\text{Rx})} \tau_{ij,k}^{(\text{cm})} \\ &+ \kappa (\alpha_{ij} \nu_{i,k} \tau_{ij,k}^{(\text{mc})} + \alpha_{ji} \gamma_{i,k} \tau_{ij,k}^{(\text{cm})}) + \zeta_k \alpha_{ij} \nu_{i,k} r_k). \end{aligned} \tag{4.12}$$

In this algorithm, an iterative algorithm is presented to find the optimal values of $\alpha_{i,k}$ and I_i for each component. The algorithm is initialized with values for the

Lagrange multipliers ($\kappa, \zeta_k, \phi_i, \forall i, k$) as well as an initial allocation of where the given component i will be executed (values of I_i). For initial conditions, the iteration index r=initial is set to 0, and the initial value of $I_i^{(r)}$ is given by:

$$I_i^{(\text{initial})} = \begin{cases} 1 & \Lambda_i < 0, \\ 0 & \Lambda_i \geq 0. \end{cases} \tag{4.13}$$

This initial schedule of components implies that the component i will be scheduled to run in the cloud if the trade-off between energy consumption and execution time for running it on the cloud is favorable to running it on the mobile. To obtain optimum $\alpha_{i,k}$s, L for component i and radio interface k is rewritten as: $L_{i,k} = \alpha_{i,k}\Omega_{i,k}+c_2$, where $\Omega_{i,k} = \sum_{j=1}^{N} \{\mu_{ij}(1-I_i)I_j[\tau_{ij,k}^{(\text{mc})}(P_k^{(\text{Tx})}+\kappa)+\zeta_k r_k]+\phi_i\}$, and c_2 is a constant w.r.t $\alpha_{i,k}$. The optimal value of $\alpha_{i,k}$, $\alpha_{i,k}^*$ for a given value of I_i is calculated as

$$\alpha_{i,k}^* = \begin{cases} (1-I_i)\left(1 - \frac{\Omega_{i,k}}{\sum_{i=1}^{N}\sum_{k=1}^{K}\Omega_{i,k}}\right) & \sum_{\substack{j=1 \\ j\neq i}}^{N} \mu_{ij} \neq 0 \\ 0 & \sum_{\substack{j=1 \\ j\neq i}}^{N} \mu_{ij} = 0 \end{cases} \tag{4.14}$$

Now by using the value of $\alpha_{i,k}^*$ by using (4.14), I_i can be updated by

$$I_i^{(r)} = \begin{cases} 1 & \Delta_i < 0, \\ 0 & \Delta_i \geq 0. \end{cases} \tag{4.15}$$

The iterations continue until Eq. (4.8) is minimized. The algorithm converges, when the Lagrange parameters have converged. The details are given in Algorithm 4.1.

4.2.2 Convergence and Complexity of the Algorithm

In line 1, I_i and $\alpha_{i,k}$ are initialized. In a nested loop, these two variable parameters are modified such that the Lagrangian formulation in Eq. (4.8) is minimized. The strategy of lines 3–17 of the algorithm has been discussed in the previous subsection. The variables I_i and $\alpha_{i,k}$ are opportunistically updated using Eqs. (4.15) and (4.14), respectively, so that the objective function is minimized (lines 12,13 of the algorithm). The outer loop updates the Lagrangian multipliers using the subgradient method. Using the logic in [3], we see that the updated multipliers (κ, ζ_k, and $\phi_i, \forall i, k$) will converge to the optimum values of I_i and $\alpha_{i,k}, \forall i, k$.

Algorithm 4.1 Radio aware offloading schedule

initialization:
  Set $r \leftarrow 0$, modification index, $s \leftarrow 1$
  Set $I_i^{(0)}$ using Eq. (4.13)
  Set $\Delta^{(0)}$ using Eq. (4.10)
  Set $\alpha_{i,k}^{(0)}$ using Eq. (4.14)
  Set initial values for parameters $\kappa^{(s)}$, $\zeta_k^{(s)}$, $\phi_i^{(s)}$
  Set $X_r = X_s \leftarrow$False
repeat:
  if $\Delta_i^{(r)} < 0, \forall i$ **then**
    while X_r=False **do**
      calculate $\Delta_i^{(r+1)} = \Delta_i|_{I_i=I_i^{(r)}, \alpha_{i,k}=\alpha_{i,k}^{(r)}}$ by (4.9)
      calculate $I_i^{(r+1)}$ by Eq. (4.15)
      calculate $\alpha_{i,k}^{(r+1)}$ by Eq. (4.14)
      if $\exists i : \Delta_i^{(r+1)}\Delta_i^{(r)} < 0$ **then**
        Find $\min_{\tilde{i}}(\Delta_i^{(r+1)}; \forall i)$
        $I_{\tilde{i}} \rightarrow 1 - I_{\tilde{i}}$,
      end if
      if $\sum_{i=1}^{N} L_i^{(r+1)} \geq \sum_{i=1}^{N} L_i^{(r)}$ **then**
        X_r=True,
      end if
      $r \rightarrow r + 1$,
    end while
  end if
  $\kappa^{(s+1)} = \kappa^{(s)} - \varepsilon_\kappa (T_{\text{req}} - \sum_{i=1}^{N} T_i)$
  $\zeta_k^{(s+1)} = \zeta_k^{(s)} - \varepsilon_\zeta \times$
    $(R_k^{(\text{u})} - \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j\neq i}}^{N} |I_i - I_j|(1 - I_i)\mu_{ij}\alpha_{i,k} r_k), \forall k$
  $\phi_i^{(s+1)} = \phi_i^{(s)} - \varepsilon_\phi (1 - \sum_{\substack{j=1 \\ j\neq i}}^{N} \mu_{ij} \sum_{k=1}^{K} \alpha_{i,k}), \forall i$
  if $\frac{|\kappa^{(s+1)} - \kappa^{(s)}|}{\kappa^{(s+1)}} < \varepsilon_\kappa$ & $\frac{|\zeta_k^{(s+1)} - \zeta_k^{(s)}|}{\zeta_k^{(s+1)}} < \varepsilon_\zeta$ &
    $\frac{|\phi_i^{(s+1)} - \phi_i^{(s)}|}{\phi_i^{(s+1)}} < \varepsilon_\phi, \forall i, k$ **then**
    X_s = True,
  end if
  $s \rightarrow s + 1$
until any constraint in Eqs. (4.4), (4.6), (4.7) is not satisfied: (X_s=False).

Complexity of the modification loop (lines 9–23) of the algorithm is $O(r_{\max} N)$, where $r_{\max}$ is the maximum number of iterations required to find the optimum vector $\boldsymbol{I}$. Note that we assume $N > K$. Overall, the complexity of the algorithm is $O(s_{\max} r_{\max} N)$, where $s_{\max}$ is the maximum required number of iterations to satisfy all the constraints in the optimization problem. The value of $s_{\max}$ depends on the initial values in line 6 and ϵ values in lines 28, 29 of the algorithm. In the simulations, the mean values of $s_{\max}$ and $r_{\max}$ are 3 and 2, respectively. The complexity of the exhaustive search method is $O(2^N \times k)$, which is prohibitively high.

4.3 Comparison with the State of the Art

Simulations on an HTC phone, running a 14-component application and using the Amazon EC2 as the cloud, show that the solution obtained through the iterative algorithm consumes only 3% more energy than the optimal solution (obtained via exhaustive search). Please note that the system setup for evaluation is elaborated in Chap. 7.

Four scenarios are compared in this section. In the first scenario all components are executed locally in the mobile. The energy consumed in this case is used to normalize all energy values. In the second scenario the entire application is executed on the cloud (other than the first and the last components, since the mobile initiates the application). In this scenario, all data must be uploaded to the cloud. The third scenario is a brute force exhaustive search for the best values of I_i for each component. That is, we manually schedule components $i = 2$ through 13 to run on either the cloud or the mobile and calculate the associated energy and time. Note that since the first and last component must run on the mobile, we are left with $2^{(14-2)}$ combinations of possible values for the I_i's. For each combination of $\mathbf{I}$, the problem turns out to be a linear optimization over the variable set $\boldsymbol{\alpha}$. Thus, the radio allocation percentages are calculated using linear programming. The sets of I_i and $\alpha_{i,k}$, $\forall i, k$, values which minimize the energy consumption give the overall optimal solution. The approach in this scenario is called "Exhaustive search." Finally, the fourth set of results is obtained using the iterative algorithm.

Figure 4.2 shows the average energy consumption for four different approaches while the application execution time equals 3.54 s. The approaches (exhaustive search and the iterative algorithm) result in lower energy consumptions in comparison to the others. Note that 3.54 s is the minimum execution time to execute the application locally, so that the execution time deadline is satisfied in all of the approaches. On an average, the iterative algorithm consumes 3% more energy in comparison to the optimal solution (exhaustive search approach) for $T_{\text{req}} = 580$ ms. This is a fairly good trade-off for the reduced complexity of the iterative algorithm. Figure 4.3 presents the execution time of different approaches in different scenarios. While local and remote execution approaches require longer application execution

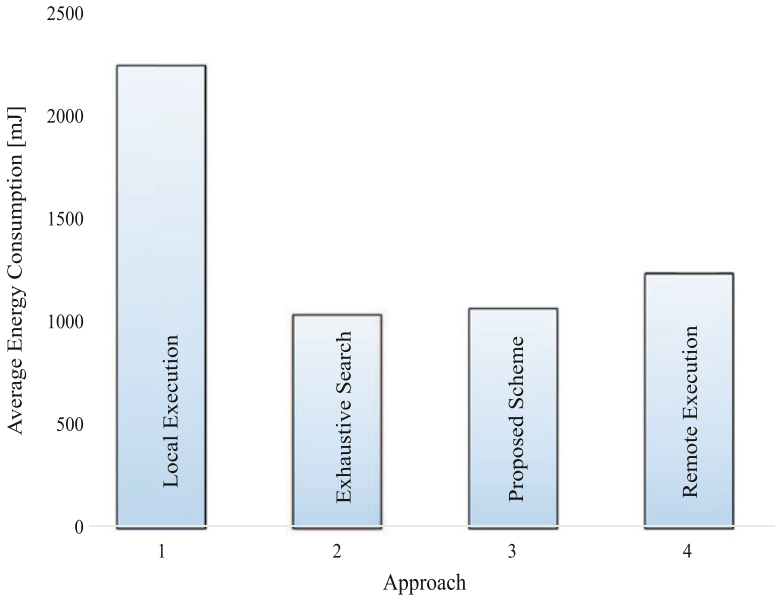


Fig. 4.2 Average energy consumption of the four approaches, while execution time equals 3.54 s

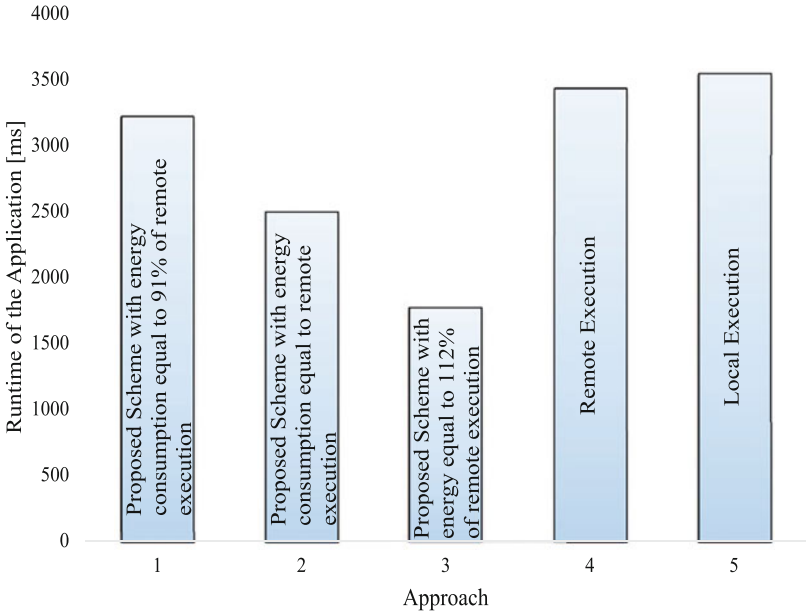


Fig. 4.3 Execution time of different approaches

time, the scheme gives us 29% and 27% faster execution time in comparison to these approaches respectively with the same amount of energy consumption to the remote execution approach. If we desire to save 9% of energy, then we have still 9% and 6% faster execution time in comparison to local and remote execution, respectively. On the other hand, if only fast execution of the application is important for us, then by expending 11% more energy than remote execution, we can achieve 50% and 48% faster runs compared to local and remote execution.

Chapter 5
Optimal Cognitive Scheduling and Cloud Offloading Using Multi-Radios

We now put together both strategies to formulate a problem that will have the maximum number of degrees of freedom in the solution. We will consider the situation where the mobile device is able to use all available (and viable) RAT interfaces as well as be unconstrained in the choice of scheduling order for all of its components. The solution to this problem can be considered the upper bound for the problem of optimal spectrum aware mobile offloading. We refer to this method as cognitive scheduling and cloud offloading (CSCO). This strategy is the benchmark for all the other approaches considered in this book. We will explore the modeling, problem formulation, solution space, and the complexity of the solution in this chapter. The next chapter will introduce a suboptimal and more practical approach for CSCO. CSCO is right in the center of all the four bubbles in the Venn diagram of Fig. 2.3.

The CSCO strategy discussed in this chapter can be managed in the cloud with the decision feedback on offloading the components sent to the mobile device. The mobile device informs the cloud of the uplink parameters from the mobile device (such as the delay for transmission, rate, queue size, and communication power) via proper control signaling before the implementation of strategy. Also, we assume that the cloud and the mobile clocks are synchronized. The feedback of decisions on offloading the components will be sent to the mobile device. The mobile device informs the cloud of the corresponding parameters via proper control signaling before cognitive cloud offloading. Note that the delay due to this control signaling feedback is negligible in comparison with the computation offloading costs which may require transferring MegaBytes of data [38, 45].

A comprehensive optimization problem is formulated with several constraints and relaxed to integer linear programming formulations. The solution is the upper-bound for time-adaptive (wireless adaptive) cloud offloading in multi-RAT devices. The objective function (net utility) is a measure of the mobile resources conserved

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_5

due to cloud offloading. This includes the energy, available memory, and CPU load minus the costs of offloading. These costs include the transmission and reception energy costs, delay due to offloading, and the data queue backlog for buffers of mobile and cloud transmitters, using all wireless interfaces. The constraints of total net utility maximization are: (1) overall execution time of the application; (2) precedence of the application components; (3) percentages of data associated with each wireless interface for mobile transmission and cloud transmission at each time slot; (4) serial computation of the components in the mobile; and (5) completion deadline of component tasks.

The CSCO scheme is applicable to a very large category of sophisticated mobile applications with arbitrary (natural) schedule order. Moreover, all the viable radio interface will be efficiently used for cognitive cloud offloading to provide the maximum available throughput.

5.1 Upper-Bound for Joint Scheduling and Cognitive Cloud Offloading

Consider a K-interface multi-radio mobile (multi-RAT) device running an N-component mobile application. Let the percentages of data (associated with offloading) uploaded to the cloud via the kth radio interface at time t be $a_{ijk}(t)$. The information from all the multi-RAT interfaces are aggregated by the cloud offload scheduler as shown in Fig. 5.1. Similarly, let the percentages of data allotted for downloading from the cloud, via the radio interface k, at time t be $b_{ijk}(t)$. Note that, $\sum_{k=1}^{K} a_{ijk}(t) = 1$, $\sum_{k=1}^{K} b_{ijk}(t) = 1$.

The parameters used in this chapter are given in Table 5.1. Consider a K-interface multi-radio mobile (multi-RAT) device running an N-component mobile application. Let the percentages of data (associated with offloading) uploaded to the cloud via the kth radio interface at time t be $a_{ijk}(t)$. Similarly, let the percentages of data allotted for downloading from the cloud, via the radio interface k, at time t be $b_{ijk}(t)$. Note that, $\sum_{k=1}^{K} a_{ijk}(t) = 1$, $\sum_{k=1}^{K} b_{ijk}(t) = 1$. For each time slot denoted by t, we define decision variable x_{pit}, which indicates whether component i completes processing at time t on the mobile ($p = 0$) or on the cloud ($p = 1$) (as shown in Fig. 2.2). The local and remote execution indicators are formulated respectively as $m_i = \sum_{t=1}^{T} x_{0it}$, $\forall i$, and $c_j = \sum_{t=1}^{T} x_{1jt}$, $\forall j$. The information from all the multi-RATs (K radio interfaces) are aggregated by the cloud offload scheduler as shown in Fig. 5.1. This figure shows that the decision maker for cognitive scheduling and cloud offloading is on the mobile side. Figure 5.1a plots the mobile to cloud transferring the required components for offloading and Fig. 5.1b illustrates the cloud to mobile transmission side for receiving the required computations by the mobile device. Here $z_{ijk}(t)$ and $y_{ijk}(t)$ are the component transferring indicators at time slot t in uplink and downlink scenarios, respectively. More clearly in Fig. 5.2a, $z_{ijk}(t)$ represents the percentage of data transferred from

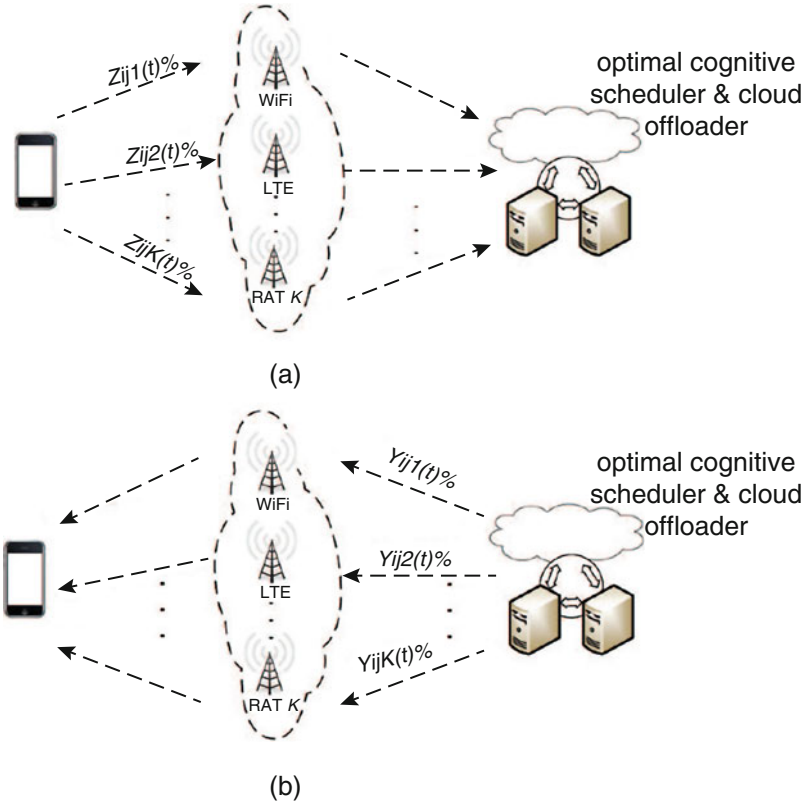


Fig. 5.1 Cognitive cloud offloading for a mobile device via multi-RATs in one time slot t for uplink and downlink scenarios ($z_{ijk}(t)$ shows the component transferring indicator from the mobile to cloud at time slot t; and $y_{ijk}(t)$ is the component transferring indicator from the cloud to mobile at time slot t). (**a**) Uplink scenario. (**b**) Downlink scenario

the mobile (where component i is executed) to the cloud (where component j will be executed) via wireless radio interface k transmitting at time slot t ($0 \leq z_{ijk}(t) \leq 1$). It is written as

$$z_{ijk}(t) = \mu_{ij} \sum_{h=1}^{t} x_{0ih} c_j a_{ijk}(t), \tag{5.1}$$

where μ_{ij} is the dependency indicator, and is "1" if execution of component j is dependent on the output data from component i, and $\sum_{h=1}^{t} x_{0ih} = 1$ after the time that component i has been processed at the mobile device and the output data of this component is ready to be offloaded. In Fig. 5.2b, $y_{ijk}(t)$ is the percentage of data transferred from the cloud (where component j is executed) to the mobile (where

Table 5.1 Parameter definitions for CSCO problem

Parameters	Definitions
x_{pit}	A binary indicator which equals to 1 if job i completes processing at time t on processing system p and otherwise equals to 0
$a_{ijk}(t)$	Percentage of allocated mobile transmission rate using radio interface k for transferring output data from component i to j at time slot t
$b_{ijk}(t)$	Percentage of allocated cloud transmission rate using radio interface k for transferring output data form component j to i at time slot t
$z_{ijk}(t)$	Component transferring indicator from the mobile to cloud at t
$y_{ijk}(t)$	Component transferring indicator from the cloud to mobile at t
m_i (c_i)	Mobile (cloud) execution indicator for component i
N	Number of the application components
K	Number of radio interfaces
T	Number of time periods to complete processing the application
t	Time index for period (t-1,t]
M_i	Memory consumed by the mobile to launch component i
code_i	Code size that is used for component i
ε	Mapping factor for the relationship between code size and the CPU instructions which is taken to be 10 [41]
w_x	Weight factor of function x
P_{ac}	Power consumed by the mobile device when it is actively processing
$P_k^{\text{tx}}(t)$ ($P_k^{\text{rx}}(t)$)	Transmit (received) power consumed by the mobile device at radio interface k in time slot t
q_i^{m} (q_i^{c})	Time to process component i in the mobile (cloud)
$A_k^{\text{mc}}(t)$($A_k^{\text{cm}}(t)$)	Data transmitted from the mobile (cloud) to the cloud (mobile) through radio interface k at time slot t
$Q_k(t)$ ($S_k(t)$)	The transmission (reception) queue of data for wireless interface k at time slot t
μ_{ij}	Dependency indicator: 1 if component i must be processed before j and 0 otherwise
$E_{\text{com}}^{\text{tx}}(t)$ ($E_{\text{com}}^{\text{rx}}(t)$)	Transmission (reception) energy consumed by the mobile device for data transferring by all possible wireless interfaces between the cloud and mobile
$\tau_{ik}^{\text{mc}}(t)$	Mobile to cloud delay that takes to transmit the output data from component i in the mobile to the cloud at wireless interface k starting by time slot t
$\tau_{ik}^{\text{cm}}(t)$	Cloud to mobile delay that takes to receive the output data from component i in cloud to the mobile at wireless interface k starting by t
U	Utility function for saving resources in the mobile
$C(t)$	The communication cost for offloading at time t
$\Omega_{\text{mc}}(t)$ ($\Omega_{\text{cm}}(t)$)	The objective function to minimize in a time-adaptive strategy for mobile (cloud) transmission communication at time slot t
V_{mc} (V_{cm})	Control parameter in mobile (cloud) transmission for Lyapunov optimization [38]

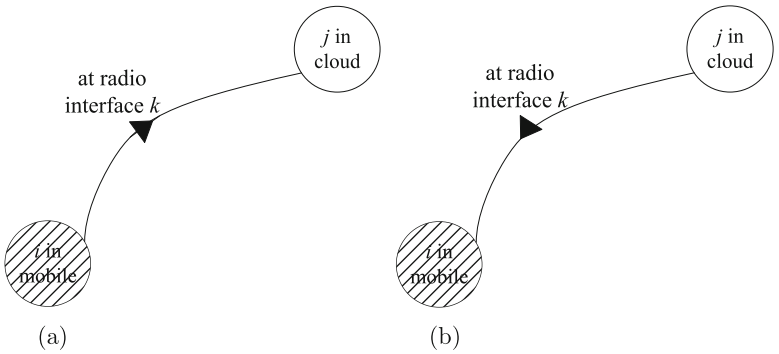


Fig. 5.2 Illustration of directed links for transferring indicators. (**a**) Directed link of $z_{ijk}(t)$. (**b**) Directed link of $y_{ijk}(t)$

component i will be executed) via wireless radio interface k receiving at time slot t ($0 \leq y_{ijk}(t) \leq 1$), and is expressed as

$$y_{ijk}(t) = \mu_{ji} \sum_{h=0}^{t} x_{1jh} m_i b_{ijk}(t), \tag{5.2}$$

where $\sum_{h=1}^{t} x_{1jh} = 1$, and is defined similar to $\sum_{h=1}^{t} x_{0ih}$, but for the component j processed in the cloud.

5.2 Optimal Cognitive Scheduling and Cloud Offloading

In this section, the formulation of cognitive scheduling and cloud offloading (CSCO) problem is presented as an integer linear programming (LP) problem. The objective of this problem is to maximize the overall utility function of the mobile device by cloud offloading. The main decision variables are: (1) x_{pit} (which indicates whether component i completes processing at time slot t in place p), (2) $a_{ijk}(t)$, and $b_{ijk}(t)$ (which indicate the percentage of allocated uplink and downlink transmission rate using radio interface k for transferring output data from component i to j at time slot t through radio interface k). The utility function trades-off the energy saved due to offloading with the cost of associated communication and can be written as:

$$\mathbf{OP}: \max \left(U - w_{\text{com}} \sum_{t=1}^{T} C(t) \right) \tag{5.3}$$

where U and $C(t)$ are the saved utility on the mobile device by remote execution and the cost of offloading at time slot t, respectively. The cost function has the weighting factor w_{com} ($0 \leq w_{\text{com}} \leq 1$) which trades-off the two parameters in the cost function. U is calculated as follows:

$$U = w_{\text{saved}} E_{\text{saved}} + w_{\text{memory}} M_{\text{saved}} + w_{\text{CPU}} \text{CPU}_{\text{saved}} \tag{5.4}$$

where the energy saved E_{saved} is given by, $E_{\text{saved}} = \sum_{i=1}^{N} c_i P_{\text{ac}} q_i^{\text{m}}$. $P_{\text{ac}} q_i^{\text{m}}$ is the energy consumed by the mobile device (local energy consumption) to execute component i, and c_i is the cloud execution indicator for component i (a list of all parameters used in this chapter is given in the table). Thus, $c_i P_{\text{ac}} q_i^{\text{m}}$ indicates the energy saved in the mobile device by offloading component i. Also, the memory saved by offloading [22] is given as: $M_{\text{saved}} = \sum_{i=1}^{N} c_i M_i$, and the CPU cycles saved in the mobile device by offloading are given by $\text{CPU}_{\text{saved}} = \sum_{i=1}^{N} c_i (\varepsilon \times \text{code}_i)$, where code_i is the code size of component i and ε maps the relationship between code size and the CPU instructions [41]. The relation between weighting factors is expressed as $w_{\text{saved}} + w_{\text{com}} + w_{\text{memory}} + w_{\text{CPU}} = 1$. These weights indicate the priorities for different objectives such as maximizing the energy saved by remote execution, memory saved, CPU saved, or minimizing the communication costs. The summation constraints for weights show the priority of each function in comparison to the others. By setting weights, the solution is pre-biased more towards the favored side for different applications based on these criteria. By adjusting the weights between "stay in the mobile" or "offload to the cloud," an extra control knob is added that lets us weight one or the other more.

Monetary costs of using cloud services could be significant in weighting [53], and we might want to favor offloading to the cloud server slightly less. The cost function is written as

$$C(t) = C_{\text{mc}}(t) + C_{\text{cm}}(t), \tag{5.5}$$

where $C_{\text{mc}}(t)$ and $C_{\text{cm}}(t)$ denote the mobile and cloud transmission costs at time slot t, respectively.

Mobile to cloud transfer costs include the energy consumed for transmission from the mobile to the cloud ($E_{\text{com}}^{\text{tx}}(t)$) plus the time averaged, aggregate radio interface queue ($\overline{Q}$). $E_{\text{com}}^{\text{tx}}(t)$ is expressed as:

$$E_{\text{com}}^{\text{tx}}(t) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} z_{ijk}(t) P_k^{\text{tx}}(t) \tau_{ik}^{\text{mc}}(t). \tag{5.6}$$

where $P_k^{\text{tx}}(t)$ shows the transmission power level consumed by the mobile device at radio interface k in time slot t, and $\tau_{ik}^{\text{mc}}(t)$ represents the mobile to cloud delay to transmit the output data from component i in the mobile to the cloud at wireless interface k starting by time slot t.

The objective is to minimize the overall communications cost, subject to a constraint on the average, aggregated queue length. That is, $\min \sum_{t=1}^{T} E_{\text{com}}^{\text{tx}}(t)$, s.t. $\overline{Q} = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{E}\{|Q_k(t)|\} < \infty$, where $Q_k(t)$ is the transmission queue of data for wireless interface k at time slot t. Using standard Lyapunov optimization formulation [38], this optimization problem is written as:

$$\begin{aligned} \min C_{\text{mc}}(t) \; &= E_{\text{com}}^{\text{tx}}(t) \\ &\quad - V_{\text{mc}} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} z_{ijk}(t) Q_k(t) A_k^{\text{mc}}(t). \end{aligned} \tag{5.7}$$

The decision variable for this cost function is $z_{ijk}(t)\ \forall p, i, j, k, t$ (which is formed based on $a_{ijk}(t)$ and x_{pit}). The parameter V_{mc} serves as a control knob [45] by adjusting the trade-off between minimization of the cost ($E_{\text{com}}^{\text{Tx}}$) and satisfaction of the queue stability constraint for all the radio interfaces. $A_k^{\text{mc}}(t)$ is the data transmitted from the mobile to the cloud through radio interface k at time slot t. The RHS of Eq. (5.7) is the representative of averaged aggregated queue length ($\overline{Q}$). The control parameter $V_{mc} > 0$ represents the trade-off between communication energy and averaged aggregated queue length, i.e., how much we shall emphasize the uplink communication energy minimization compared to averaged aggregated queue length. When V_{mc} decreases, then it emphasizes the effect of $E_{\text{com}}^{\text{tx}}(t)$.

The cloud to mobile communication costs, $E_{\text{com}}^{\text{rx}}(t)$, can be written similar to the energy costs for mobile to cloud communications shown in Eq. (5.6). Hence, $E_{\text{com}}^{\text{rx}}(t)$ is formulated as:

$$E_{\text{com}}^{\text{rx}}(t) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} y_{ijk}(t) P_k^{\text{rx}}(t) \tau_{jk}^{\text{cm}}(t), \tag{5.8}$$

where all parameters are as defined in the table.

The overall cloud to mobile communication costs must be minimized w.r.t constraints on the reception queue. Hence, $\min \sum_{t=1}^{T} E_{\text{com}}^{\text{rx}}(t)$ s.t. $\overline{S} = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{E}\{|S_k(t)|\} < \infty$. $S_k(t)$ is the reception queue for wireless interface k at time slot t. Just as in the transmitter, the conditional optimization can be rewritten as $\min C_{\text{cm}}(t) = E_{\text{com}}^{\text{rx}}(t) - V_{\text{cm}} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} y_{ijk}(t) S_k(t) A_k^{\text{cm}}(t)$, with parameters as defined in Table 5.1. V_{cm} is the control parameter for the cloud transmission queue that trades-off the receiver energy costs and the satisfaction constraint of averaged aggregated queue length ($\overline{S}$).

Overall Execution Time for the Application Since most applications also come with execution deadlines, the following condition is imposed:

$$0 < \sum_{t=1}^{T} t x_{0Nt} \leq T \qquad\qquad t = 1, \ldots, T \tag{5.9}$$

where $\sum_{t=1}^{T} t \cdot x_{0Nt}$ denotes the completion time of the last component (N) on the mobile ($p = 0$) and T is the application deadline.

In order to make sure that the solution only allows for *processing each component exactly once*, we note that $m_i + c_i = 1, \forall i$.

Precedence Constraint As mentioned earlier, some components will depend on the execution of other components. In order to ensure that this precedence constraints are met, the following constraint is added:

$$\begin{aligned} &\mu_{ij} \sum_{p=0}^{1} \sum_{s=1}^{t+v_j(t)} x_{pjs} \leq \mu_{ij} \sum_{p=0}^{1} \sum_{s=1}^{t} x_{pis}, \\ &\text{if } i \prec j, t = m_i q_i^{\mathrm{m}} + c_i q_i^{\mathrm{c}}, \ldots, T - v_j, \end{aligned} \tag{5.10}$$

where $i \prec j$ implies that component i must complete execution before j can begin execution, and

$$v_j(t) = m_j q_j^{\mathrm{m}} + c_j q_j^{\mathrm{c}} + \sum_{k=1}^{K} (y_{jik}(t)\tau_{jk}^{\mathrm{cm}}(t) + z_{ijk}(t)\tau_{ik}^{\mathrm{mc}}(t)). \tag{5.11}$$

Completion Deadline Each component j can complete execution only after all of its precedent components have completed execution, plus the time to process component j itself, and the time to transfer required data to the execution site of j if i is not on that same site. This constraint is given as:

$$\begin{aligned} &\sum_{p=0}^{1} \sum_{h=1}^{t} h x_{pih} + \sum_{k=1}^{K} (z_{ijk}(t)\tau_{ik}^{\mathrm{mc}}(t) + y_{jik}(t)\tau_{jk}^{\mathrm{cm}}(t)) \\ &\qquad + \sum_{h=1}^{t} x_{0jh} q_j^{\mathrm{m}} + \sum_{h=1}^{t} x_{1jh} q_j^{\mathrm{c}} \leq \sum_{p=0}^{1} \sum_{s=1}^{T} s x_{pjs}, \ i \prec j, \forall i, j, t. \end{aligned} \tag{5.12}$$

$\sum_{p=0}^{1} \sum_{h=1}^{t} h x_{pih}$ is the number of time slots that component i takes to complete execution, within time period [0, t). $\sum_{k=1}^{K} (z_{ijk}(t)\tau_{ik}^{\mathrm{mc}}(t) + y_{jik}(t)\tau_{ik}^{\mathrm{cm}}(t))$ represents the time to transfer the output data related to component i either from mobile to cloud or cloud to mobile by time slot t. Also, the time to process component j itself (either in the mobile or cloud) by time slot t is set to $\sum_{h=1}^{t} x_{0jh} q_j^{\mathrm{m}} + \sum_{h=1}^{t} x_{1jh} q_j^{\mathrm{c}}$. Summation of all these values should be less than the time to complete processing of component j $\left(\sum_{p=0}^{1} \sum_{t=1}^{T} t x_{pjt} \right)$.

Serial Execution in Mobile The components processing in the mobile must execute serially. Thus, for each time interval $[t-1, t)$, we can have at most one component for processing in the mobile, which can be written as

$$\sum_{i=1}^{N} \sum_{s=t}^{\min\{t+m_i q_i^{\mathrm{m}}+c_i q_i^{\mathrm{c}}-1, T\}} x_{0is} \leq 1 \qquad t = 1, \ldots, T \tag{5.13}$$

Recently, the new smartphones have the capability of parallel processing. However, a wireless-based scheduler is required in the operating system of device, which increases the cost of system complexity in the mobile devices and may not be practical in the real-world scenarios.

Data Rate Constraints The fraction of uplink data transfer between two components, say i and j, through all the radio interfaces should sum to 1, at most. Therefore, the overall constraint for uplink transfer can be written as:

$$\sum_{t=1}^{T} \sum_{k=1}^{K} z_{ijk}(t) \leq 1 \qquad \forall i, j. \tag{5.14}$$

Considering the definition of $z_{ijk}(t)$ as given in Eqs. (5.1) and (5.14) must be equal to 1 if $i \prec j$ ($\mu_{ij} = 1$), and also component i is executed in the mobile and component j is executed in the cloud. Otherwise it gets zero. It also needs to guarantee all the data will be offloaded through the available radio interfaces, which means: $\sum_{t=1}^{T} \sum_{k=1}^{K} a_{ijk}(t) = 1, \forall i \prec j$, and $a_{ijk}(t) \geq 0$, $\forall i, j, k, t$. Moreover, moving data from execution of component in cloud happens only at one time after component i has been processed, that is $\sum_{t=1}^{T} \sum_{k=1}^{K} f_{ijkt} \leq 1$,

Similarly for the downlink scenario, the fraction of downlink data transfer between two components through all the radio interfaces must be at most 1, which can be written as: $\sum_{t=1}^{T} \sum_{k=1}^{K} y_{ijk}(t) \leq 1$, $\forall i, j$. Also, the other constraints in downlink transfer are given as: $\sum_{t=1}^{T} \sum_{k=1}^{K} b_{ijk}(t) = 1, \forall j \prec i$, and $b_{ijk}(t) \geq 0$, $\forall i, j, k, t,$, and $\sum_{t=1}^{T} \sum_{k=1}^{K} g_{ijk}(t) \leq 1$,

In the raw formulation, we have indicator variables $z_{ijk}(t)$ and $y_{ijk}(t)$ (see Eqs. (5.1) and (5.2)) that are cubic terms thereby violating the linearity of the constraints.

We now state and prove a mathematical lemma and corollary that shows that a cubic constraint can be linearized in a special case, which is valid for this problem.

5.2.1 Linearizing the Optimization Problem

The following lemma states that a quadratic constraint can be linearized in special cases.

Lemma 5.1 *In a linear program, there exists a method to linearize a quadratic term, XY, when at least one of the variables is a $0-1$ binary variable and the other is bounded (Rubin, http://orinanobworld.blogspot.de/2010/10/binary-variables-and-quadratic-terms.html, Sept 2014).*

Proof without loss of generality assume X is the binary variable and Y is the bounded variable such that $\sigma \leq Y \leq \rho$ (σ and ρ are known). Note that when $X = 0$, $Z = 0$ and when $X = 1$, $Z = Y$.

Add the following four linear constraints:

$$Z \leq \rho X \tag{5.15}$$

$$Z \geq \sigma X \tag{5.16}$$

$$Z \leq Y - \sigma(1 - X) \tag{5.17}$$

$$Z \geq Y - \rho(1 - X) \tag{5.18}$$

When $X = 0$, the first pair of inequalities above forces $0 \leq Z \leq 0$ which implies $Z = 0$. The second pair of inequalities results in $Y - \rho \leq Z \leq Y - \sigma$ which is satisfied by $Z = 0$.

When $X = 1$, the second pair of inequalities above forces $Y \leq Z \leq Y$ which implies $Z = Y$. The first pair of inequalities results in $\sigma \leq Z \leq \rho$ which is satisfied by $Z = Y$.

Therefore, by letting $Z = XY$ and adding the above four constraints, the quadratic term XY is linearized by replacing it with the bounded variable Z. ■

Corollary 5.1 *A product of n variables, where one variable is bounded and the remaining $n-1$ are $0-1$ binary variables, can be linearized by successively applying Lemma 5.1.*

Proof Let $Z = X_1X_2X_3X_4 \cdots X_n$ where X_1 is bounded and $X_2, \ldots, X_n$ are $0-1$ variables. Applying Lemma 5.1, X_1X_2 can be replaced by a bounded variable Z_2. Now $Z = Z_2X_3X_4 \cdots X_n$. Again applying Lemma 5.1, Z_2X_3 can be replaced by a bounded variable Z_3 resulting in $Z = Z_3X_4 \cdots X_n$. After $n-3$ more such iterations, we will have $Z = Z_n$ where Z_n is a bounded variable with all linear constraints and no non-linear terms. ■

In the formulation $z_{ijk}(t)$ (and $y_{ijk}(t)$) is expressed as a cubic term where two factors in the cubic term are $0-1$ variables and the other one is bounded: $0 \leq a_{ijk}(t) \leq 1$ ($0 \leq b_{ijk}(t) \leq 1$). By Corollary 5.1, each can be linearized resulting in z_{ijk} (and $y_{ijk}(t)$) being expressed as a linear term.

In the uplink transfer of this chapter, we define $f_{ijkt} = \sum_{h=0}^{t} x_{0ih} \times a_{ijk}(t)$ as a new mathematical parameter, and hence $z_{ijkt} = f_{ijkt} \times (\mu_{ij}c_j)$. Now we apply Lemma 5.1 to obtain the following constraints: $z_{ijk}(t) \leq f_{ijkt}$, $z_{ijk}(t) \leq \mu_{ij}c_j$, $z_{ijk}(t) \geq 0$, $z_{ijk}(t) \geq \mu_{ij}c_j - (1 - f_{ijkt})$, $\forall i, j, k, t$.

We need to apply the Lemma 5.1 once again for the quadratic term of f_{ijkt} as: $f_{ijkt} \leq \sum_{h=0}^{t} x_{0ih}$, $f_{ijkt} \leq a_{ijk}(t)$, $f_{ijkt} \geq a_{ijk}(t) - (1 - \sum_{h=0}^{t} x_{0ih})$, $f_{ijkt} \geq 0$, $\forall i, j, k, t$. Thus, the cubic term of three decision variables is converted to a new linear decision variable.

Similarly in the downlink transfer, for $y_{ijk}(t) = \mu_{ji} \sum_{h=0}^{t} x_{1jh} m_i b_{ijk}(t)$, we can write: $y_{ijk}(t) \leq g_{ijkt}$, $y_{ijk}(t) \leq \mu_{ji} m_i$, $y_{ijk}(t) \geq \mu_{ji} m_i - (1 - g_{ijkt})$, $y_{ijk}(t) \geq 0$, $\forall i, j, k, t$, where $g_{ijkt} = \sum_{h=0}^{t} x_{1jh} \times b_{ijk}(t)$. We apply Lemma 5.1 for g_{ijkt} as $g_{ijkt} \leq \sum_{h=0}^{t} x_{1jh}$, $g_{ijkt} \leq b_{ijk}(t)$, $g_{ijkt} \geq b_{ijk}(t) - (1 - \sum_{h=0}^{t} x_{1jh})$, $g_{ijkt} \geq 0$, $\forall i, j, k, t$.

System Complexity Since we have an integer linear problem (ILP), the number of constraints determines the scheduling overhead and affects memory usage more than the number of variables (http://www-01.ibm.com/support/docview.wss?uid=swg21399933). In this chapter, the upper bound for number of constraints is $9TN^2 + N^2 + 2T$, which is a function of application runtime and number of components (order of complexity is $O(TN^2)$). Also, the number of variables is $6KTN^2$ (order of complexity is $O(TN^2)$. However, we do not experience schedule overhead in the offloading scenario because the strategy will be executed in the cloud server where the RAM is high enough.

5.3 Comparison with the State of Art

The performance of the CSCO is evaluated using real data from an HTC smartphone (mobile device) and NSFCloud (remote server) using two radios WiFi and LTE (the evaluation setup is detailed in Chap. 7). Data was gathered from indoor as well as outdoor wireless environments. Evaluations show that CSCO lowers energy consumption by 23–68% and reduces latency by 28–66% in comparison to the best-interface protocol, offline schemes, and mobile-only and cloud-only executions. We compare the optimal CSCO scheme with several variants of the CSCO scheme in both the indoor and outdoor environments as well as current state of the art in this field. The CSCO scheme will be compared with several other schemes including: (1) mobile-only execution where all the application components are processed in the mobile device (extreme case for mobile execution); (2) On-Off remote cloud offloading (On-Off RCO) where all components must be offloaded and at each time epoch, the offload is scheduled on the best available wireless interface (On-Off) (extreme case for cloud execution); (3) the dynamic offloading algorithm (DOA) proposed in [19], 14], a partial offline strategy which uses On-Off multi-RAT strategy; (4) HELVM proposed in [40], where a dynamic scheduler for partial offloading is studied; (5) the revised HEFT algorithm [48] for joint scheduling tasks (RHJS) on multiple cores that is proposed in [28], and (6) several variants of the CSCO work, described below:

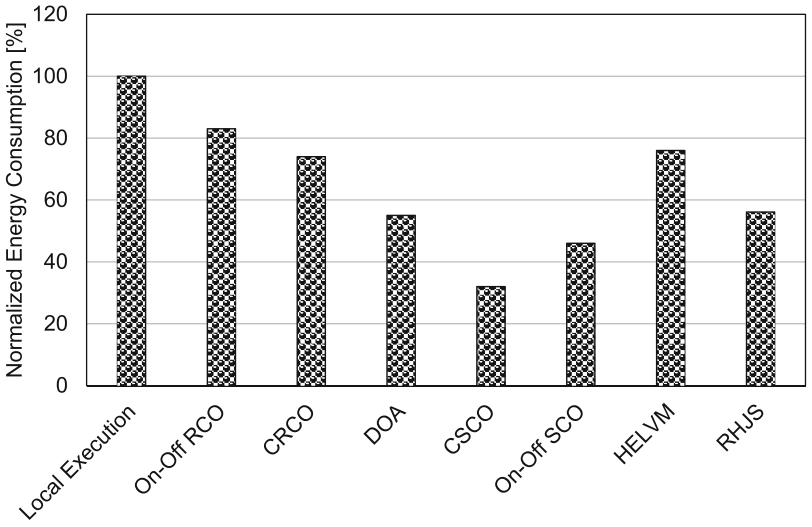


Fig. 5.3 Total energy consumed by the mobile device (normalized to the energy consumed by mobile-only execution) for all the schemes using the face recognition application in http://darnok.org/programming/face-recognition/

- Cognitive scheduling and cloud offloading (CSCO): This is the optimal scheme for cognitive joint scheduling and offloading.
- On-Off scheduling and cloud offloading (On-Off SCO): In this variant, we use the joint scheduling–offloading strategy, with an additional constraint of using only the best radio interface at every time slot. This is done to enable a fair comparison with CSCO, where only the best radio interface is used at any time slot.
- Cognitive remote cloud offloading (CRCO) where all components must be offloaded and at each time epoch, the cognitive offload is scheduled on the multi-RAT networking.

For Figs. 5.3 and 5.4, a face recognition application with 10 components (http://darnok.org/programming/face-recognition/) was used along with the same wireless network parameters in http://www.3gpp.org/ftp/tsg-ran/wg4-radio/ in order to compare our results with the results in the state of the art.

Figure 5.3 shows the comparison of normalized (normalized w.r.t. the energy consumed by mobile-only execution) energy consumed by the mobile device for eight schemes. We observe that CSCO consumes 68%, 51%, 42%, 23%, 14%, 44%, and 24% less energy in comparison to the schemes using local execution, On-Off RCO, CRCO, DOA, On-Off SCO, HELVM, and RHJS, respectively.

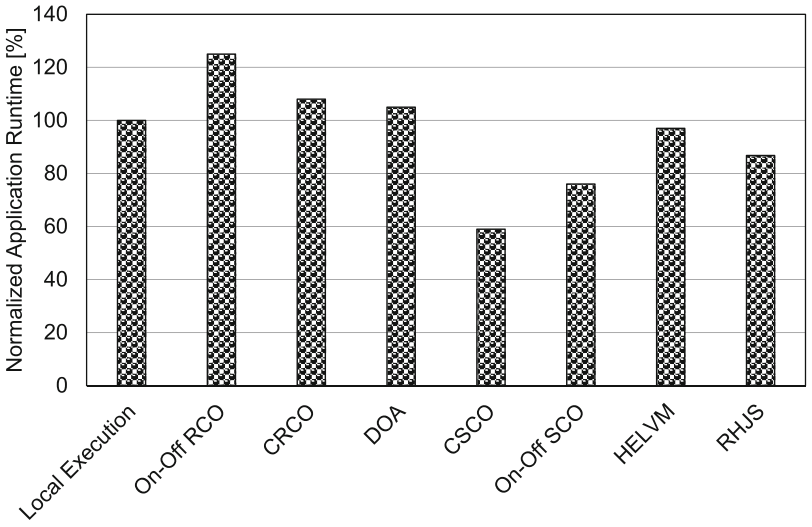


Fig. 5.4 Total application runtime (normalized to application runtime in the mobile) for all the schemes using the face recognition application in http://darnok.org/programming/face-recognition/

In Fig. 5.4, runtime values of the application (http://darnok.org/programming/face-recognition/) (normalized to the scheme with local execution) for the eight schemes are compared. The CSCO enables the application to be processed 41%, 66%, 49%, 46%, 17%, 38%, and 28% faster compared to the schemes using local execution, On-Off RCO, CRCO, DOA, On-Off SCO, HELVM, and RHJS, respectively. Therefore, CSCO is *joint energy and time efficient* in comparison to the other schemes.

Chapter 6
Time-Adaptive and Cognitive Cloud Offloading Using Multiple Radios

The solution developed in Chap. 5 can be computationally heavy, especially for very complex applications. It is therefore useful to develop practical heuristics that can achieve near optimal performances. This chapter introduces a practical time-adaptive and wireless aware heuristic for CSCO that will achieve two goals simultaneously: (1) assign the schedule order of application's components; and (2) determine the offloading strategy while optimally allocating the percentage of data to be sent by the mobile and the cloud via each wireless interface. This chapter investigates a suboptimal but more practical approach for cognitive scheduling and cloud offloading. The schemes introduced in this chapter (like the one introduced in Chap. 5) are also included in the center of the Venn diagram shown in Fig. 2.3. This means that this approach covers all the four desired characteristics of offloading. This scheme maximizes the net utility, including the mobile device resources (battery, CPU, and memory) saved by cloud offloading with the penalty of communication costs such as latency, energy consumed for offloading, and queue instability of radio interfaces imposed by cognitive cloud offloading.

The heuristic cognitive cloud offloading uses a fast and low-complex strategy for multi-RAT-aware computation offloading of multi-component mobile applications in a time-adaptive scenario. This allows this algorithm to adapt quickly to the changing wireless and network parameters of all the available wireless interfaces such as rates, delay, and power. For optimal offloading decisions, the percentages of data to be sent by the mobile and the cloud via each wireless interface are updated at each time slot. A comprehensive model for the net utility saved by offloading components to be maximized based on the new scenario of time-adaptive multi-RAT networking was provided. 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 since using multiple wireless interfaces has been considered at the same time, queue stability of the mobile and cloud transmission buffers for aggregated interfaces is guaranteed in the offloading strategy.

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_6

6.1 Network and Application Model

As before, let us consider a mobile device with K radio interfaces in a wireless network, running an N-component mobile application. 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.

The system can be explained using the system model shown in Fig. 6.1. In this figure, at time slot t $\forall t$, $\alpha_k(t)\%$ of the required data for offloading is sent by the

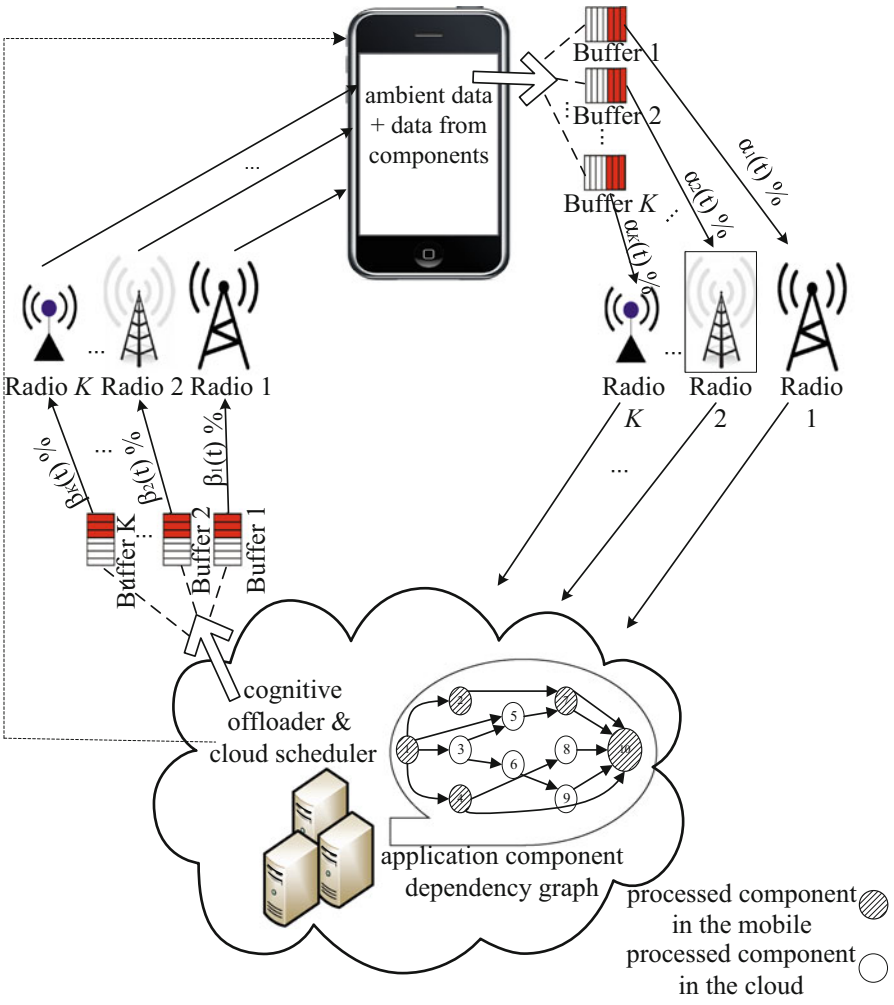
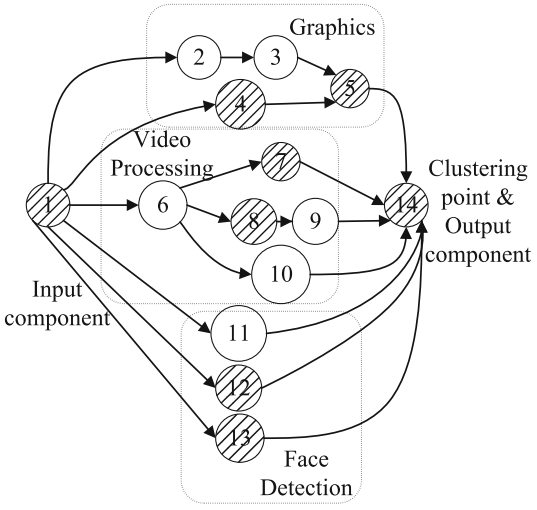


Fig. 6.1 Cognitive offloading for multi-RAT enabled wireless devices

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.

6.1.1 Example Component Dependency Graph for a Practical Application

We will use the same example video navigation application introduced in Chap. 3 with $N = 14$ components to explain the salient points of the heuristics developed in this chapter. The component dependency graph of these applications was first shown in Fig. 3.1a and is repeated in Fig. 6.2a for ease of reference. The 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. Figure 6.2b shows an example of the stages of processing for the application with the CDG shown in Fig. 6.2a at the first 5 time slots. More details of the algorithm are described in Sect. 6.2. Some components (e.g., components 2, 6, and 11 in the cloud and 4 in the mobile for this example) can be scheduled for parallel execution.



(a)

Fig. 6.2 Time-adaptive scheduling–offloading for an example of 14-component mobile application. (**a**) CDG of the application. (**b**) Scheduling strategy

Fig. 6.2 (continued)

A smart cognitive cloud offloading algorithm will trade-off the benefits of wireless offloading, namely energy and time savings (when components can be parallelly 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) is negligible in comparison to when the data must be transferred between entities. Also, we assume that the cloud and the mobile clocks are synchronized [7, 30]. 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. 6.1). In the online stage for offloading at each time slot, two sub-strategies are studied for cloud offloading: (1) 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 (2) 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.

6.1.2 Net Utility Function

To determine the best strategy for joint scheduling–offloading, we need to redefine an appropriate net utility function for this scenario. The notations used for these terms and the other parameters in this chapter are defined in Table 6.1. To determine the scheduling–offloading strategy for component i, two decision variables are defined for a time slot t as follows:

$$I_i(t) \equiv \begin{cases} 1 & \text{component } i \text{ starts offloading at } t, \\ 0 & \text{otherwise,} \end{cases}$$
$$X_i(t) \equiv \begin{cases} 1 & \text{component } i \text{ starts executing locally at } t, \\ 0 & \text{otherwise.} \end{cases}$$

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:

$$\begin{aligned} U(t) = {} & w_{\text{saved}} E_{\text{saved}}(t) + w_{\text{memory}} M_{\text{saved}}(t) \\ & + w_{\text{CPU}} \text{CPU}_{\text{saved}}(t) - w_{\text{com}} C_{\text{com}}(t). \end{aligned} \tag{6.1}$$

Table 6.1 Parameter definitions for the heuristic problem

Parameters	Definitions
$U(t)$	Net utility function at time t
T	Number of time slots to complete processing the application
x	Span of each time slot
M_i	Memory consumed by the mobile device to launch component i
code_i	Code size to launch component i
ε	Mapping factor to relate code size and the CPU instructions [41]
w_x	Weight factor of function x
P_i^{ac}	Active power consumed by the mobile to process 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_i^{m} (q_i^{c})	Number of time slots to process component i in the mobile (cloud)
γ	Weight factor (to adjust the wait time for offloading)
$A_k^{\text{mc}}(t)$($A_k^{\text{cl}}(t)$)	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
μ_{ij}	Dependency indicator: 1 if component i processed before j and 0 otherwise
$Q_k(t)$ ($S_k(t)$)	The transmission queue of data from the mobile (cloud) side for wireless interface k at time slot t
$I_i(t)$	Offloading indicator: 1 if the mobile starts to offload component i at 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
$X_i(t)$	Local execution indicator: 1 if the mobile starts to execute component i locally at time slot t
$m_i(t)$	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
$z_{ij}(t)$	Indicator for communication requirement: 1 if component i is executed in the mobile and j is offloaded to the cloud by time 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
$E_{\text{com}}^{\text{Tx}}(t)$ ($E_{\text{com}}^{\text{Rx}}(t)$)	Energy consumed for the mobile transmission (reception)
$\tau_{i,k}^{\text{mc}}(t)$ ($\tau_{i,k}^{\text{cm}}(t)$)	Delay (# slots) to transmit the output data from component i in the mobile (cloud) to the cloud (mobile) at interface k starting by t
$T_{\text{th}}^{\text{Tx}}$ ($T_{\text{th}}^{\text{Rx}}$)	Threshold number of time slots for transmission from mobile (cloud) to cloud (mobile)
$l_{ji}(t)$	The time slots to process the preceding component j, and transfer the output data from component j to i by t
$\Omega_{\text{mc}}(t)$ ($\Omega_{\text{cm}}(t)$)	The objective function for mobile (cloud) transmission at time t
V_{mc} (V_{cm})	Control parameter in mobile (cloud) transmission

The weights for the individual costs and benefits are chosen such that $w_{\text{saved}} = 1 - w_{\text{com}}$, and $w_{\text{CPU}} = 1 - w_{\text{memory}}$.

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 ($P_i^{\text{ac}} q_i^{\text{m}}$), which is given by:

$$E_{\text{saved}}(t) = \sum_{i=1}^{N} c_i(t) P_i^{\text{ac}} q_i^{\text{m}}, \tag{6.2}$$

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:

$$M_{\text{saved}}(t) = \sum_{i=1}^{N} c_i(t) M_i, \tag{6.3}$$

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:

$$\text{CPU}_{\text{saved}}(t) = \sum_{i=1}^{N} c_i(t)(\varepsilon \text{code}_i), \tag{6.4}$$

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.

6.2 Cognitive Offloading and Scheduling Heuristic

In this section, a heuristic approach is studied to find a time-adaptive 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. The complete algorithm is depicted in Algorithm 6.1. A detailed description of the algorithm follows.

6.2.1 Optional Offline Stage

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, some of the components are eliminated as unsuitable for offloading at the outset, maximizing the instantaneous utility values at time t_0 (e.g., components 5 and 7 in Fig. 6.2b). 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, the components that contribute the most to an increase in the net utility are identified if scheduled in the mobile device and then eliminated from being considered for offloading. First an approximate value is obtained for the optimal solution based on the information available at time t_0. To do this the instantaneous net utility given by Eq. (6.1) is maximized 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,

$$\begin{aligned}\hat{C}_{\text{com}}(t_0) = w_{\text{com}} \Bigg\{ \sum_{i=1}^{N}\sum_{j=1}^{N} \mu_{ij} m_i(t_0) c_j(t_0) \min_k \left(P_k^{\text{Tx}}(t_0)\tau_{\text{th}}^{\text{mc}}\right) \\ + \mu_{ij} c_i(t_0) m_j(t_0) \min_k \left(P_k^{\text{Rx}}(t_0)\tau_{\text{th}}^{\text{cm}}\right) \Bigg\},\end{aligned} \tag{6.5}$$

where μ_{ij} represents the dependency indicator (1 if component i must be processed before j, and 0 otherwise), and $\tau_{\text{th}}^{\text{mc}}$ and $\tau_{\text{th}}^{\text{cm}}$ are threshold values for the transmission at the mobile and cloud ends for each component, respectively. To obtain the approximation given by Eq. (6.5) , we assume that the Tx and Rx power levels ($P_k^{\text{Tx}}(t_0)$ and $P_k^{\text{Rx}}(t_0)$) are fixed when computing these values for the offline stage. By selecting the wireless interface with the lowest Tx and Rx energy levels, the minimum energy consumed for communication over the wireless interfaces ($\forall k = 1, 2, \ldots, K$) is obtained in Eq. (6.5) with the initial information in the offline stage (i.e., in the transmission, we have: $\min_k \left(P_k^{\text{Tx}}(t_0)\tau_{\text{th}}^{\text{mc}}\right)$). The optimization problem in the offline stage can be written as:

$$\begin{aligned}\text{OP}_{\text{off:}} \max_{\mathbf{c}} \; & w_{\text{saved}} E_{\text{saved}}(t_0) + w_{\text{memory}} M_{\text{saved}}(t_0) \\ & + w_{\text{CPU}} \text{CPU}_{\text{saved}}(t_0) - \hat{C}_{\text{com}}(t_0),\end{aligned} \tag{6.6}$$

s.t.

$$\sum_{i=1}^{N} m_i(t_0) q_i^{\mathrm{m}} + \sum_{i=1}^{N} c_i(t_0) q_i^{\mathrm{c}} + \sum_{i=1}^{N} \sum_{j=1}^{N} \mu_{ij} (m_i(t_0) c_j(t_0) \tau_{\mathrm{th}}^{\mathrm{mc}} + c_i(t_0) m_j(t_0) \tau_{\mathrm{th}}^{\mathrm{cm}}) \leq T, \tag{6.7}$$

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 ($P_i^{\mathrm{ac}} q_i^{\mathrm{m}}$), 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 Eq. (6.5) 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. 6.2b). Otherwise, it will be scheduled for local execution based on the precedence constraints dictated by the component dependency graph 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 non-linear. To convert this to a linear optimization problem, the terms $m_i(t_0) c_j(t_0), \forall i, j$, are replaced with a new variable $z_{ij}(t_0)$ and new constraints are added to make the new optimization problem equivalent to the original one (Rubin, http://orinanobworld.blogspot.de/2010/10/binary-variables-and-quadratic-terms.html, Sept 2014). These constraints are as follows:

$$z_{ij}(t_0) \leq m_i(t_0), \tag{6.8}$$

$$z_{ij}(t_0) \geq 0, \tag{6.9}$$

$$z_{ij}(t_0) \leq c_j(t_0), \tag{6.10}$$

$$z_{ij}(t_0) \geq c_j(t_0) - (1 - m_i(t_0)), \tag{6.11}$$

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 heuristic.

6.2.2 *Online Stage*

In the online stage of the algorithm (starting from t_1 in Fig. 6.2b), the following precedence constraints must be checked to see if a component is eligible for execution at the current time slot t:

1. Each component must be processed only once, either in the mobile or in the cloud. This constraint is mathematically written as:

$$m_i(t-1) + c_i(t-1) < 1, \qquad \forall i. \tag{6.12}$$

This equation shows that component i has not started execution (either locally or remotely) by time slot t.

2. All the components on which component i depends should have completed execution before starting the offload process or local execution of component i. That is j precedence symbol i, for all components j that must be completed before component i. This should be checked from the start of the application runtime until $(t-1)$th time slot. Therefore, the precedence constraints are:

$$\begin{aligned} & m_i(t - l_{ji}(t-1)) + c_i(t - l_{ji}(t-1)) \leq \\ & \quad m_j(t-1) + c_j(t-1), \\ & \quad \forall j \prec i, \quad m_j(t-1) + c_j(t-1) = 1, \\ & \quad t = l_{ji}(t-1) + 1 \ldots T, \end{aligned} \tag{6.13}$$

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 is time dependent (because of varying mobile device resource availability and wireless data rates). The duration $l_{ji}(t)$ is expressed as:

$$\begin{aligned} l_{ji}(t) = {} & m_j(t) q_j^{\mathrm{m}} + c_j(t) q_j^{\mathrm{c}} \\ & + \sum_{s=1}^{t} \sum_{k=1}^{K} \left(z_{ji} \alpha_k(s) \tau_{j,k}^{\mathrm{mc}}(s) + z_{ij} \beta_k(s) \tau_{j,k}^{\mathrm{cm}}(s) \right), \end{aligned} \tag{6.14}$$

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 Eq. (6.14) 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 $s \in \{1, 2, \ldots, t-1\}$.

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 the 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:

$$\overline{Q} = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{E}\{|Q_k(t)|\} < \infty, \tag{6.15}$$

where $Q_k(t)$ is the transmission queue of wireless interface k at time slot t from the mobile side. The above problem is cast as a Lyapunov optimization [38] 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 $\Delta_{\text{mc}}(\boldsymbol{Q(t)}) \triangleq \mathbb{E}\{L(\boldsymbol{Q(t+1)}) - L(\boldsymbol{Q(t)})|\boldsymbol{Q(t)}\}$. The Lyapunov drift is opportunistically minimized, taking into account the cost of the energy consumed for mobile transmission: $\Delta_{\text{mc}}(\boldsymbol{Q(t)}) + V_{\text{mc}}\mathbb{E}\{E_{\text{com}}^{\text{Tx}}(t)|\boldsymbol{Q(t)}\}$ [15], 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 ($E_{\text{com}}^{\text{Tx}}(t)$), and

$$E_{\text{com}}^{\text{Tx}}(t) = \sum_{k=1}^{K} P_k^{\text{Tx}}(t) \sum_{i=1}^{N} I_i(t)\alpha_k(t)\tau_{i,k}^{\text{mc}}(t), \tag{6.16}$$

where all the components i, prepared for offloading at time slot t, have $I_i(t)$ set to one.

Lemma 6.1 *As proved in [38], the upper bound of the Lyapunov drift is obtained by:*

$$\begin{aligned}\Delta_{\mathrm{mc}}(\boldsymbol{Q}(t)) + V_{\mathrm{mc}}\mathbb{E}[E^{\mathrm{Tx}}_{\mathrm{com}}(t)|\boldsymbol{Q}(t)] \leq \frac{(A^{\mathrm{mc}}_{\max})^2}{2} \\ + V_{\mathrm{mc}}\mathbb{E}\{E^{\mathrm{Tx}}_{\mathrm{com}}(t)|\boldsymbol{Q}(t)\} \\ + \sum_{k=1}^{K}\mathbb{E}\left\{Q_k(t)\left(\sum_{i=1}^{N}B^{\mathrm{mc}}_i(t) - A^{\mathrm{mc}}_k(t)\right)|\boldsymbol{Q}(t)\right\}.\end{aligned} \tag{6.17}$$

where $A^{\mathrm{mc}}_k(t)$ represents data rate transmitted from the mobile device to the cloud through radio interface k at time slot t, $B^{\mathrm{mc}}_i(t)$ is the arrival data rate in the mobile transmission buffer at time slot t. Note that $B^{\mathrm{mc}}_i(t)$ will include both the data that the application needs to transfer due to offload operations of component i and other ambient data that the mobile generates and which is unrelated to the offloading. Also, $A^{\mathrm{mc}}_{\max}$ is the maximum transmitted data rate.

Following the Lyapunov optimization framework, the upper bound of the objective function in Eq. (6.17) must be minimized. This can be done by simplifying the RHS of Eq. (6.17) as follows:

$$\mathrm{OP}_{\mathrm{Tx}}: \min_{\boldsymbol{\alpha}} \Omega_{\mathrm{mc}}(t) = V_{\mathrm{mc}}E^{\mathrm{Tx}}_{\mathrm{com}} - \sum_{k=1}^{K}\left(Q_k(t)A^{\mathrm{mc}}_k(t)\right), \tag{6.18}$$

s.t.

$$\sum_{k=1}^{K}\alpha_k(t)\sum_{i=1}^{N}I_i(t)\tau^{\mathrm{mc}}_{i,k}(t) \leq T^{\mathrm{Tx}}_{\mathrm{th}}, \tag{6.19}$$

$$\sum_{k=1}^{K}\alpha_k(t) = 1, \alpha_k(t) \geq 0, \forall k, \tag{6.20}$$

where $\boldsymbol{\alpha} = [\alpha_1(t)\ \ \alpha_2(t)\ \ \ldots\alpha_K(t)]$. Constraints (6.19) and (6.20) respectively ensure that the transmission time lies below a certain threshold, $T^{\mathrm{Tx}}_{\mathrm{th}}$, 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 [38] for the transmission queue stability and energy consumed for transmission, respectively, are expressed as:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\overline{Q}_i(t) \leq \frac{\frac{(A^{\mathrm{mc}}_{\max})^2}{2} + V_{\mathrm{mc}}E^{*\mathrm{Tx}}_{\mathrm{com}}}{\varepsilon_{\max}}, \tag{6.21}$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\overline{E}_{\text{com}}^{\text{Tx}}(t)\leq\frac{(A_{\max}^{\text{mc}})^2}{2V_{\text{mc}}}+E_{\text{com}}^{*\text{Tx}},\tag{6.22}$$

where $\overline{a}$ represents the mean value for parameter a, and $E_{\text{com}}^{*\text{Tx}}$ is the optimal value of $E_{\text{com}}^{\text{Tx}}$ obtained from solving the optimization problem in Eq. (6.18).

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 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, components 6 and 11 at t_3 in the example of Fig. 6.2b). If the optimization problem does not have a feasible solution, then there are two options: (1) wait for the next time slot (e.g., components 6 and 11 wait at time t_2 in Fig. 6.2b); (2) 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 $|t-t_i^{\text{req}}|<\gamma q_i^{\text{m}}$, 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 $|t-t_i^{\text{req}}|\geq\gamma q_i^{\text{m}}$, 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:

$$Q_k(t+1)=\max[Q_k(t)-A_k^{\text{mc}}(t),0]+\alpha_k(t)\sum_{i=1}^{N}B_i^{\text{mc}}(t),\tag{6.23}$$

where the first term on the RHS of Eq. (6.23) 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. 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 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: $\tau_{i,k}^{\text{mc}}(t+1)+1$. 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.

Cloud Transmission Strategy

Just as in the case of the transmission strategy from the mobile, there is also a need to optimize the cloud transmission strategy taking into account delays and energy consumed by the mobile device for receiving this data. The percentage of data that

needs to be allocated to each wireless interface to send the necessary information from the cloud to the mobile is optimally chosen.

To ensure that no cloud Tx buffer overflows, the time-averaged summation of buffer occupancies must remain finite:

$$\overline{S} = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{E}\{|S_k(t)|\} < \infty, \tag{6.24}$$

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 $\boldsymbol{S(t)} = [S_1(t)S_2(t)\ldots S_K(t)]$. The Lyapunov drift in the data transfer from the cloud to the mobile is expressed as $\Delta_{\text{cm}}(\boldsymbol{S(t)}) \triangleq \mathbb{E}\{L(\boldsymbol{S(t+1)}) - L(\boldsymbol{S(t)})|\boldsymbol{S(t)}\}$. This Lyapunov drift is opportunistically minimized, considering the penalty of energy consumed for downlink mobile reception as $\Delta_{\text{cm}}(\boldsymbol{S(t)}) + V_{\text{cm}}\mathbb{E}\{E_{\text{com}}^{\text{Rx}}(t)|\boldsymbol{S(t)}\}$, 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 ($E_{\text{com}}^{\text{Rx}}(t)$) is applied, and

$$E_{\text{com}}^{\text{Rx}}(t) = \sum_{k=1}^{K} P_k^{\text{Rx}}(t) \sum_{i=1}^{N} \beta_k(t)\tau_{i,k}^{\text{cm}}(t), \tag{6.25}$$

Following Lemma 6.1, the upper bound of the objective function for the Lyapunov drift of data transfer from the cloud to the mobile is obtained by:

$$\begin{aligned} \Delta_{\text{cm}}(\boldsymbol{S(t)}) + V_{\text{cm}}\mathbb{E}[E_{\text{com}}^{\text{Rx}}(t)|\boldsymbol{S(t)}] &\le \frac{(A_{\text{max}}^{\text{cm}})^2}{2} \\ &+ V_{\text{cm}}\mathbb{E}\{E_{\text{com}}^{\text{Rx}}(t)|\boldsymbol{S(t)}\} \\ &+ \sum_{k=1}^{K} \mathbb{E}\{S_k(t)(\sum_{i=1}^{N} B_i^{\text{cm}}(t) - A_k^{\text{cm}}(t))|\boldsymbol{S(t)}\}, \end{aligned} \tag{6.26}$$

where $A_k^{\text{cm}}(t)$ represents the data rate transmitted from the cloud to the mobile device through radio interface k at time slot t, $B_i^{\text{cm}}(t)$ is the arrival data rate in the cloud transmission buffer at time slot t. $B_i^{\text{cm}}(t)$ will include both the data that the cloud transfers due to offload operations of component i and other ambient data. Also, $A_{\text{max}}^{\text{cm}}$ is the maximum data rate from the cloud. After simplifying the upper bound on the RHS of Eq. (6.26), the objective functionof 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:

$$\text{OP}_{\text{Rx:}} \min_{\boldsymbol{\beta}} \Omega_{\text{cm}}(t) = V_{\text{cm}} E_{\text{com}}^{\text{Rx}} - \sum_{k=1}^{K} \left(S_k(t) A_k^{\text{cm}}(t) \right), \tag{6.27}$$

s.t.

$$\sum_{k=1}^{K} \beta_k(t) \sum_{i=1}^{N} \tau_{i,k}^{\text{cm}}(t) \leq T_{\text{th}}^{\text{Rx}}, \tag{6.28}$$

$$\sum_{k=1}^{K} \beta_k(t) = 1, \beta_k(t) \geq 0, \forall k, \tag{6.29}$$

where $\boldsymbol{\beta} = [\beta_1(t) \ \beta_2(t) \ \ldots \beta_K(t)]$. Constraints (6.28) and (6.29), respectively, ensure that the latency from the cloud to the mobile lies below a certain threshold, $T_{\text{th}}^{\text{Rx}}$, 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:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \overline{S}_i(t) \leq \frac{\frac{(A_{\text{max}}^{\text{cm}})^2}{2} + V_{\text{cm}} E_{\text{com}}^{*\text{Rx}}}{\varepsilon_{\text{max}}}, \tag{6.30}$$

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \overline{E}_{\text{com}}^{\text{Rx}}(t) \leq \frac{(A_{\text{max}}^{\text{cm}})^2}{2V_{\text{cm}}} + E_{\text{com}}^{*\text{Rx}}. \tag{6.31}$$

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 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. 6.2b). If the optimization problem 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: $\tau_{i,k}^{\text{cm}}(t+1) + 1$. 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 ($\forall k$) are updated as follows:

Algorithm 6.1 Cognitive offloading and scheduling heuristic

Offline stage at t_0 (optional):
Solve OP_{off} to calculate which components can be offloaded
otherwise all components are analyzed for offloading (except 1, N)
Online stage:
repeat $t \rightarrow t+1$
    Check scheduling constraints given by Eqs. (6.12) and (6.13)
    For offloadable components:
    Solve OP_{Tx}
    if OP_{Tx} has feasible solution **then**
        offload related components with the corresponding values for $\alpha_k(t)\ \forall k$ obtained by OP_{Tx}
    else if $|t - t_i^{\text{req}}| < \gamma q_i^{\text{m}}$ **then**
        wait for the next time slot to check offloading
    else
        component i will be scheduled for local execution
    end if
    Solve OP_{Rx}
    if OP_{Rx} has feasible solutions **then**
        Send output data from the cloud with corresponding values for $\beta_k(t)\ \forall k$ obtained by OP_{Rx}
    else
        Do not send data from cloud and wait for the next slot
    end if
    For the components scheduled for local execution:
    if local execution constraint given by Eq. (6.33) is satisfied **then**
        Execute component i locally at time slot t
    end if
    Update $Q_k(t+1)\ \forall k$ using Eq. (6.23)
    Update $S_k(t+1)\ \forall k$ using Eq. (6.32)
    Add delay $(\tau_{i,k}^{\text{mc}}(t+1)\ \forall k)$ for waiting components in the mobile
    Add delay $(\tau_{i,k}^{\text{cm}}(t+1)\ \forall k)$ for waiting components in the cloud
until $t = T$.

$$S_k(t+1) = \max[S_k(t) - A_k^{\text{cm}}(t), 0] + \beta_k(t) \sum_{i=1}^{N} B_i^{\text{cm}}(t). \tag{6.32}$$

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. 6.2b). 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, there is a need to ensure that no other application's component is currently running on the mobile. This is expressed as:

$$\sum_{i=1}^{N} \sum_{s=t-1-q_i^{\mathrm{m}}}^{t-1} X_i(s) < 1. \tag{6.33}$$

6.3 Comparison with the State of the Art

Simulations have been performed using the real data measurements from an HTC phone running multi-component applications, using Amazon EC2 as the cloud and two radio interfaces, LTE and WiFi, for cognitive offloading. Results show that the cognitive approach provides 7% higher net utility in comparison to a simple use of multiple radio interfaces due to selection of the best interface to the use for data transfer. The scalability of the heuristic approach is further analyzed using various real delay values, communication power levels, sizes of component dependency graphs, and energy-delay trade-off factors. More details of the simulation results are elaborated in Chap. 6 while in this section a gist of comparison analysis with the state of the art is discussed.

In Fig. 6.3, the total energy consumption of the CSCO approach is compared with three other schemes: (1) the scheme where all the components are executed locally; (2) the scheme where all the components are offloaded for remote execution; and (3) the dynamic offloading algorithm (DOA) proposed in [19], which uses ON-OFF multi-RAT strategy. Moreover, the variants of the introduced approach are considered as: (1) exhaustive search, where the optimization problem is solved using brute force. This gives an upper bound of performance for all algorithms; (2) heuristic with no offline stage (H-1Stage), where this strategy eliminates the offline stage and proceeds with the rest of the heuristic algorithm where all components are eligible for offloading; (3) two-stage heuristic algorithm (H-2Stage), where the preprocessing stage is used to eliminate some of the components from being considered for offloading; (4) single stage heuristic under On-Off model for the wireless interfaces (H-1S-OnOff), where the single stage heuristic algorithm is used, and all components are considered for offloading at the online stage, but the components are offloaded using only one best wireless interface; (5) two-stage heuristics with ON/OFF wireless interfaces (H-2S-OnOff), where the two-stage algorithm is used for offloading, but the one best wireless interface policy is used for offloading. Please note that these variants of the heuristic algorithms are more elaborated in Chap. 7 as well.

In the simulations for Fig. 6.3, a face recognition application with 10 sequential components was utilized (http://darnok.org/programming/face-recognition/). Considering radio interfaces WiFi and 3G for this bar graph, the wireless network parameters in http://www.3gpp.org/ftp/tsg-ran/wg4-radio/ are used such that exactly the same parameters used for the simulation of DOA in [19] were repeated for all the other seven schemes. This comparison is normalized to the scheme with local execution of all the components. It is observed that H-1Stage consumes 73%, 51%, 28%, and 3% less energy in comparison to the schemes using local

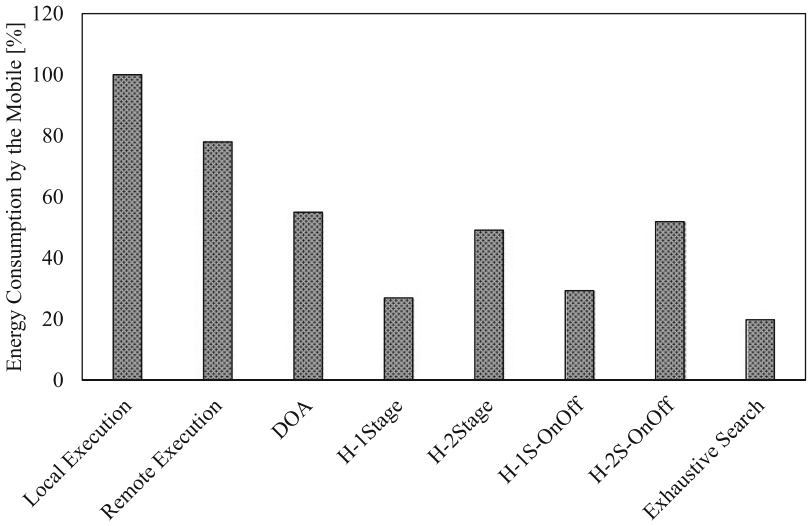


Fig. 6.3 Total energy consumed by the mobile device for the heuristic and classical schemes, normalized to the energy consumed by local execution (using the face recognition application in http://darnok.org/programming/face-recognition/)

execution, remote execution, DOA, and H-1S-OnOff, respectively. Also, H-1Stage consumes 8% more energy in comparison to the best case scenario that is obtained by the exhaustive search. We can also compare the performances of the schemes that use offline stage (H-2Stage, H-1S-OnOff, and DOA) in one category. 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 non-time adaptive 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.

Chapter 7
Evaluation of Cloud Offloading and Scheduling Mechanisms in Different Scenarios

To understand the impact of the schemes introduced in the previous chapters various evaluations and comparisons with classic approaches were done. While in each chapter a summary of the performance of the scheme described in that chapter was given, in this chapter we go into further details. We can think of the scheduling and computation offloading schemes in the following scenarios: (1) joint scheduling and cloud offloading for single-radio enabled mobile devices; (2) cognitive offloading using multiple radios; (3) optimal cognitive offloading and scheduling using multiple radios; and (4) time-adaptive cognitive offloading and scheduling using multiple radios.

The first and second scenarios imply two different dimensions of computational cloud offloading for mobile applications. In Sect. 7.2, the proposed scheme in scenario (1) (joint scheduling and cloud offloading (JSCO)) using only one radio is compared to five recent classic schemes. Scenario (2) is discussed in Chap. 3 and Sects. 7.4 and 7.5 show that these schemes outperform the classic schemes as well. Section 7.4 compares all the schemes with one another. It is seen that the mobile device consumes least energy when using the optimal scheme discussed in Chap. 5, the next most efficient scheme was the heuristic cognitive cloud offloader strategy discussed in Chap. 6, followed by the JSCO method using the best of the multiple RATs discussed in Chap. 4. The least efficient performance was observed when using the classic remote offloading scheme using the best RAT at any given time.

7.1 Evaluation Setup

An HTC Vivid smartphone with a 1.2 GHz dual-core processor was used to gather real data. This phone is equipped with two radio interfaces ($k = 2$): WiFi and LTE (note that in single-radio scenario, only WiFi is activated). Moreover, we

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_7

assume that whereas LTE is always available, the WiFi interface can sometimes be unavailable (as is common in real life scenarios). A multi-component video navigation application was used where video processing, face detection, graphics, and clustering were the main features. In all, 14 components were used, four of which are related to the graphics feature, three are for the face detection feature, six are for video processing, and one is for clustering. Note that the first and last components are executed locally so that the input–output of the application is accessed by the mobile user. In addition, graphics library tools from the OpenGL mobile Android applications were used (http://www.opengl.org/, March 2014); face detection was taken from http://www.developer.com/ws/android/programming/face-detection-with-android-apis.html, July 2014; and all the video processing features were obtained from http://opencv.org/, April 2014. The CDG of this application is illustrated in Fig. 3.1a. The execution times of the components in the HTC phone and the cloud, uplink and downlink rates, delay at the WiFi interface were measured. The Amazon elastic compute cloud (Amazon EC2) was used as the cloud computing server (http://aws.amazon.com/ec2/. July 2014). The average transmission and reception power levels of the mobile device for WiFi service were 257.83 and 123.74 mW, respectively. The active and idle power levels of the phone were 644.9 and 22 mW, respectively. The power consumption of the last component in the mobile device was 55 mW. These power measurements were obtained using the "CurrentWidget: Battery monitor" application (http://code.google.com/p/currentwidget/. July 2014). The average wireless service rates for WiFi were 0.80 Mbps for the uplink transmission and 1.76 Mbps for the downlink transmission, respectively. The wireless service delays and data rates in both uplink and downlink through WiFi and LTE (indoor and outdoor environments) were obtained by using the Android secure FTP tool (http://www.lysesoft.com/). Figure 7.12 shows the uplink and downlink delay values in indoor and outdoor environments for WiFi and LTE wireless radios for a range of data size from 10 kB to 105 MB of data. As can be seen from Fig. 7.12, delay in outdoor wireless is even lower than indoor wireless environment for some ranges while indoor wireless delay on average is lower. The local execution time for the 14 components was measured as [30, 340, 345, 125, 30, 80, 70, 30, 185, 125, 650, 571, 904, 56] ms. Because processing of the components in the mobile device is performed in serial, application runtime in the local execution equals the sum of the processing times for the 14 components (3541 ms).

7.2 Joint Scheduling and Cloud Offloading for Single-Radio Enabled Mobile Devices

In this section, first the performance of joint scheduling and cloud offloading schemes is discussed for single-radio enabled devices based on real data measurements from the 14-component application whose CDG was presented in Fig. 3.1a.

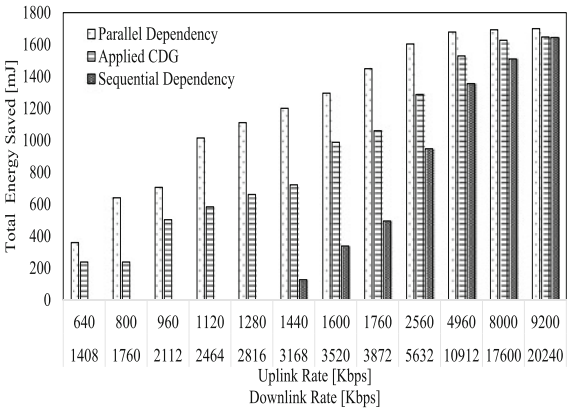
To further understanding of the model's adaptability and scalability, some randomly generated CDGs are considered whose layered structure and Fan-in/Fan-out ratio could be controlled.

7.2.1 Simulations for the Video Navigation Mobile Application

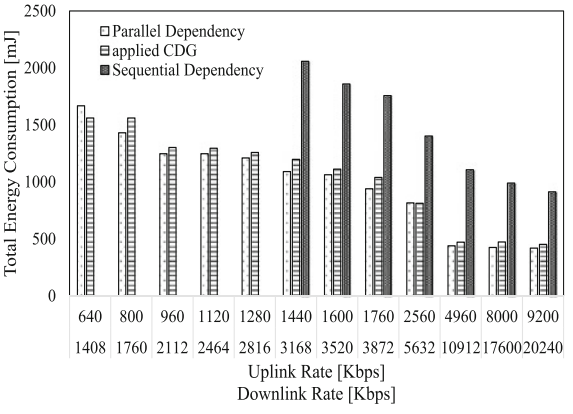
The performance of the strategy is tested on the real 14-component application with arbitrary dependencies (Fig. 3.1) against a 14-component application with fully parallel dependencies (Fig. 2.2b) and a 14-component application with fully sequential dependencies (Fig. 2.2a). Since the parallel and sequential dependency graphs show, respectively, the lowest and highest dependencies between components, lower and upper bounds for the cost of offloading could be obtained for the applied CDG.

Rate Plots

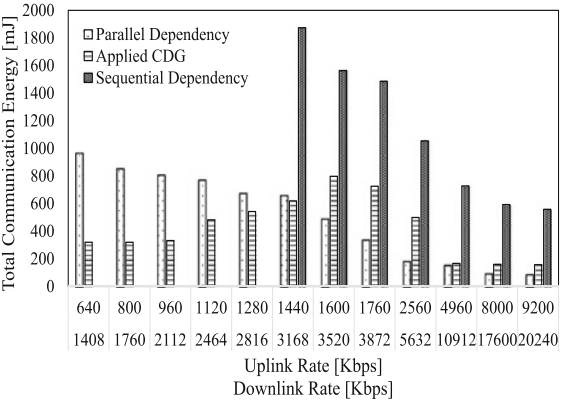
Figure 7.1 shows the total energy values for several uplink and downlink rates of the WiFi interface provided for cloud offloading. Figure 7.1a presents the total energy saved through remote execution (the objective function in Eq. (3.10)) versus wireless rates. We see that while rates increase, more energy is saved by the mobile device with cloud offloading. This is expected, because with higher rates, data communication is no longer a bottleneck and it is more energy efficient to offload as many components as possible to the cloud. More energy is saved in the parallel dependency graph, while less energy is saved in the sequential dependency graph. We observe that in the sequential dependency graph, no energy can be saved by cloud offloading for lower ranges of rates, and the application cannot be processed with these low rates in 3 s (the time for local execution is 3541 ms). However, in the higher ranges of rates (uplink (downlink) rate= 9200 (20,240 kbps)), most of the components are offloaded to the cloud for computations in all three CDGs. Thus, the performances of these three are closer to each other when the wireless rates increase. In Fig. 7.1b, the total energy consumed by the mobile device (summation of active energy while the mobile device is executing components locally, communication energy, and idle energy while the mobile's processor is not executing any component) is plotted. We see that less total energy is consumed by the mobile device when WiFi rates increase. Moreover, Fig. 7.1c illustrates the energy consumed by communication, E_{com} (given in Eq. (3.9)), versus wireless rates. It is observed that the energy consumed by communication decreases with an increase in rates for the sequential and parallel dependency graphs. Although this is true for the applied CDG in higher rate ranges, more energy is consumed by offloading while rates increase in the lower ranges. The reason is that more computations are offloaded when rates increase so more energy is required for offloading, while in the higher ranges of rates, the time to offload decreases thereby decreasing the communication energy. Note that the application with sequential



(a)



(b)



(c)

Fig. 7.1 Total energy for the 14-component application versus uplink and downlink rates in WiFi while $T = 3$ s, $P_{\text{Tx}} = 257.83$ mW, $P_{\text{Rx}} = 123.74$ mW. (**a**) Total energy saved. (**b**) Total energy consumption. (**c**) Total communication energy

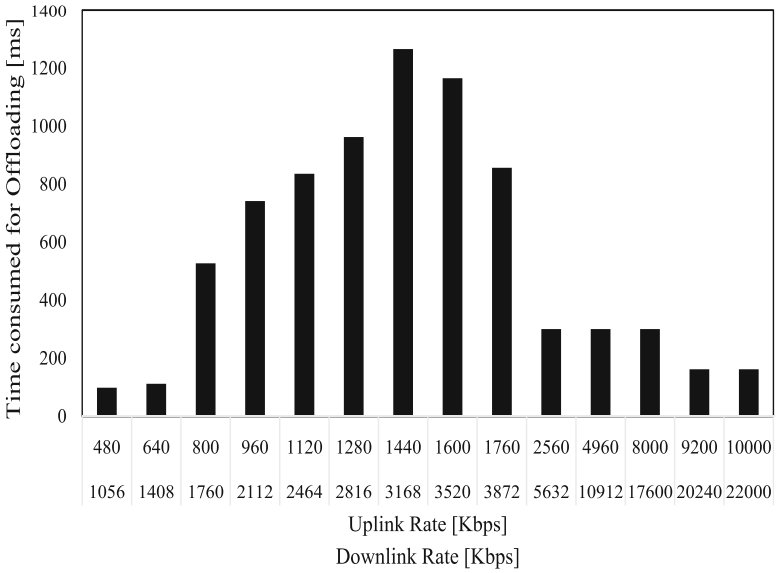


Fig. 7.2 Time consumed for offloading versus rates of the WiFi link while $T = 3$ s, $P_{\text{Tx}} = 257.83$ mW, $P_{\text{Rx}} = 123.74$ mW

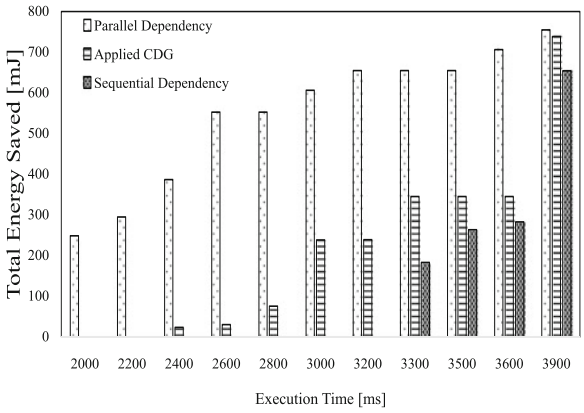
dependency cannot be executed until rates reach 1440/3168 kbps. In the lower rate ranges, wireless delay is high, and offloading is not preferred. On the other hand, local execution takes 3541 ms when the application deadline, T, is set to 3000 ms in the simulations for this figure. Therefore, the scheme using sequential dependency graph is not plotted at lower rates because the application cannot be executed in $T = 3000$ ms.

Figure 7.2 depicts the time span for communication versus uplink and downlink rates for the applied CDG. More components for offloading lead to the consumption of more energy and time for offloading, as shown in Figs. 7.1c and 7.2, respectively.

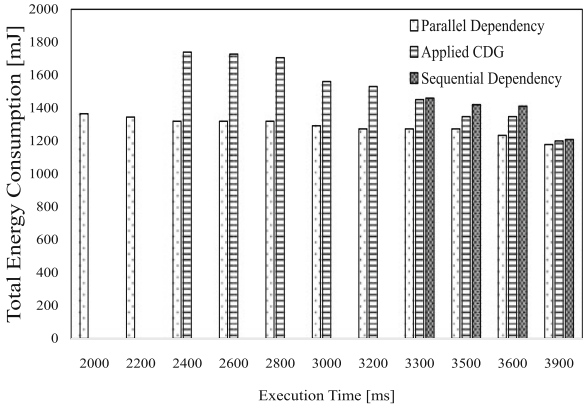
Time Plots

Figure 7.3a, b, c respectively plots the total energy saved, total energy consumption, and the energy consumed by communication versus execution time of the application (T) for the three different CDGs considered—sequential, applied, and parallel. When more time is allotted for the execution of the application, cloud offloading is preferred and leads to a decrease in energy expenditure by the mobile device.

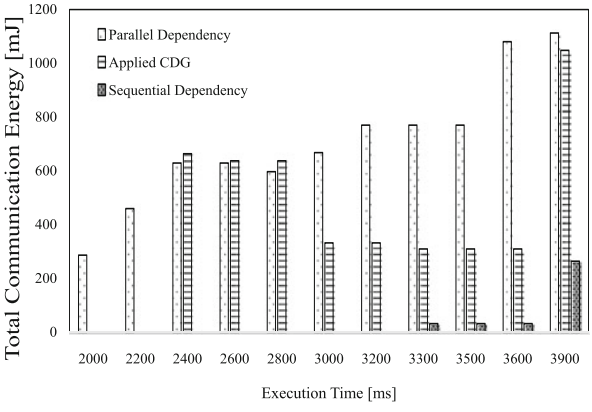
Figures 7.1 and 7.3 show that using the JSCO scheme (the scenario where the applied CDG is used) works better than using an optimal offloading scheme that uses a compiler pre-determined sequential traversal of an arbitrary CDG (the scenario where the sequential dependency is used). Examples of sequential traversals of arbitrary CDGs include [10, 23, 49]. Specifically, we see from Fig. 7.3 that the



(a)



(b)



(c)

Fig. 7.3 Total energy versus execution time (T) while $R_u = 0.8$ Mbps and $R_d = 1.76$ Mbps. (**a**) Total energy saved. (**b**) Total energy consumption. (**c**) Total communication energy

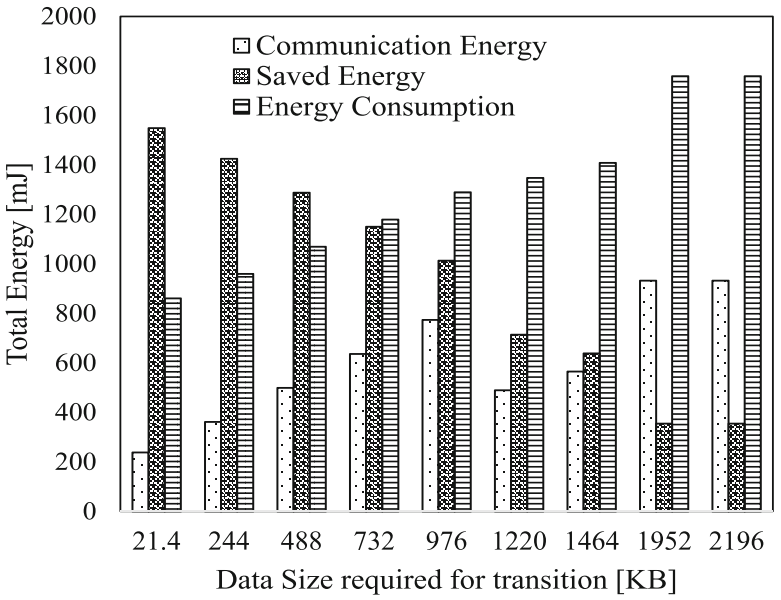


Fig. 7.4 Total energy versus size of data transferred, when $T = 3$ s, $R_u = 0.8$ Mbps and $R_d = 1.76$ Mbps using the applied CDG

processing of an application with sequential traversal CDG can be completed in no less than 3300 ms, while the application with applied CDG can be processed in 2400 ms and the application with parallel CDG can be processed in 2000 ms (rates are set to 800/1760 kbps). In addition, the application with sequential dependency cannot be executed until rates reach 1440/3168 kbps, whereas the applied CDG is processed at much lower rates, 640/1408 kbps, while T is set to 3000 ms (Fig. 7.1).

The Data Plot

We next look at the impact on energy consumption and savings when the amount of data to be transferred increases. Here the required data transfer for face detection components is increased from 21.4 kB to 2.2 MB to consider the performance of total energy as a function of the data size required for transition. In Fig. 7.15, we see that, as expected, while the size of the transferred data increases, more energy is consumed for communication, less energy is saved, and more energy is consumed by the mobile device (Fig.7.4).

7.2.2 Simulations for Variety of Component Dependencies

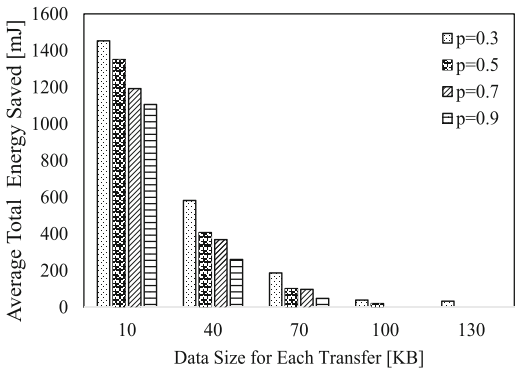
So far, the system performance was analyzed based on the fixed CDG from the 14-component video navigation application shown in Fig. 3.1, as well as the two extreme cases of fully sequential CDG and fully parallel CDG. In this section, the performance of the system is analyzed based on two different categories of random CDGs: (1) Layer-by-Layer, and (2) Fan-in/Fan-out.

In Layer-by-Layer CDGs, a random number of nodes are generated for each of the layers and edges are added with a probability p going from a node in an earlier layer to a node in one of the successive layers. In Fan-in/Fan-out CDGs, the Fan-in/Fan-out ratio of each node is constrained to the given threshold. Since usually mobile-initiated applications must start on the mobile device and have an output display on the mobile device, the first and last components are processed in the mobile device. Note that the parallel and sequential dependency graphs show the lowest and highest dependencies between components respectively and can be used to obtain the lower and upper ranges for the cost of offloading on applications exhibiting these extremes of CDGs. Since random CDGs are used in this subsection, the simulations for each data point are run over three CDGs and the average of these three values is plotted. Each CDG generated is constrained to have only 14 components for comparison purposes.

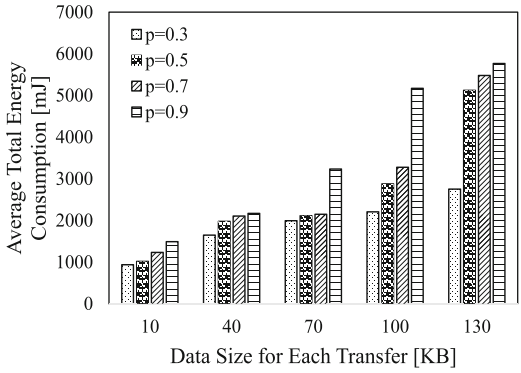
Figures 7.5 and 7.6 show the performance of the JSCO scheme for randomly generated Layer-by-Layer CDGs. In Fig. 7.5, the average total energy saved through remote execution, the average total energy consumed by the mobile, and the average total energy for communication are plotted against the size of data transferred. These bar graphs are compared as a function of the probability of edge connections (p). When this probability increases, more components are dependent on each other, and the density of the CDG increases. Therefore, the energy consumed by cloud offloading increases (Fig. 7.5c), and the energy saved through remote execution decreases (Fig. 7.5a). Moreover, it can be observed that when data size for transferring the components increases, the total energy consumed by the mobile device and the energy consumed for communication increase (Fig. 7.5b, c), and the energy saved through remote execution decreases (Fig. 7.5a). Also note that for high values of p and size of data transfer, the energy costs of offloading are so high that the energy saved through remote execution gets closer to zero (as shown in Fig. 7.5a).

In Fig. 7.6, the average total energy saved through remote execution, the average total energy consumed by the mobile, and the average total energy for communication are plotted against uplink and downlink rates. These bar graphs are also compared as a function of the probability of edge connections. We can observe that while the wireless rates increase, the energy consumed by offloading increases (Fig. 7.6b), the energy saved through cloud offloading decreases (Fig. 7.6a), and the total energy consumption decreases (Fig. 7.6b). Moreover, we see that while the probability and rates increase, the energy saved through remote execution decreases.

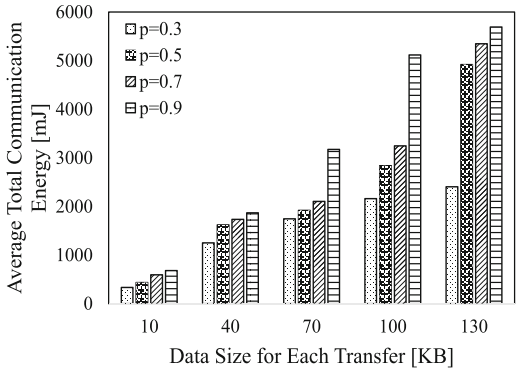
Figures 7.7 and 7.8 show the performance of the JSCO scheme for randomly generated Fan-in/Fan-out CDGs. In Fig. 7.7, energy saved, energy consumed by the mobile, and energy consumed for communication are respectively plotted versus



(a)



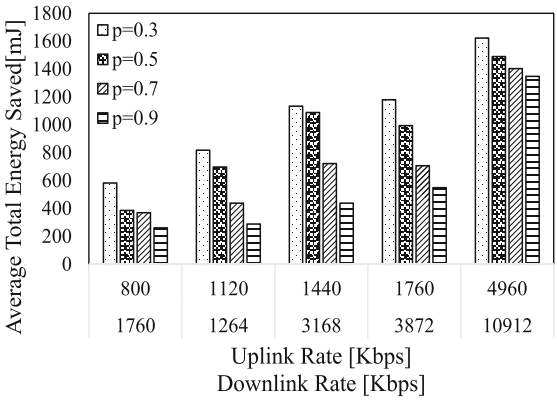
(b)



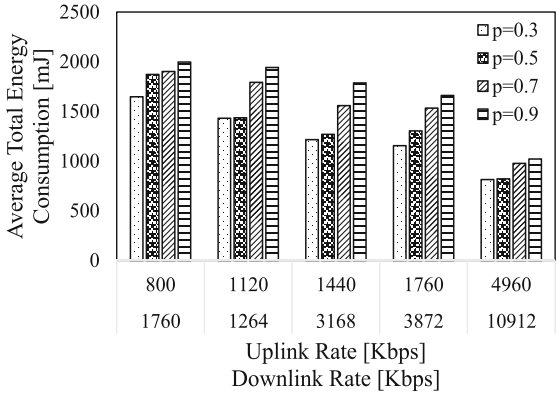
(c)

Fig. 7.5 Average total energy versus required data size for transferring each component in the apps with Layer-by-Layer CDG ($s = 5$) and 14 components while $T = 3$ s, $P_{\text{Tx}} = 257.83$ mW, $P_{\text{Rx}} = 123.74$ mW. (**a**) Average total energy saved. (**b**) Average total energy consumption. (**c**) Average total communication energy

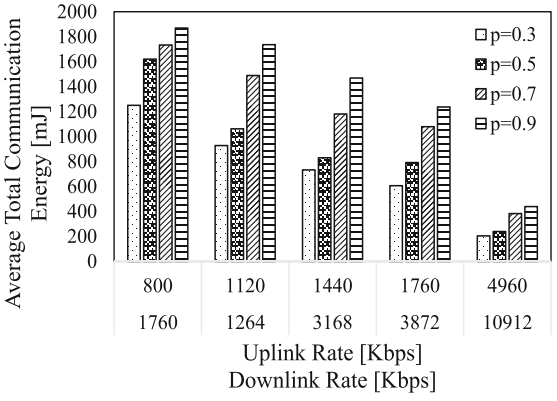
the average data size for each transfer. Our results indicate that the performance of the JSCO scheme is independent of the Fan-in/Fan-out ratio of these graphs but dependent on the total Fan-in plus Fan-out degrees. When the in+out degree



(a)

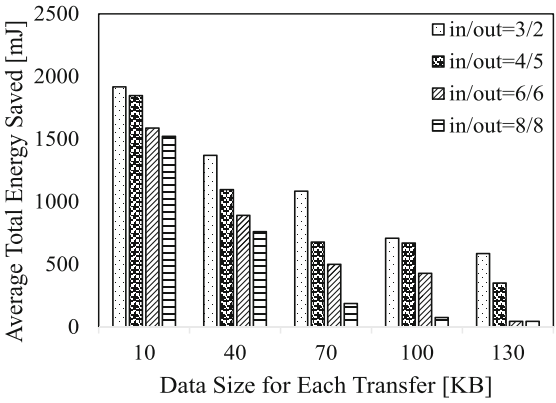


(b)

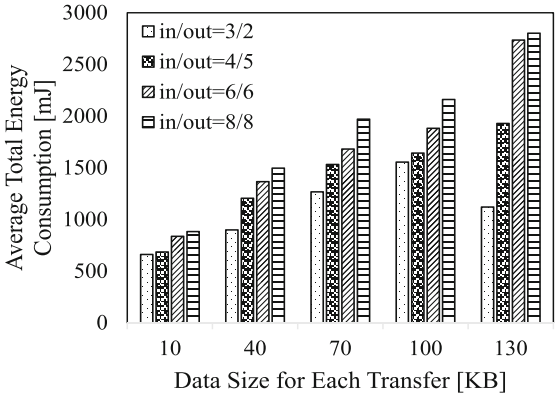


(c)

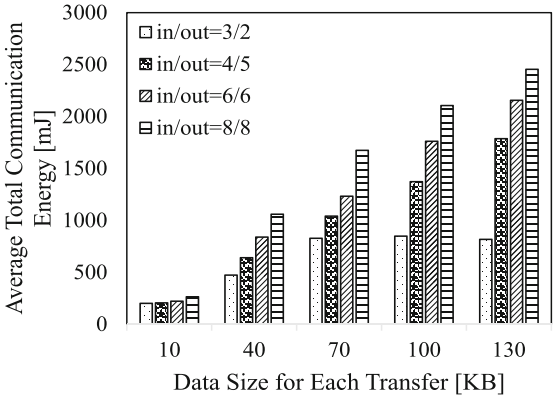
Fig. 7.6 Average total energy versus uplink and downlink rates in WiFi for the apps with Layer-by-Layer CDG ($s = 5$) and 14 components while $T = 3$ s, $P_{\text{Tx}} = 257.83$ mW, $P_{\text{Rx}} = 123.74$ mW. (**a**) Average total energy saved. (**b**) Average total energy consumption. (**c**) Average total communication energy



(a)

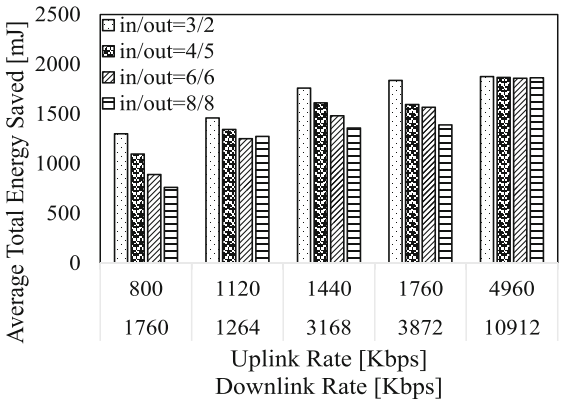


(b)

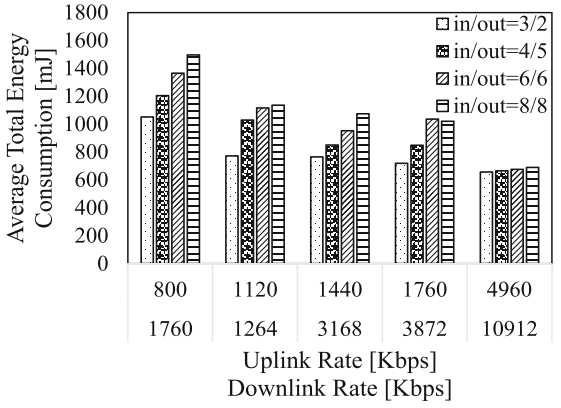


(c)

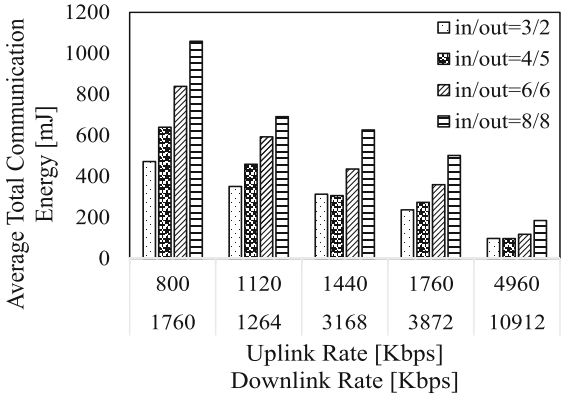
Fig. 7.7 Average total energy versus required data size for transferring each component in the apps with Fan-in/Fan-out CDGs and 14 components while $T = 3$ s, $P_{\text{Tx}} = 257.83$ mW, $P_{\text{Rx}} = 123.74$ mW. (**a**) Average total energy saved. (**b**) Average total energy consumption. (**c**) Average total communication energy



(a)



(b)



(c)

Fig. 7.8 Average total energy versus uplink and downlink rates in WiFi for the apps with Fan-in/Fan-out CDGs and 14 components while $T = 3$ s, $P_{Tx} = 257.83$ mW, $P_{Rx} = 123.74$ mW. (**a**) Average total energy saved. (**b**) Average total energy consumption. (**c**) Average total communication energy

increases, the dependency and offloading costs increase such that the energy saved through cloud offloading decreases and the energy consumed by the mobile device increases (Fig. 7.7a, b). Also when the data size for transferring increases, the energy consumed by the mobile increases (Fig. 7.7c). Figure 7.8 presents the energy as a function of the uplink/downlink rates. While rates increase and the in+out degree decreases, the energy consumed for communication decreases (Fig. 7.8c). Therefore, the energy saved through remote execution increases (Fig. 7.8a), and the energy consumed by the mobile decreases (Fig. 7.8c).

7.2.3 *Scalability of the JSCO Scheme*

In this subsection, the scalability of the JSCO scheme is discussed. Specifically, we want to address the largest application that the JSCO scheme can handle in terms of the number of components and total execution time. In the discussions so far, only a 14-component application (either real or randomly generated) is used. In order to maintain the same probability distribution of the measurements when scaling up the application, the histogram of the current real data measurements (q_k^{m}, q_k^{c}, P_{ac} $\forall k$) is calculated from the 14-component video navigation application. Using the obtained distribution, the new data is generated for applications with a greater number of components (25, 45, 65, 85, and 105 components). Increasing the number of components requires a corresponding increase in the runtime deadline (T); for example, for a 25-component application $T = 6500$ ms; for $N = 45$, $T = 12{,}000$ ms; for $N = 65$, $T = 15{,}000$ ms; for $N = 85$, $T = 30{,}000$ ms; and for $N = 105$, $T = 100{,}000$ ms.

Three scenarios are considered in this part: (A) the scenario where the average data size to transfer is fixed at 1220 kB and the uplink/downlink rate is fixed at 1.28/2.816 Mbps; (B) the scenario where the average data size to transfer is fixed at 1220 kB (the same as A) and the uplink/downlink rate is fixed at 4.96/10.912 Mbps (more than A); and (C) the scenario where the average data size to transfer is fixed at 2196 kB (more than A) and the uplink/downlink rate is fixed at 1.28/2.816 Mbps (the same as A).

Table 7.1 shows the program runtimes using the discussed scheme for the two types of randomly generated CDGs—Layer-by-Layer and Fan-in/Fan-out. In this

Table 7.1 Program runtimes of the CPLEX optimizer for the discussed LP using Layer-by-Layer and Fan-in/Fan-out CDGs

N	Mobile-only execution time [ms]	T [ms]	Runtime for Layer-by-Layer [s]	Runtime for Fan-in/Fan-out [s]
25	7714	6500	561	439
45	16,230	13,500	924	834
65	17,412	14,250	1764	1649
85	27,877	17,100	2862	2700
105	28,098	21,000	7654	8647

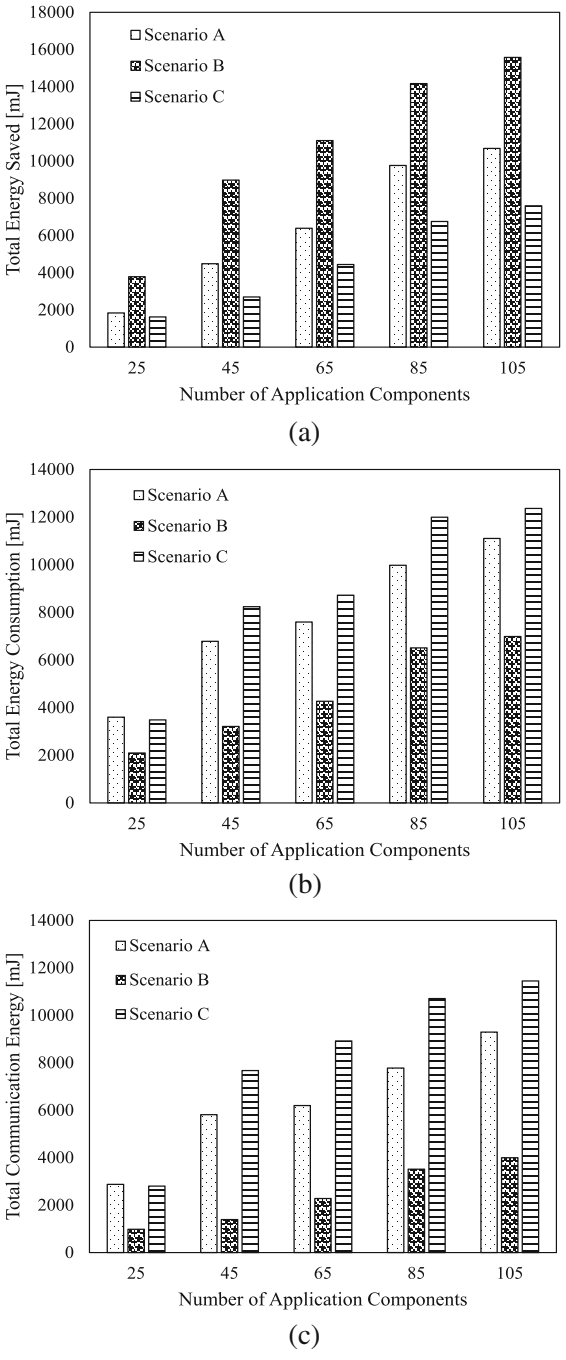


Fig. 7.9 Total energy versus the number of application components with Layer-by-Layer CDG ($p = 0.2$ and $s = 5$), presented in scenarios A, B, and C. (**a**) Total energy saved. (**b**) Total energy consumption. (**c**) Total communication energy

table, the total execution time is considered in accordance with the number of components (N). We see that while the number of components and total execution time increase, the runtime of the JSCO scheme increases. The JSCO scheme is capable of handling over 100 components with a mobile-only execution time of 28 s. The simulations were done on a single server machine with an Intel Xeon(R) E7340 processor @ 2.5 GHz CPU and 60 GB of RAM. Although the runtime to solve the associated integer linear program increases with the number of components to over 2 h, this time can be reduced through parallel implementation using more powerful processors.

In Fig. 7.9, the Layer-by-Layer CDG with a larger number of components is considered. In this figure, the energy saved, total energy consumed, and energy consumed for communication are respectively shown as a function of the number of application components for the three scenarios, *A, B, C*. Note that here, $p = 0.2$ and $s = 5$ (which is the number of layers). When the number of application components increases, the edges between components increase and therefore the costs of offloading increase. Thus, all the energy values increase. We can see that while the rates increase in Scenario *B* in comparison to Scenario *A*, the energy saved through remote execution increases, energy consumed for offloading decreases, and the total energy consumed by the mobile device decreases. On the other hand, while the data size increases in Scenario *C* in comparison to Scenario *A*, the energy saved decreases, communication energy increases, and the total energy consumed by the mobile device also increases, which is all as expected.

7.3 Cognitive Offloading Using Multiple Radios

This section investigates the performance of cognitive cloud offloading while there is a forced sequential ordering for multi-radio enabled devices based on real data measurements from the 14-component application.

Figure 7.10 plots the energy-execution time trade-off in the scheme in comparison to the local and remote execution, while the scheme takes advantage of three scenarios for radio resources: (1) WiFi and LTE are used jointly; (2) only WiFi is used for offloading; and (3) only LTE is used for offloading. The four points in the plot show local and remote executions by using only LTE, only WiFi, or both. We see that although remote execution by using LTE consumes much more energy in comparison to the others, the execution time for this scenario would be less than the others. Thus, there is a trade-off between energy consumption and execution time of the application which is relied on the delay of offloading. On the other hand by using the offloading scheme lesser energy is consumed with reasonable value for execution time. When the execution time deadlines are longer, there is more flexibility in offloading jobs to the cloud and hence energy consumptions for the mobile device reduced. Also, it is clear that joint use of radio resources gives less energy consumption and requires less execution time.

Figure 7.11 plots the percentage of data stream to the cloud through WiFi (radio interface 1) versus RTT of the WiFi and LTE in the mentioned scheme. We observe that by increase of RTT in WiFi for the range of 40–160 ms, less data stream is allocated to WiFi and more data stream is allocated through LTE for computation offloading. On the other hand, when RTT of LTE increases in the range of 50–200 ms, more data stream is allocated to WiFi and less data is allocated to LTE.

7.4 Optimal Cognitive Offloading and Scheduling Using Multi-Radios

In this section, several versions of the CSCO are introduced, evaluated, and compared with five schemes, in both indoor and outdoor wireless environments. NSFCloud (https://www.chameleoncloud.org/nsf-cloud-workshop/) was used as the cloud computing server where remote execution of sophisticated multi-component applications is processed. This platform supports both experiments involving cloud computing architectures and experiments involving mobile cloud applications (http://www.nsf.gov/pubs/2013/nsf13602/nsf13602.htm). Although the analysis is

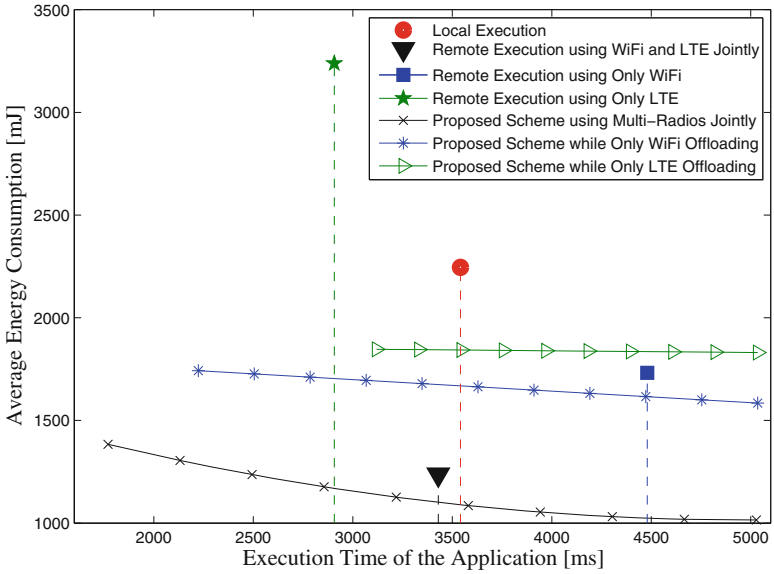


Fig. 7.10 Average energy consumption versus execution time of the application. Jointly, it shows the trade-off between the cost of energy consumption and execution time in the scheme. We observe that the application can be executed in half of the time (0.52) it takes to be executed in the cloud with the cost of 12% more energy consumption, and also the application can be executed with 20% more energy saving in comparison to the remote execution with the cost of 42% execution time extension in comparison to remote execution

developed for a general, K, radio interfaces, the experiments in this section were conducted using LTE and WiFi ($K = 2$) in both indoor and outdoor environments. For Figs. 7.12, 7.13, 7.14, 7.15, 7.16, and 7.17, the 14-component video navigation mobile application ($N = 14$) discussed in earlier chapters was used. To recap, four components of this application involved graphic feature processing, three involved face detection and recognition, six involved video processing, and the last component involved clustering the video points and output functions. Figure 2.2c shows the series-parallel CDG for this real application.

For Figs. 7.18 and 7.19, simulations were conducted with the same mobile application using varieties of CDG topologies including sequential, parallel, series-parallel (CDG shown in Fig. 2.2c), random Layer-by-Layer, and random Fan-in/Fan-out (the structure of two random CDGs is discussed in [9]). Finally in Figs. 5.3 and 5.4, the 10-face recognition application in http://darnok.org/programming/face-recognition/ was used.

The wireless service delays and data rates in both uplink and downlink through WiFi and LTE (indoor and outdoor environments) were obtained by using the Android secure FTP tool (http://www.lysesoft.com/). Please note that the secure FTP is used for only data transferring of components for computations between the mobile device and the cloud. Figure 7.12 shows the uplink and downlink delay values in indoor and outdoor environments for WiFi and LTE wireless radios for a range of data size from 10 kB to 105 MB of data. This range of data size presents

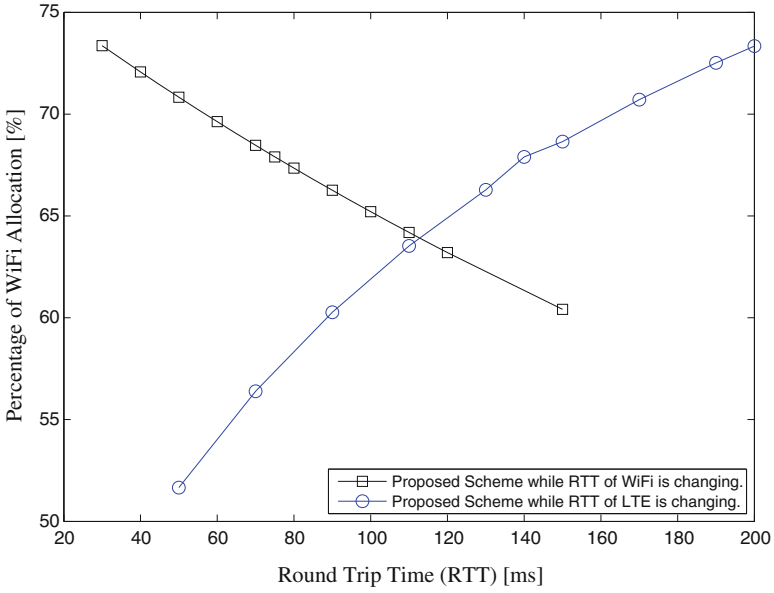


Fig. 7.11 Percentage of WiFi allocation in the iterative scheme versus round trip time (RTT) of WiFi and LTE

the average size of data transferred relied on each other for running the video application. As can be seen from Fig. 7.12, delay in outdoor wireless is even lower than indoor wireless environment for some ranges while indoor wireless delay on average is lower. This figure shows the dynamics of wireless networks for LTE and WiFi in both indoor and outdoor scenarios where the strategies are tested in the figures of this section.

A Poisson distributed background arrival data is added in the mobile device transmission buffer to simulate included ambient traffic besides the mobile application [45]. The average transmission and reception power of the mobile device for WiFi and LTE are respectively 257.83 ($\frac{1}{T}\sum_{t=1}^{T} P_k^{\text{tx}}(t)$ using WiFi radio ($k = 1$)), 123.74 mW ($\frac{1}{T}\sum_{t=1}^{T} P_k^{\text{rx}}(t)$ using WiFi radio ($k = 1$)), 356.1 ($\frac{1}{T}\sum_{t=1}^{T} P_k^{\text{tx}}(t)$ using LTE ($k = 2$)) and 197.1 mW ($\frac{1}{T}\sum_{t=1}^{T} P_k^{\text{rx}}(t)$ using LTE ($k = 2$)). The active power (P_{ac}) and idle power (P_{idle}) of the mobile device are respectively

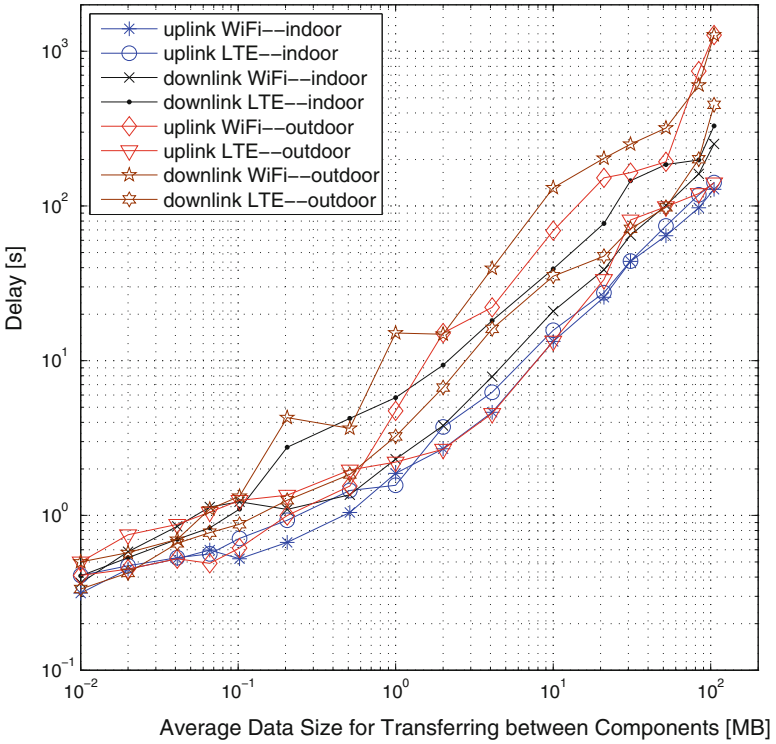


Fig. 7.12 Uplink/Downlink delay of WiFi and LTE radio interfaces in indoor/outdoor wireless environment versus average data size required for transferring data related to the mobile video navigation applications between HTC smartphone and the NSFCloud server. The average data size for transferring between components shows the average of required data size for each individual transfer of component tasks between the mobile device and the cloud

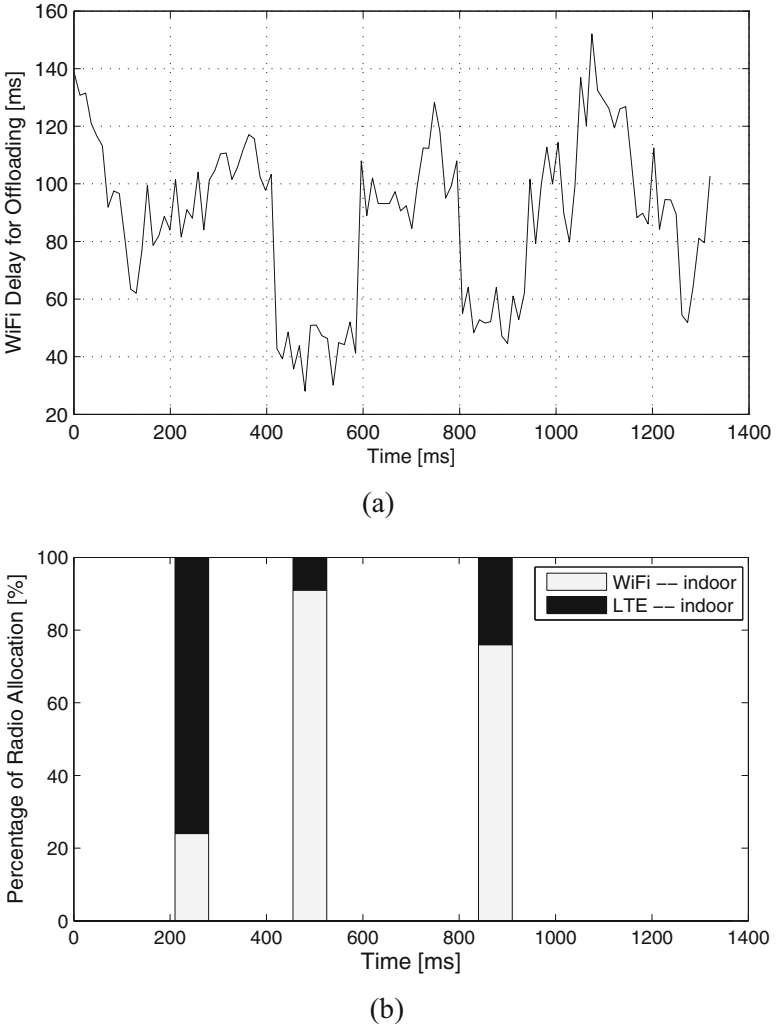


Fig. 7.13 Illustration of time-adaptivity of the CSCO strategy. (**a**) Uplink indoor WiFi delay versus time. (**b**) Percentage of radio allocation versus time using CSCO scheme

644.9 and 22 mW. The power measurements were obtained using "Current Widget: Battery monitor" tool (http://opencv.org/, April 2014). The time period $(t-1, t]$, $\forall t$, is set to 100 ms. The simulations are averages of 100 independent runs. The proposed scheme is solved using measured real data using the LP in Sect. 5.2. The IBM CPLEX optimizer (http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. Sept 2014) is used to solve the integer linear problem, which is known to be NP hard.

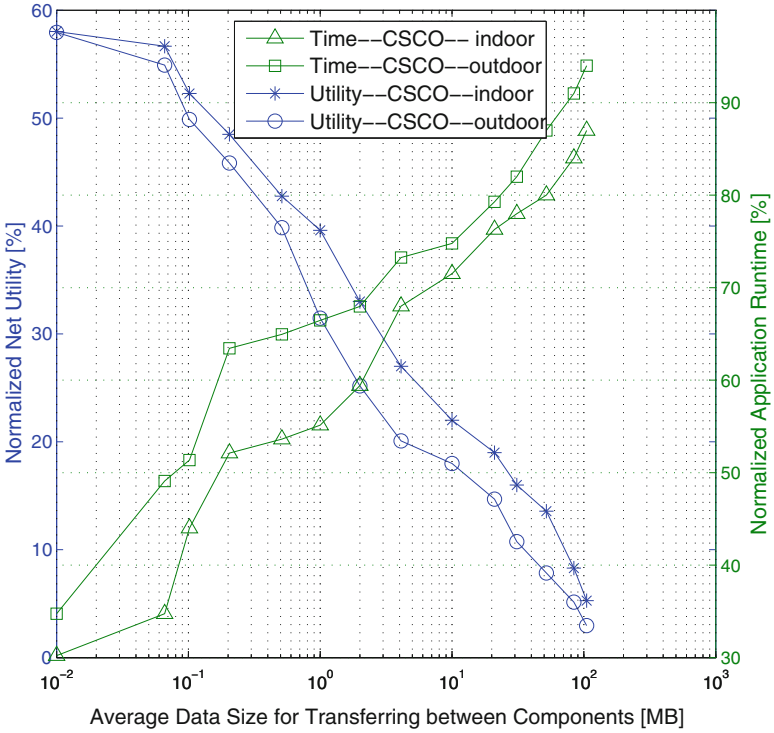


Fig. 7.14 Total net utility percentage (normalized to the ideal net utility) and total application runtime (normalized to application runtime in the mobile) versus average transferred data size between components for $N = 14$, $w_{\text{com}} = 0.5$, and CDG is series-parallel

If monetary constraints are important to the end user, then the weights as control knobs to the individual interfaces can be addressed to reflect the concerns. In some cases the solution can be weighted accordingly. Example of the other types of scenarios includes business clients who normally travel with their full subscription LTE connections. In tactical and emergency response conditions, the most important consideration is to get the job done, rather than minimizing the monetary costs. The proposed solution can work for all these use cases.

7.4.1 Results and Discussion

In this subsection, we address the results of proposed strategies based on latencies for offloading, overall performance (net utility), application runtime, and energy consumption. Also we investigate the behavioral conditions and effects of discussed strategies.

Figure 7.13 shows the time-adaptiveness of CSCO strategy in indoor scenario. Figure 7.13a plots the uplink delay at WiFi radio interface for the 1300 ms during

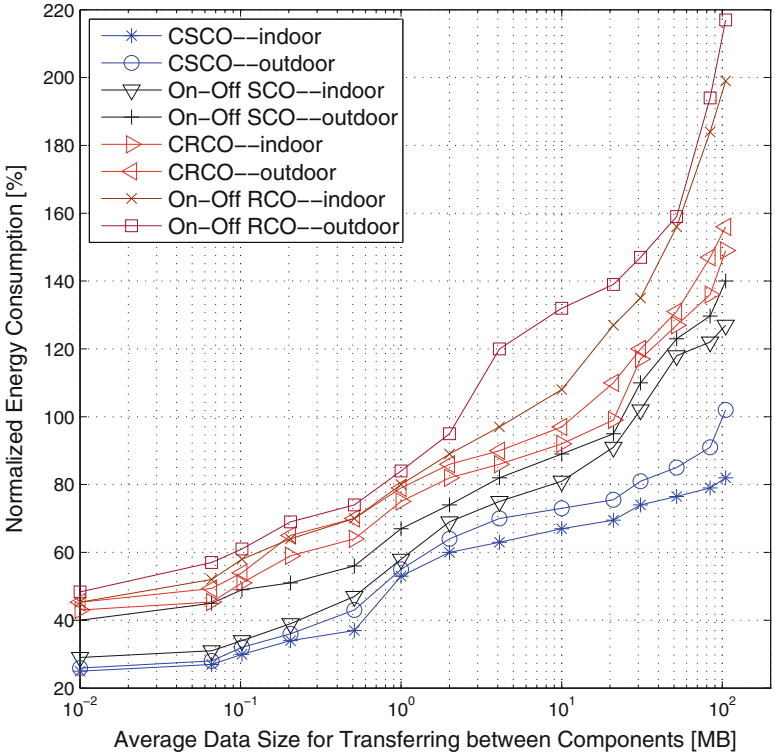


Fig. 7.15 Total energy consumption of the application by the mobile device (normalized to the energy consumed by mobile-only execution) for eight scenarios versus average data size required for transferring between components while $N = 14$, $w_{\text{com}} = 0.5$, and CDG is series-parallel

which the 14-component application is running. We can see that the delay varies widely during this time period. The proposed strategy relies on many wireless parameters in all the radio interfaces for both uplink and downlink scenarios such as communication power, delay, queue backlogs of mobile transmission buffers and cloud transmission buffers. Here an average of 100 simulations is taken using current measurements for delay parameters in Fig. 7.13a while other wireless parameters change at each run, and so we can see the effect of changes of WiFi delay in the final decision.

Figure 7.13b illustrates the percentage of radio allocation using CSCO (cognitive cloud offloader) in uplink scenario ($a_k(t)$, $k = 1$: WiFi, $k = 2$: LTE) for running the 14-component (video navigation) application in 1300 ms. We observe that after 245 ms running the application, offloading is done using 24% of WiFi for communication while 76% is allocated to LTE. Looking at Fig. 7.13a, we observe that at this time, the range of WiFi delay is between 80 and 100 ms. However, the percentage of WiFi allocation at the next step is 91% after 490 ms running of the application. At this time slot we can seen in Fig. 7.13a that the range of WiFi delay

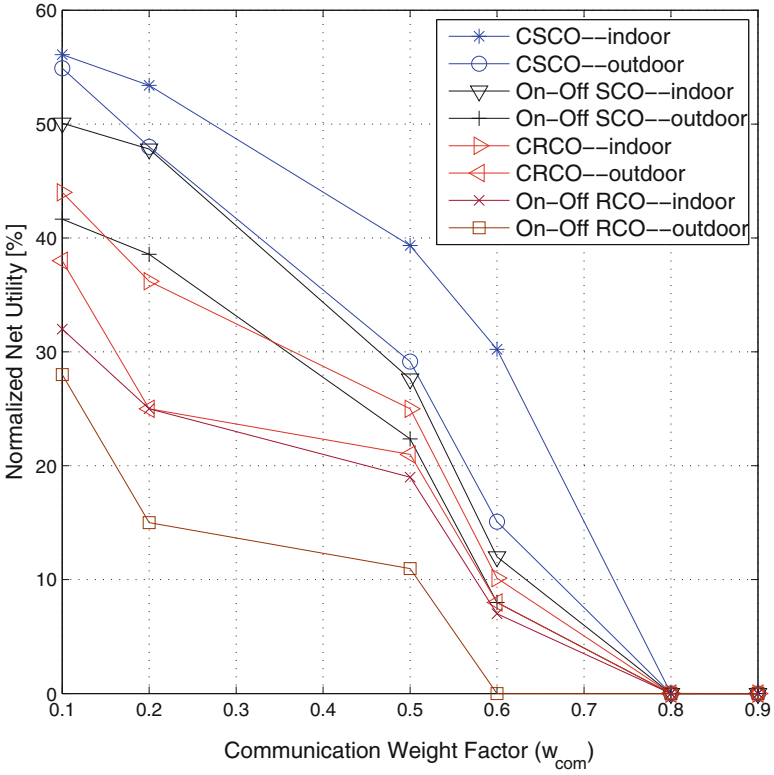


Fig. 7.16 Total net utility (normalized to the ideal net utility) for eight scenarios versus communication weight factor while $N = 14$, average data size for transferring is 11.28 MB, and CDG is series-parallel

is at lowest point (between 30 and 50 ms). Also we observe that the percentage of WiFi allocation decreases to 76% when the WiFi delay has increased (between 50 and 65 ms).

Figure 7.14 illustrates the total net utility as given in Eq. (5.3) (normalized to the ideal net utility) and total application runtime (normalized to the runtime of application using mobile-only execution) versus average transferred data size between components in both indoor and outdoor environments for the CSCO scheme. We define the ideal net utility as the utility obtained in an "ideal world" where there is no cost associated with offloading ($w_{\text{com}} = 0$) and hence all components can be offloaded to the cloud. This would be an upper bound on the utility. From Fig. 7.14, we see that as average data size increases, the normalized net utility decreases (please note that the absolute value for net utility increases) and the normalized application runtime increases. Result shows that, when the average data size for transferring between components increases to 105 MB, the application runtime using CSCO strategy is 48% faster than the application runtime in the local

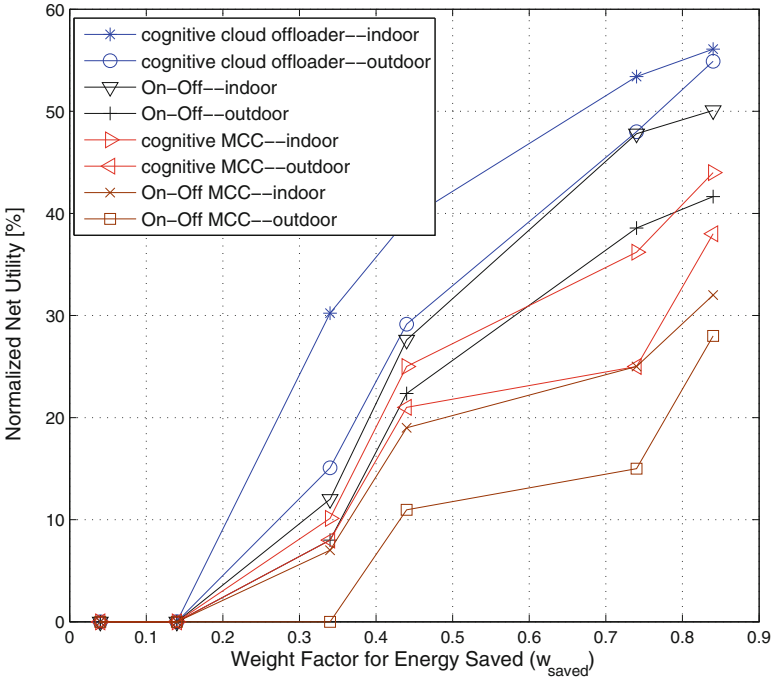


Fig. 7.17 Total net utility (normalized to the ideal net utility) for eight scenarios versus weight factor for energy saved in the mobile device while $N = 14$, average data size for transferring is 11.28 MB, and CDG is series-parallel

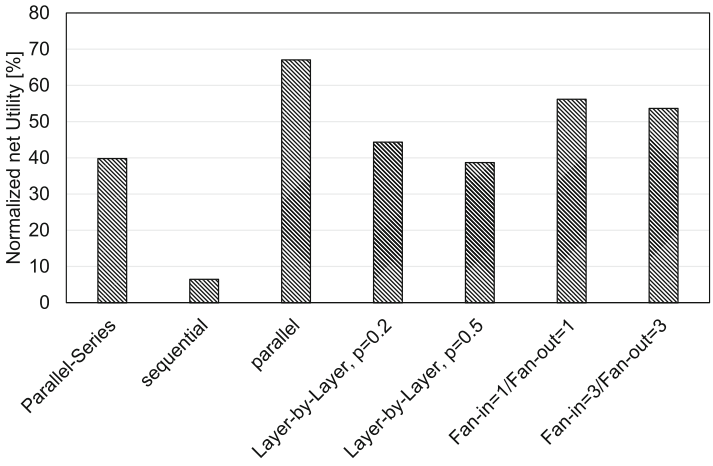


Fig. 7.18 Total net utility (normalized to the ideal net utility) for seven applications with different CDGs while average data size for transferring is 11.28 MB, application runtime is 285 s, $w_{\text{com}} = 0.5$, and $N = 14$

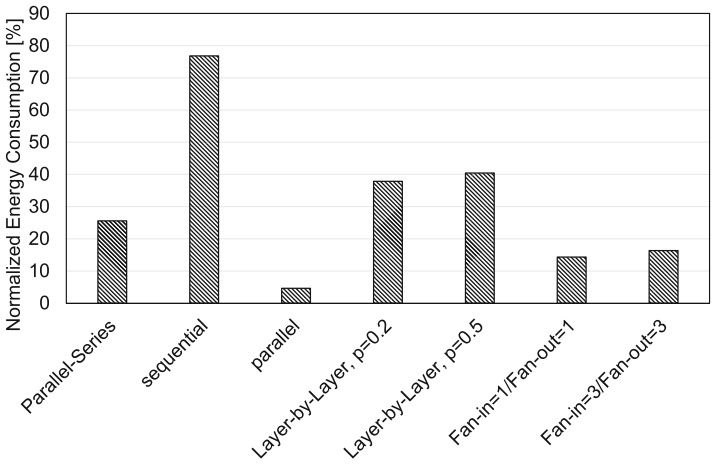


Fig. 7.19 Total energy consumption of the application by the mobile device (normalized to the energy consumed by mobile-only execution) for seven applications with different CDGs while average data size for transferring is 11.28 MB, application runtime is 285 s, $w_{\text{com}} = 0.5$, and $N = 14$

scenario while the cost of offloading using CSCO strategy is also still 6% lower than the net utility saved by remote execution, so it is totally worth it to use the CSCO strategy. We also see that the normalized net utility is higher for indoor environments than outdoor, as expected. This figure shows only two scenarios to more concentrate on the rates of changes in net utility and application runtime at the same time while the average data size for transferring increases. The performance comparison of eight scenarios will be done in the next figures.

Figure 7.15 plots the total normalized (w.r.t energy consumed to run the application fully on the mobile) versus the average offload data size for various schemes: CSCO, On-Off SCO, CRCO, and On-Off RCO in indoor and outdoor environments. Please note that CSCO is the optimal proposed scheme for cognitive joint scheduling and offloading; On-Off scheduling and cloud offloading (On-Off SCO) shows the joint scheduling–offloading (JSCO) strategy, with an additional constraint of using only the best radio interface at every time slot; and cognitive remote cloud offloading (CRCO) is the scheme where all components must be offloaded and at each time epoch, the cognitive offload is scheduled on the multi-RAT networking.

As expected, the energy consumed increases with data size, for all schemes and environments. In all the schemes, mobile device consumes less energy in the indoor environment in comparison to the outdoor. The proposed CSCO outperforms all the other schemes. Note that, the CSCO in the *outdoor* environment performs better than the On-Off SCO in the *indoor* environment. Moreover, we can see that the schemes using cloud-only execution (CRCO and On-Off RCO) consume higher

energy even in comparison to the mobile-only execution in higher ranges of data sizes. This happens for data sizes exceeding 10 MB.

Figure 7.16 shows the total net utility (normalized to the ideal net utility) for 8 scenarios versus weight factor of communication costs. Note that higher w_{com} leads to overall lower net utility, but can be more representative of practical scenarios, communication costs could be more important, than costs to access cloud service. Figure 7.16 also shows that the proposed CSCO performance is better than On-Off SCO.

Figure 7.17 plots the total net utility (normalized to the ideal net utility) for eight scenarios versus weight factor of energy saved in the mobile device. Higher battery power leads to higher overall net utility because of more battery saving.

In Figs. 7.18 and 7.19, normalized net utility and normalized energy consumed by the mobile device are compared for different CDG topologies. Parallel and sequential dependency graphs show the lowest and highest dependencies between components, respectively. Therefore, lower and upper bounds for the cost of offloading could be obtained by using these CDGs. Parallel-series CDG, which is the real CDG applied for the video navigation application given in Fig. 2.2c, gives net utilities and energy values between the two bounds. Also, two randomly generated CDGs of Layer-by-Layer and Fan-in/Fan-out (with parallel-series topologies) methods [9] are compared in these bar graphs. In the Layer-by-Layer method, the number of layers is 5. While p (probability of edge connection) increases from 0.2 to 0.5, more components are dependent on each other, and the density of the CDG increases. Therefore, the total net utility decreases, and more energy is consumed. The results show that when the in+out degree increases from 2 to 6, the dependency and offloading costs increase such that the energy saved through cloud offloading decreases and the energy consumed by the mobile device increases.

7.5 Time-Adaptive Cognitive Offloading and Scheduling Using Multi-Radios

Here in the simulations of Fig. 7.22, synthetic applications with arbitrary CDGs were used in order to test the heuristic scheme for large number of components as well as different types of CDG structures [9]. Also in Fig. 6.3, the face recognition application in http://darnok.org/programming/face-recognition/ was used. The wireless parameters such as uplink and downlink rates in each time slot, packet latencies, and the power consumed for transmission and receiving in each time slot were measured. Also, a Poisson distributed background arrival data in the mobile device is added at each time slot [45] to simulate ambient traffic not related to this particular application.

7.5.1 Versions of the Heuristics

This work is compared to several other scenarios (1) no offload (mobile-only) execution, (2) all offload (cloud-only) execution, (3) the dynamic offloading algorithm (DOA) in [19], and several variants of offload strategies that are shown below:

- **Exhaustive Search:** In this scheme, the 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. Note that this exhaustive search also includes the local execution of components. Also, this method is computationally prohibitive for large and complex applications. This is provided here only to give an idea of the performance of the heuristics vis-a-vis the optimal solution.
- **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. Hence, this strategy constantly checks all the components for offloading in the online stage (except the first and last ones).
- **Two-Stage Heuristic Algorithm (H-2Stage):** This is the algorithm in which the preprocessing stage (described in Sect. 6.2.1) 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 Sect. 6.2. Note that since the offline optimization problem (OP_{off}) is considered with incomplete online information, 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. Also, the joint multi-RAT allocation is considered for offloading strategy in this scheme.
- **Single Stage Heuristic under On-Off model for the Wireless Interfaces (H-1S-OnOff):** In this scheme, the single stage heuristic algorithm is used, where all components are considered for offloading at the online stage, but the components are offloaded using only *one* of the wireless interfaces. The wireless interface with the best characteristics for that time slot is selected. Although [19] and [45] also use On-Off model for the wireless interface, they are different from this heuristic variant. In [45], 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 an optimal/suboptimal scheduling order

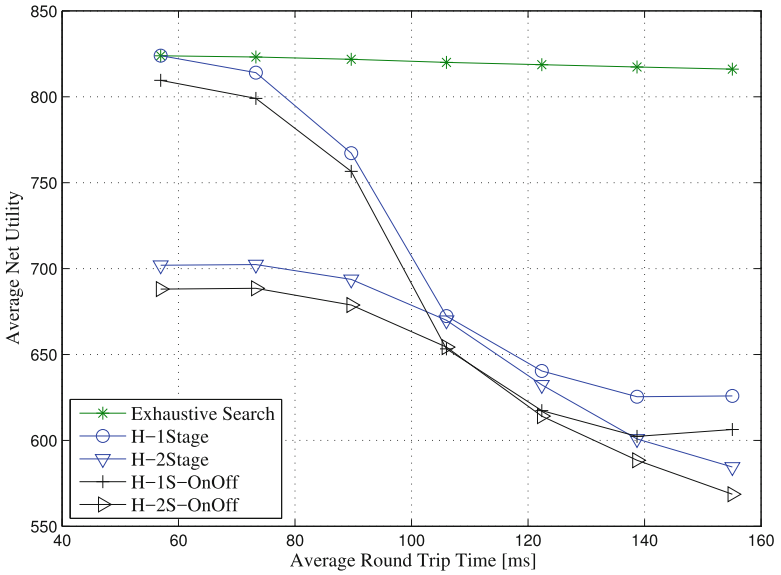


Fig. 7.20 Average net utility versus average round trip time (execution deadline of the application =1330 ms, time threshold for offloading = 550 ms)

for the components but rather uses a pre-determined, serial execution order for the components. Also, [19] is not a time-adaptive strategy.

- **Two-Stage Heuristics with ON/OFF Wireless Interfaces (H-2S-OnOff):** In this scheme, the two-stage algorithm is used for offloading, but the wireless interface with the best characteristics for data transfers of both the mobile to cloud and the cloud to mobile is used for offloading.

7.5.2 Results and Discussion

In Fig. 7.20, the average net utility (Eq. (6.1)) 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) in WiFi ($\tau_{i,1}^{\mathrm{mc}}(t)$, $\tau_{i,1}^{\mathrm{cm}}(t)$, $\forall i, t$) and LTE ($\tau_{i,2}^{\mathrm{mc}}(t)$, $\tau_{i,2}^{\mathrm{cm}}(t)$, $\forall i, t$) radio 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 heuristic schemes with cognitive networking outperform in comparison to 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.

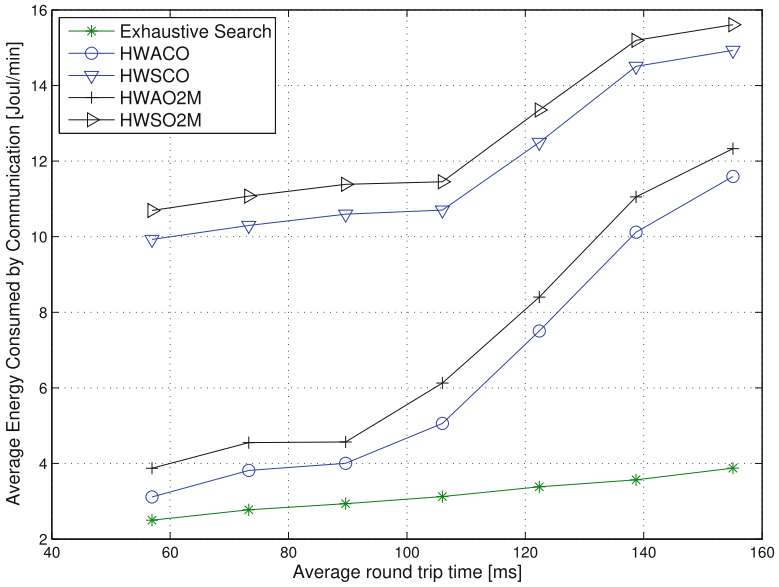


Fig. 7.21 Average energy consumed for communication versus average round trip time (execution deadline of the application = 1330 ms, time threshold for offloading = 550 ms)

This means that cognitive networking strategy covers the low performance caused by the offline stage. Also in Fig. 7.21, the five schemes are plotted for comparison of average energy consumed for offloading versus the average RTT. We can see that while latency increases, more energy is consumed for communication.

So far, we have observed the results for specific mobile applications with 10 and 14 components. We need to investigate the scalability of the heuristic by the larger multi-component applications. Figure 7.22 plots the average net utility as a function of the number of application's components. This experiment is also tested for random applications with [10, 25, 40, 55, 70, 85, 100, 115, 130, 145] components, while the corresponding CDGs were obtained based on a random graph generation of Fan-in/Fan-out with average input degree and output degree of one [9]. The maximum acceptable delay for offloading and RTT is set to 550 ms and 100 ms, respectively. In order to maintain the same probability distribution of the measurements when scaling up the application, the histogram values of the current real data measurements (q_i^{m}, q_i^{c}, P_i^{m}, $\forall i$) are calculated from the 14-component video navigation application. Using the obtained distribution, the new data is generated for applications with a greater number of components. We see that while the number of components increases, the complexity of the application (more execution times and more component dependencies) increases so that higher net utility is saved. Note that as the number of application's components increases, the heuristic is considered for a larger space of components such that the performance

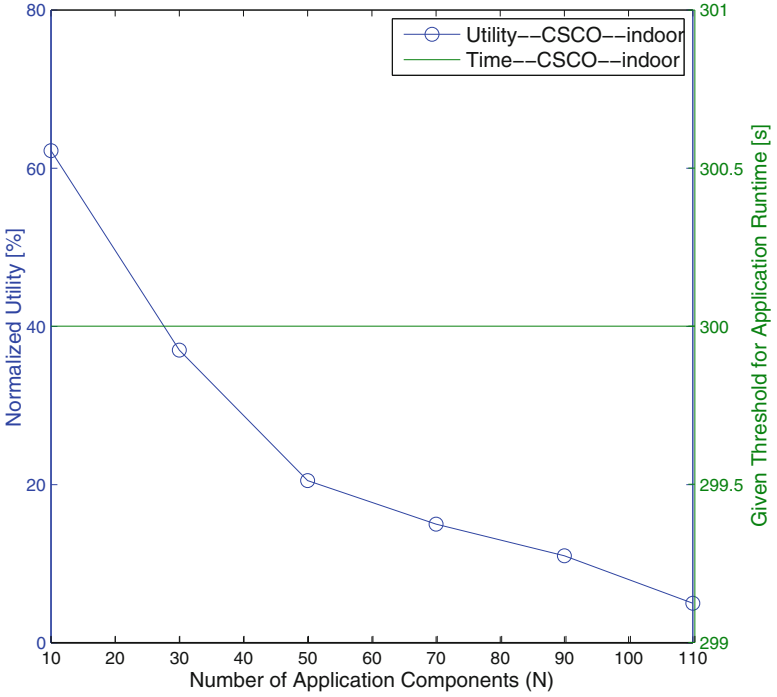


Fig. 7.22 Average net utility versus number of application's components where CDGs are based on a random graph of Fan-in/Fan-out (execution deadline of the application = 1330 ms)

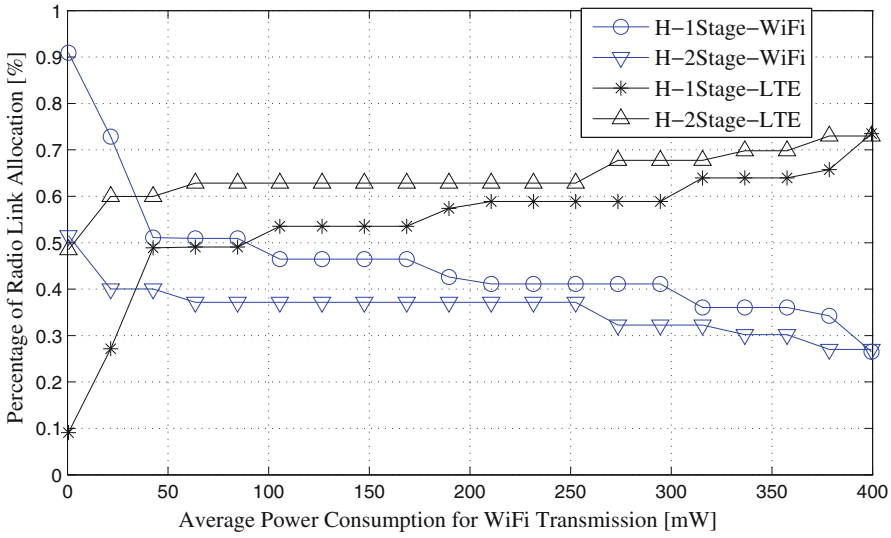


Fig. 7.23 Percentage of radio interface allocation versus time average power consumption for WiFi transmission by the mobile device

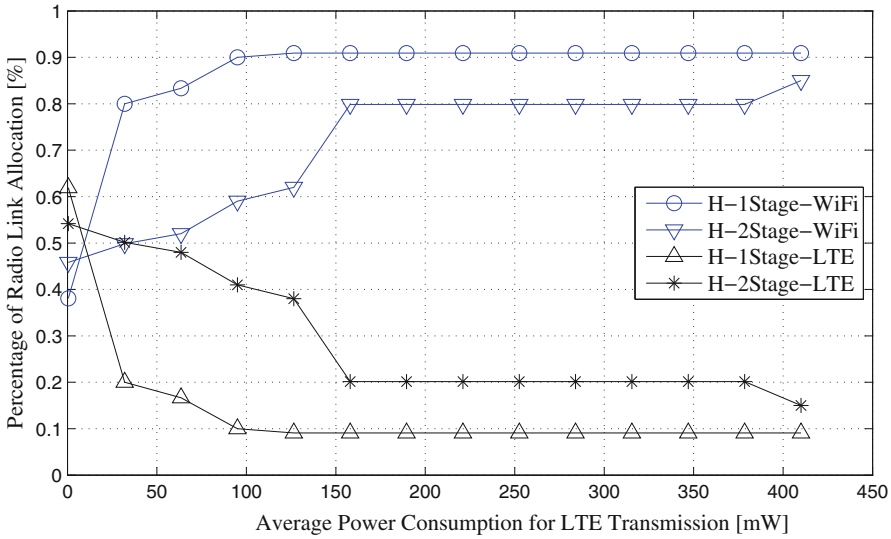


Fig. 7.24 Percentage of radio interface allocation versus time average power consumption for LTE transmission by the mobile device

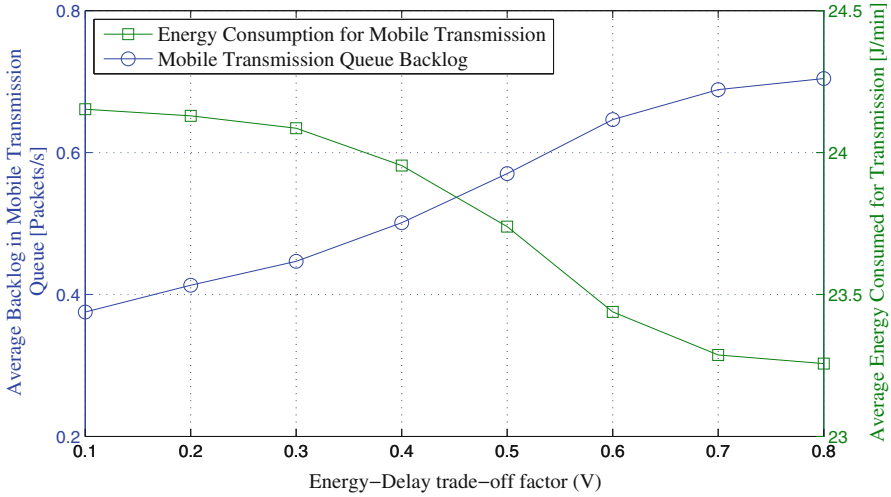


Fig. 7.25 The impact of energy-delay trade-off factor on the average values of transmission queue backlog and energy consumed for transmitting the offloaded data (the maximum acceptable delay for offloading = 550 ms, execution deadline of the application = 1330 ms)

distance of best case scenario (exhaustive search) and the scheme (H-1Stage) increase.

Figures 7.23 and 7.24 show the percentages of radio interface allocation for both WiFi and LTE versus the time-averaged transmission power levels of WiFi and

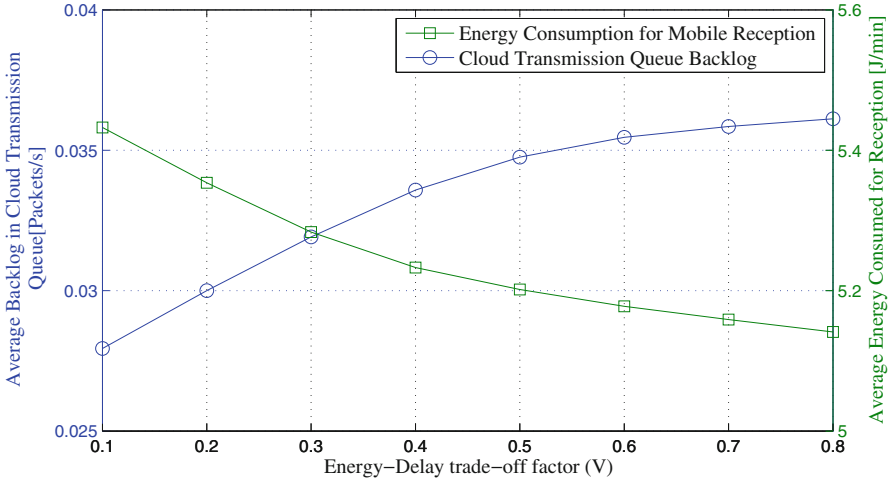


Fig. 7.26 The impact of energy-delay trade-off factor on the average values of cloud transmission queue backlog and energy consumed for receiving the offloaded data (maximum acceptable delay for offloading = 550 ms, execution deadline of the application = 1330 ms)

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. 7.23, results show that when the transmission power level of WiFi increases, the percentage of WiFi allocation decreases and percentage of LTE interface allocation increases. We can see that in the very low ranges of transmission power by WiFi, 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 the transmission power level at WiFi, we observe that the performance of both schemes is close to each other. In Fig. 7.24, we observe that when the rate of transmission power at LTE interface increases, the percentage of interface allocation for LTE decreases and percentage of interface allocation by WiFi increases.

In Figs. 7.25 and 7.26, queue backlogs of mobile transmission buffers and cloud transmission buffers are presented respectively as functions of the trade-off control factors of 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. We observe that when the Lyapunov control parameter of V increases, more time-averaged queue backlog is provided by the scheme and less energy is consumed by the mobile device for offloading. Note that the scales in energy consumption and queue backlog in the mobile transmission part are both higher than the cloud transmission part to the mobile. Thus, the transmission strategy affects more in the resource management of the whole system.

Finally, Fig. 7.27 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

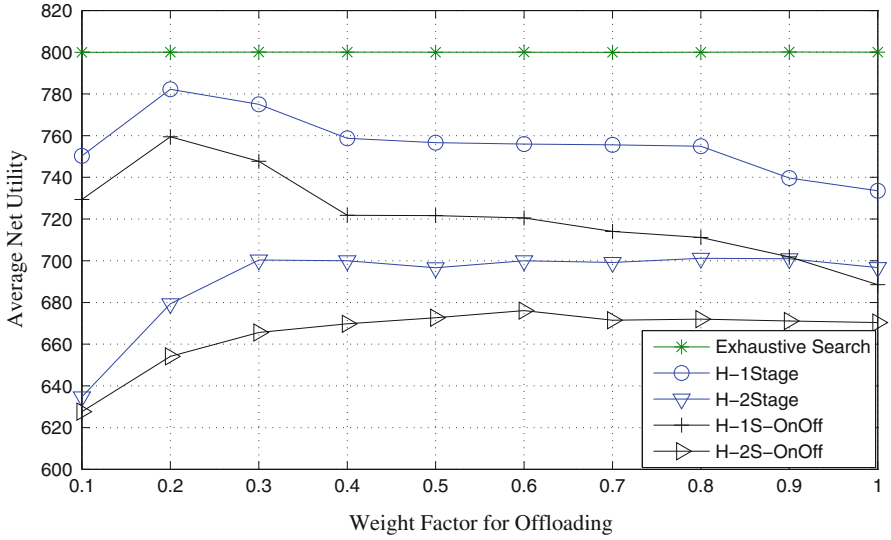


Fig. 7.27 Average net utility versus weight factor for offloading, γ (the maximum acceptable delay for offloading = 550 ms, execution deadline of the application = 1330 ms)

time to wait for offloading the components, rather than local execution, increases. In H-1Stage and H-1S-OnOff, where all the components are checked for offloading, there is a maximum point that gives the highest net utility (at $\gamma = 0.2$). These maximum points for H-2Stage and H-2S-OnOff are obtained at $\gamma = 0.3$ and $\gamma = 0.6$. Thus, each of these schemes has a γ that gives the highest net utility for the application. However, we observe that the net utility changes by less than 1.25% in the exhaustive search scheme, while γ changes.

7.6 Summary of Performance Evaluation

This chapter included a case study for performance evaluation of scheduling and computation offloading schemes in the following scenarios: (1) joint scheduling and cloud offloading for single-radio enabled mobile devices; (2) cognitive offloading using multiple radios; (3) time-adaptive cognitive offloading and scheduling using multiple radios; and (4) optimal cognitive offloading and scheduling using multiple radios. In the first scenario, the results illustrated that the energy saved increases with longer application runtime deadline, higher rates of wireless radio interface, and smaller offload data size. In the second scenario, the algorithm for one-shot offloading solution consumes within 4% of the optimal solution (obtained via brute force search) and also offers 31% less energy consumption in comparison to offloading the entire application to the cloud. Also, the performance of the heuristic strategy was analyzed in the third scenario. The computations were offloaded

via two wireless interfaces of WiFi and LTE to the Amazon EC2 cloud. Several versions of the heuristic strategy were compared with the optimal scheme (which has been provided by exhaustive search method), a recent strategy in dynamic computation offloading [19], and the classical strategies where all the components are executed locally and remotely. Finally, the results for a holistic approach for cognitive offloading and scheduling using multi-radios were analyzed. The results from collected real data measurements illustrated that the energy consumed by the mobile device using CSCO is respectively 51%, 23%, 68%, and 42% lower in comparison to the best-interface protocol, non-time adaptive schemes, mobile-only and cloud-only executions. The experiments were run in both outdoor and indoor wireless environments.

Chapter 8
The Future: Spectrum Aware Cloud Offloading

As articulated in the previous chapters, the future of mobile computing will become application and RAT aware. Looking further into the future, it is possible to imagine that the spectrum aware cloud offloader will become spectrum-opportunistic, in the sense that the cognitive cloud offload manager will not only respond to known available wireless networks, but will also actively seek spectrum opportunities at runtime to distribute the load more effectively. This will be enabled by the immense amount of work done in the cognitive radio networking and dynamic spectrum access networking.

Furthermore, it will be possible not only to manage computation offloading, but also to integrate it with spectrum and traffic management. We envisage a future as depicted in Fig. 8.1. This figure shows multiple mobile users running various sophisticated apps (augmented reality (AR), computer vision, face recognition, social media, etc.) on devices that can simultaneously take advantage of multi-RAT technology to not only access multiple wireless networks but also dynamically manage spectrum access. There are multiple wireless networks in the architecture such as WiFi, LTE, 4.9 GHz, and TV white space (TVWS) channels. TVWS channels refer to the unused TV channels among the active broadcasters while unused spectrum bands can provide reliable spectrum access. The dynamic variations in spectrum use are observed for the wireless networks (TVWS, WiFi, and LTE) over a period of time. Using the right dynamic spectrum management strategies will allow an overall best decision for spectrum access in conjunction with computation to be made.

This architecture will be an integrated dynamic cloud offloader–spectrum manager (DCSM) comprising of two parts: (a) the dynamic computation offloader (DCO) that determines optimal/near optimal energy and radio spectrum aware schedulers for components of mobile applications and (b) the dynamic spectrum manager (DSM) that will provide related spectrum management decisions taking into account the end-to-end network parameters. This entity will provide end-to-

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6_8

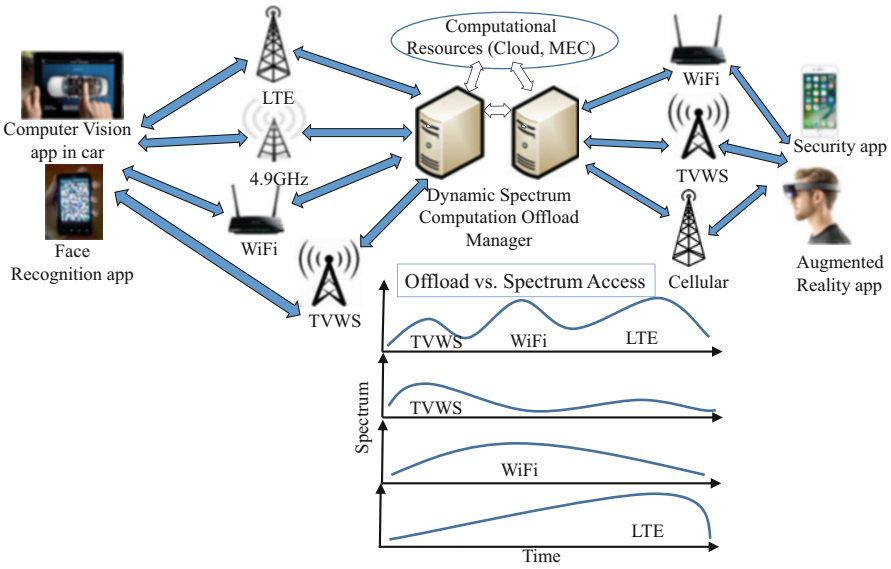


Fig. 8.1 The future: dynamic spectrum aware cloud offloading. The future of mobile computing will be able to use cognitive radio technologies to sense for spectrum opportunities when offloading components to the cloud while also being able to aggregate bandwidths from appropriate wireless backhaul networks such as Verizon and AT&T

end dynamic spectrum management architecture and offloading solutions that take into account real-time measurement of multiple spectrum bands as well as the parameters of the associated core networks (e.g., Comcast, Verizon LTE core).

The first forays into this area [33] has shown significant promise while working with components and CDGs as noted in the earlier chapters of this book. Further research is needed to integrate these approaches to more fine-grained partitioning of the code. This should provide a finer control of the solutions space and give the practitioner a broader set of choices for operating regions. As measurements are being made and models constructed for the "newer" network types including millimeter wave technologies, it would be interesting to expand the modeling and optimization frameworks for these other networks as well. Significant measurement and simulation work will be needed to understand the trade-offs that will be brought about by the widely varying parameters of these networks. Mobility models can be incorporated to further enhance the understanding of these trade-offs in a mobile multi-network environment. In short this concept offers a plethora of new research topics that will result in more resource aware, faster applications that can satisfy the promise of 5G technologies.

Bibliography

1. P. Balakrishnan, C.K. Tham, Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing, in *IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, December 2013, pp. 34–41
2. S. Barbarossa, S. Sardellitti, P. Di 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
3. S. Boyd, A. Mutapcic, Subgradient methods. Lecture Notes of EE364b, Stanford University, Stanford, CA, Spring Quarter (2008)
4. C. Buschmann, D. Pfisterer, S. Fischer, S.P. Fekete, A. Kröller, Spyglass: a wireless sensor network visualizer. ACM Sigbed Rev. **2**(1), 1–6 (2005)
5. W. Cai, V.C. Leung, M. Chen, Next generation mobile cloud gaming, in *IEEE International Symposium on Service Oriented System Engineering (SOSE)* (2013), pp. 551–560
6. X. Chen, J. Wu, Y. Cai, H. Zhang, T. Chen, Energy-efficiency oriented traffic offloading in wireless networks: a brief survey and a learning approach for heterogeneous cellular networks. IEEE J. Sel. Areas Commun. **33**(4), 627–640 (2015)
7. B.-G. Chun, P. Maniatis, Augmented smartphone applications through clone cloud execution, in *Proceedings of the Conference on Hot Topics in Operating Systems, HotOS'09* (2009), pp. 8–8
8. B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud, in *Proceedings of the Sixth Conference on Computer Systems* (2011), pp. 301–314
9. D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J.-M. Vincent, F. Wagner, Random graph generation for scheduling simulations, in *Proceedings of the International ICST Conference on Simulation Tools and Techniques*, vol. 10(60) (2010), pp. 60:1–60:10
10. E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: making smartphones last longer with code offload, in *Proceedings of the International Conference on Mobile Systems, Applications, and Services, MobiSys* (ACM, New York, 2010), pp. 49–62
11. T. Dao, I. Singh, H.V. Madhyastha, S.V. Krishnamurthy, G. Cao, P. Mohapatra, TIDE: a user-centric tool for identifying energy hungry applications on smartphones, in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2015, pp. 123–132
12. S. Deng, L. Huang, J. Taheri, A. Zomaya, Computation offloading for service workflow in mobile cloud computing. IEEE Trans. Parallel Distrib. Syst. PP(99), 1–1 (2014)

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6

13. W. Dong, S. Rallapalli, R. Jana, L. Qiu, K. K. Ramakrishnan, L. Razoumov, Y. Zhang, T.W. Cho, iDEAL: incentivized dynamic cellular offloading via auctions. IEEE/ACM Trans. Netw. **22**(4), 1271–1284 (2014)
14. T.A. ElBatt, S.V. Krishnamurthy, D. Connors, S. Dao, Power management for throughput enhancement in wireless ad-hoc networks, in *IEEE International Conference on Communications (ICC)*, vol. 3 (2000), pp. 1506–1513
15. L. Georgiadis, M.J. Neely, L. Tassiulas, Resource allocation and cross-layer control in wireless networks. Found. Trends Netw. **1**(1), 1–144 (2006)
16. X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, D. Milojicic, Adaptive offloading for pervasive computing. IEEE Pervasive Comput. **3**(3), 66–73 (2004)
17. C. Gui, P. Mohapatra, Power conservation and quality of surveillance in target tracking sensor networks, in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, MobiCom'04* (2004), pp. 129–143
18. K. Hong, S. Sengupta, R. Chandramouli, Spiderradio: a cognitive radio implementation using IEEE 802.11 components. IEEE Trans. Mob. Comput. **12**(11), 2105–2118 (2013)
19. D. Huang, P. Wang, D. Niyato, A dynamic offloading algorithm for mobile computing. IEEE Trans. Wirel. Commun. **11**(6), 1991–1995 (2012)
20. D. Kaspar, Multipath aggregation of heterogeneous access networks. SIGMultimedia Rec. **4**(1), 27–28 (2012)
21. S. Kosta, A. Aucinas, P. Hui, R. Mortier, 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
22. D. Kovachev, T. Yu, 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
23. U. Kremer, J. Hicks, J.M. Rehg, Compiler-directed remote task execution for power management, in *Workshop on Compilers and Operating Systems for Low Power* (2000)
24. K. Kumar, Y.H. Lu, Cloud computing for mobile users: can offloading computation save energy? Computer **43**, 51–56 (2010)
25. Y. Li, M. Qian, D. Jin, P. Hui, Z. Wang, S. Chen, Multiple mobile data offloading through disruption tolerant networks. IEEE Trans. Mob. Comput. **13**(7), 1579–1596 (2014)
26. S. Li, E. Ekici, N. Shroff, Throughput-optimal queue length based CSMA/CA algorithm for cognitive radio networks. IEEE Trans. Mob. Comput. **14**(5), 1098–1108 (2015)
27. Y.-S. Lim, Y.-C. Chen, E.M. Nahum, D. Towsley, R.J. Gibbens, Improving energy efficiency of MPTCP for mobile devices (2014). arxiv preprint arXiv:1406.4463
28. X. Lin, Y. Wang, Q. Xie, M. Pedram, Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. IEEE Trans. Serv. Comput. **8**(2), 175–186 (2015)
29. Y. Liu, M. Liu, To stay or to switch: multiuser multi-channel dynamic access. IEEE Trans. Mob. Comput. **14**(4), 858–871 (2015)
30. J. Liu, B. Priyantha, T. Hart, H.S. Ramos, A.A.F. Loureiro, Q. Wang, Energy efficient GPS sensing with cloud offloading, in *Proceedings of the ACM Conference on Embedded Network Sensor Systems, SenSys '12* (2012)
31. S. Liu, L. Lazos, M. Krunz, Cluster-based control channel allocation in opportunistic cognitive radio networks. IEEE Trans. Mob. Comput. **11**(10), 1436–1449 (2012)
32. X. Ma, Y. Zhao, L. Zhang, H. Wang, L. Peng, When mobile terminals meet the cloud: computation offloading as the bridge. IEEE Mag. Netw. **27**(5), 28–33 (2013)
33. S.E. Mahmoodi, Spectrum aware cognitive mobile cloud computing. PhD Dissertation, Stevens Institute of Technology (2017)
34. S.E. Mahmoodi, K.P.S. Subbalakshmi, A time-adaptive heuristic for cognitive cloud offloading in multi-rat enabled wireless devices. IEEE Trans. Cogn. Commun. Netw. **2**(2), 194–207 (2016)

35. S.E. Mahmoodi, K.P. Subbalakshmi, V. Sagar, Cloud offloading for multi-radio enabled mobile devices, in *IEEE International Communication Conference (ICC)*, June 2015, pp. 1–6
36. S.E. Mahmoodi, R.N. Uma, K.P. Subbalakshmi, Optimal joint scheduling and cloud offloading for mobile applications. IEEE Trans. Cloud Comput. **PP**(99), 1–1, Early Access (2016)
37. S.E. Mahmoodi, K.P. Subbalakshmi, R.N. Uma, Optimal cognitive scheduling and cloud offloading for mobile applications in multi-Radio enabled devices. IEEE Trans. Cloud Comput. **PP**(99), 1–1 (2017, submitted)
38. M.J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems* (Morgan and Claypool Publishers, San Rafael, 2010)
39. M. Nir, A. Matrawy, M. St-Hilaire, An energy optimizing scheduler for mobile cloud computing environments, in *IEEE Conference on Computer Communications Workshops (INFOCOM Workshops)*, April 2014, pp. 404–409
40. S. Ou, K. Yang, J. Zhang, An effective offloading middleware for pervasive services on mobile devices. Pervasive Mob. Comput. **3**(4), 362–385 (2007)
41. J.K. Ousterhout, Scripting: higher level programming for the 21st century. IEEE Comput. Mag. **31**, 23–30 (1998)
42. J. Part, H.C. Yu, E.Y. Lee, Fault tolerance technique based on monitoring and pattern for reliable resource management in mobile cloud computing. J. Internet Technol. **14**(6), 997–1005 (2013)
43. M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al., Mobile-edge computing introductory technical white paper, in *White Paper, Mobile-Edge Computing (MEC) Industry Initiative* (2014)
44. C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, 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, MobiHoc '14* (ACM, New York, 2014), pp. 287–296
45. P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, eTime: energy-efficient transmission between cloud and mobile devices, in *IEEE Conference on Computer Communications (INFOCOM)*, April 2013, pp. 195–199
46. T. Shuminoski, T. Janevski, Lyapunov optimization framework for 5G mobile nodes with multi-homing. IEEE Commun. Lett. **20**(5), 1026–1029 (2016)
47. A.S.M. Toma, J.-J. Chen, Computation offloading for frame-based real-time tasks under given server response time guarantees. Leibniz Trans. Embed. Syst. **1**(2), 1–21 (2014)
48. H. Topcuoglu, S. Hariri, M.-Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
49. W. Zhang, Y. Wen, D. Wu, Energy-efficient scheduling policy for collaborative execution in mobile cloud computing, in *IEEE Proceedings on INFOCOM*, April 2013, pp. 190–194
50. W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, D. Wu, Energy-optimal mobile cloud computing under stochastic wireless channel. IEEE Trans. Wirel. Commun. **12**(9), 4569–4581 (2013)
51. Y. Zhang, D. Niyato, P. Wang, Offloading in mobile cloudlet systems with intermittent connectivity. IEEE Trans. Mob. Comput. **14**(12), 2516–2529 (2015)
52. W. Zhang, Y. Wen, D. Wu, Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans. Wirel. Commun. **14**(1), 81–93 (2015)
53. K. Zhu, E. Hossain, D. Niyato, Pricing, spectrum sharing, and service selection in two-tier small cell networks: a hierarchical dynamic game approach. IEEE Trans. Mob. Comput. **13**(8), 1843–1856 (2014)
54. M. Zorzi, A. Zanella, A. Testolin, M. De Filipo De Grazia, M. Zorzi, Cognition-based networks: a new perspective on network optimization using learning and distributed intelligence. IEEE Access **3**, 1512–1530 (2015)

Index

S. E. Mahmoodi et al., *Spectrum-Aware Mobile Computing*,
Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-02411-6