Energy-Efficient Synchronization and Resource Allocation Strategies for Device-to-Device Communications

Onur Karatalay

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

April 2022

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Device-to-Device (D2D) communication technology offers energy-efficient, high throughput, and ultra-low latency data transmissions while significantly reducing the total overhead and data traffic at the core network. Nevertheless, there are several practical aspects to be considered in the realization of this type of peer-to-peer, close-proximity communication framework. Among these, in this thesis, we focus on two fundamental challenges, namely, synchronization and resource allocation, towards the implementation of reliable, robust and efficient D2D communications.

Since synchronization is the foundation of reliable data communications, we begin with addressing the clock synchronization problem in distributed D2D networks. In this regard, we first develop a scalable, pulse-based half-duplex synchronization algorithm that operates in a timing-advance fashion by taking the signal propagation time into account. Then, in light of this algorithm, we propose a fully distributed synchronization protocol that allows synchronized devices to proceed into data communication after acquiring the global synchronization status of the network. The proposed protocol also allows synchronized devices to become idle and save energy while maintaining synchronization. Compared to a benchmark from the literature, the proposed protocol not only achieves faster synchronization but also attains a lower synchronization error under various simulation conditions such as multi-path channels, clock skew, dynamic number of devices, and different network topology.

After resolving the synchronization problem, we consider D2D communications as an enabling technology for a task offloading framework. In this regard, we address the energy-efficient resource allocation problem in a D2D-aided task offloading scenario, in which computation intensive tasks can be offloaded to nearby available wireless devices via D2D links. However, high operating temperatures during task processing result in CPU throttling, which randomly alters the task processing time. We formulate the resource allocation problem to minimize the expected total energy consumption subject to probabilistic constraints on the task processing time. Since the formulated problem is non-convex, we develop two sub-optimal methods to solve it. The first method relies on Difference of Convex (DC) programming combined with chance-constraint optimization to handle the probabilistic constraints. However, the performance of DC programming depends on a good initial point, hence, we propose a second method, which only uses convex programming. As a benchmark, we adopt the total energy consumption when the task is computed only locally. The simulation results show that both methods significantly reduce the energy consumption in comparison to the benchmark while the latter method outperforms the former in terms of energy efficiency and run time.

Finally, we investigate a multi-device D2D-aided task offloading scenario by including a more powerful, centralized computation server to be utilized simultaneously. Due to the non-convex nature of the formulated problem, we propose two methods to sub-optimally solve it. First, by investigating the relationship between the task processing time and the total energy consumption, the original problem is relaxed into a sequence of convex sub-problems whose solutions can be efficiently obtained by using the convex optimization techniques. Second, to further reduce computational complexity, we develop a low-complexity heuristic task offloading strategy, which does not require computing gradients and Hessian matrices. For performance comparison, we calculate a lower bound on the total energy consumption for an ideal scenario to be used as a benchmark. The computer simulations show that both methods significantly reduce the total energy consumption compared to processing tasks only locally while attaining near-optimal solutions with respect to the lower bound.

Sommaire

La technologie de communication d'appareil à appareil (D2D) offre des transmissions de données écoénergétiques, à haut débit et à latence ultra-faible, tout en réduisant la surcharge totale et le trafic de données au niveau du réseau central. Néanmoins, plusieurs aspects pratiques sont à considérer dans la réalisation de ce type de cadre de communication pair-à-pair et de proximité. Dans cette thèse, nous nous concentrons sur deux défis fondamentaux, à savoir, la synchronisation et l'allocation des ressources, dans le but d'assurer des communications D2D fiables, robustes et efficaces.

Puisque la synchronisation est le fondement de communications de données fiables, nous commençons par aborder le problème de synchronisation d'horloge dans les réseaux D2D distribués. À cet égard, nous développons tout d'abord un algorithme de synchronisation semi-duplex évolutif et basé sur des impulsions; l'algorithme exploite l'avance temporelle prenant en compte le temps de propagation du signal. Nous proposons ensuite un protocole de synchronisation entièrement distribué qui permet aux appareils synchronisés de passer à la communication de données après avoir acquis l'état de synchronisation globale du réseau. Le protocole proposé permet également aux appareils synchronisés de devenir inactifs et d'économiser de l'énergie tout en maintenant la synchronisation. En comparaison aux méthodes existantes, le protocole proposé permet non seulement une synchronisation plus rapide, mais atteint également une erreur de synchronisation plus faible dans diverses conditions de simulation telles que les canaux multi-chemins, le décalage d'horloge, le nombre dynamique d'appareils et la topologie de réseau.

Après avoir résolu le problème de synchronisation, nous considérons les communications D2D comme une technologie habilitante pour un cadre de délestage de tâches. À cet égard, nous abordons le problème d'allocation de ressources écoénergétique dans un scénario de déchargement de tâches assisté par D2D, dans lequel les tâches gourmandes en calculs peuvent être délestées vers des appareils sans fil disponibles à proximité via des liaisons D2D. Cependant, des températures de fonctionnement élevées pendant le traitement des tâches entraînent une limitation du processeur, ce qui modifie aléatoirement le temps de traitement. Nous formulons le problème d'allocation des ressources pour minimiser la consommation totale d'énergie attendue sous réserve de contraintes probabilistes sur le temps de traitement de la tâche. Puisque le problème formulé est non convexe, nous développons deux méthodes sous-optimales pour le résoudre. Les résultats de la simulation montrent que les deux méthodes réduisent considérablement la consommation d'énergie par rapport à une approche de référence, tandis que la deuxième méthode surpasse la première en termes d'efficacité énergétique et d'autonomie.

Enfin, nous étudions un scénario de délestage de tâches assisté par D2D multi-appareils en incluant un serveur de calcul centralisé plus puissant à utiliser simultanément. En raison de la nature non convexe du problème formulé, nous proposons deux méthodes pour le résoudre de manière sous-optimale. Premièrement, le problème d'origine est décomposé en une séquence de sous-problèmes convexes dont les solutions peuvent être efficacement obtenues en utilisant les techniques d'optimisation convexe. Deuxièmement, pour réduire la complexité de calcul, nous développons une stratégie de délestage des tâches heuristique de faible complexité. Pour la comparaison des performances, nous calculons une limite inférieure sur la consommation totale d'énergie pour un scénario idéal, à utiliser comme référence. Les simulations informatiques montrent que les deux méthodes réduisent considérablement la consommation totale d'énergie par rapport aux tâches de traitement effectuées uniquement sur l'unité locale, tout en atteignant des solutions presque optimales par rapport à la limite inférieure.

Acknowledgement

First and foremost, I would like to express my deepest gratitude towards my supervisor Prof. Ioannis Psaromiligkos and my co-supervisor Prof. Benoit Champagne for giving me the privilege to work under their guidance and learn from their wisdom. Without their expertise and knowledge as well as their continuous support and patience throughout my studies, this thesis would have never been possible. I would also like to thank my Ph.D. supervisory committee members Prof. Xiao-Wen Chang, Prof. Aditya Mahajan and Prof. Michael Rabbat for their constructive criticisms and valuable feedbacks to constantly improve my work.

I am also deeply grateful for the funding from McGill University given under the McGill Engineering Doctoral Awards (MEDA). Also, I profoundly thank my supervisor Prof. Ioannis Psaromiligkos for the financial support for my studies through the research grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and InterDigital Canada Ltée under the Mitacs Accelerate Internship program. Hereby, I would also like to thank Dr. Benoit Pelletier for his technical guidance and expertise as well as his welcoming attitude towards me during my internship at InterDigital Canada Ltée.

Finally, I would like to thank my friends who are always by my side. And most importantly, I would like to express how much I am grateful for my family to have their continuous support and love during my studies. Without their encouragement and guidance, I would not have enough strength to finish this journey. Hereby, I dedicate this thesis to my lovely family.

Preface and Contributions of the Author

The research presented in this dissertation was carried out in the Department of Electrical and Computer Engineering (ECE) of McGill University from September 2016 to August 2021. This dissertation is the result of my original work that is created under the supervisions of Prof. Ioannis Psaromiligkos and Prof. Benoit Champagne from the same department. I, Onur Karatalay, conducted the literature survey and accordingly formulated the problems with the guidance of Prof. Ioannis Psaromiligkos and Prof. Benoit Champagne. Then, I proposed the methods, developed the algorithms, and implemented the computer simulations. Finally, I wrote the first draft of each paper, which are reviewed and edited by the co-authors Prof. Ioannis Psaromiligkos, Prof. Benoit Champagne and Dr. Benoit Pelletier.

- **Journal Papers**

  (J-1) O. Karatalay, I. Psaromiligkos, B. Champagne and B. Pelletier, "A distributed pulse-based synchronization protocol for half-duplex D2D communications," *IEEE Open Jour. of the Commun. Soc.*, vol. 2, pp. 245–261, 2021.

  (J-2) O. Karatalay, I. Psaromiligkos, and B. Champagne, "Energy-efficient resource allocation for D2D-assisted fog computing," (submitted).

- **Conference Papers**

  (C-1) O. Karatalay, I. Psaromiligkos, B. Champagne and B. Pelletier, "Fast converging Distributed pulse-coupled clock synchronization for half-duplex D2D communications over multipath channels," in *Proc. IEEE Int. Symp. on Signal Process. and Info. Tech.*, pp. 123–128, Dec. 2018.

  (C-2) O. Karatalay, I. Psaromiligkos, B. Champagne and B. Pelletier, "Fully distributed energy-efficient synchronization for half-duplex D2D communications," in *Proc. IEEE Pers. Indoor and Mobile Commun.*, pp. 1–7, Sept. 2019.

  (C-3) O. Karatalay, I. Psaromiligkos, and B. Champagne, "Energy-efficient D2D-aided fog computing under probabilistic time constraints," in *Proc. IEEE Glob. Commun. Conf.*, pp. 01–07, Dec. 2021.

Contents

List of Acronyms

3GPP	Third Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
AD	Active Device
AP	Access Point
B5G	Beyond 5G
BS	Base Station
CCP	Convex-Concave Procedure
CDF	Cumulative Distribution Function
CFO	Carrier Frequency Offset
CPU	Central Processing Unit
CSI	Channel State Information
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CUs	Cellular Users
D2D	Device-to-Device
DC	Difference of Convex
DCF	Difference of Convex Functions
DPLLs	Distributed Phase-Locked Loops
DTM	Dynamic Thermal Management
DUs	D2D Users

eNB	evolved Node B
ES	Edge Server
FD	Fog Device
ID	Idle Device
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
LTE	Long Term Evolution
LTE-A	Long Term Evolution-Advanced
MAC	Medium Access Layer
MEC	Mobile Edge Computing
P2P	Peer-to-Peer
PHY	Physical Layer
ppm	parts per million
ProSe	Proximity Services
PUs	Primary Users
QoS	Quality of Service
SINR	Signal-to-Interference-plus-Noise Ratio
SNR	Signal-to-Noise Ratio
SUs	Secondary Users
TA	Timing-Advance
TO	Time-Offset
UE	User Equipment

V2X	Vehicle-to-Everything
WSNs	Wireless Sensor Networks
ZC	Zadoff-Chu

Chapter 1

Introduction

In this chapter, we first introduce the concept of Device-to-Device (D2D) communications and present its role and possible use cases in the Fifth Generation (5G) and Beyond 5G (B5G) networks. We then give the main objectives of our research and outline the contributions of the thesis. Finally, we give an overview of the thesis organization and the mathematical notations used in the thesis.

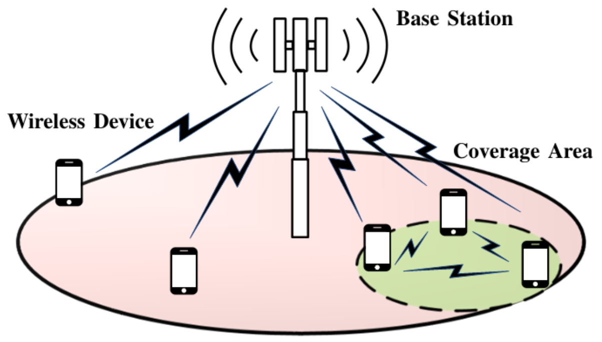
1.1 D2D Networks

By the end of 2023, there will be 13.1 billion mobile devices worldwide, which represents a nearly 50% increase since 2018 [1, 2]. Along with the proliferation of connected devices, the tremendous growth of data traffic has resulted in larger bandwidth demands in the Fourth Generation (4G) of wireless networks, which have initiated the standardization of new technologies by the Third Generation Partnership Project (3GPP) [3–5]. In 4G Long Term Evolution (LTE) and Long Term Evolution-Advanced (LTE-A) networks, low-power consuming fixed Base Stations (BS)s called evolved Node B (eNB), managed efficient spectrum usage and data traffic until recent years; however, they will not be able to support the envisioned data traffic in the next decade [5]. As opposed to 4G networks, where devices connect to the network only via a BS, one of the enabling technologies envisaged for the 5G and B5G networks[1] to satisfy future demands is direct connections between devices in close proximity, also known as D2D communications.

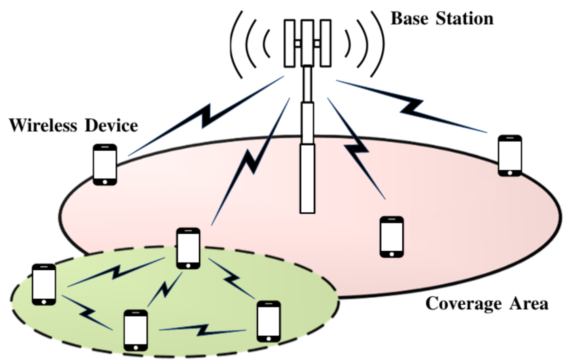
[1]The worldwide deployment of 5G technologies is now progressing rapidly, but initially, only a limited subset of the envisaged functionalities will be available.

D2D communications has been proposed as a way to improve the performance of cellular networks in numerous aspects [6, 7]. First, D2D functionality enables Peer-to-Peer (P2P) communication via *side links* between proximate devices, called D2D Users (DUs). Compared to conventional cellular networks, where BSs must initiate, maintain and relay the connections between Cellular Users (CUs), D2D communications not only reduces the overhead by bypassing the BSs but also mitigates end-to-end transmission delays, which is beneficial for applications that require ultra-low latency [8]. Moreover, by allowing the DUs to operate in dedicated frequency bands, the network capacity can be further improved compared to conventional cellular networks, in which devices must use cellular uplink/downlink channels to communicate through BSs. Additionally, DUs may also share the cellular spectrum by using the same physical resources with CUs as long as harmful interference to CUs is prevented, hence, the spectrum can be utilized in a more efficient manner [9]. Second, the close proximity between the DUs yields high data rates and improves the overall throughput without needing to use high transmit powers, which is both energy efficient and beneficial for interference management [10]. Third, the connectivity of CUs which have poor signal reception from the BS due to path loss and low Signal-to-Interference-plus-Noise Ratio (SINR), can be significantly improved by using D2D communications for device relaying and range extension [11].

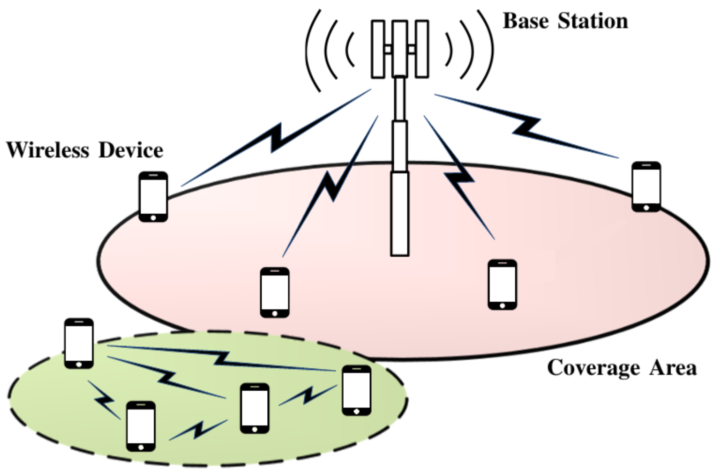
Depending on the cellular network coverage and the received signal strength at devices, D2D networks deployment scenarios can be divided into three main categories, namely, in-network coverage, partial network coverage and out-of-coverage as shown in Fig. 1.1. In the case of the in-network coverage scenario, similar to a conventional network setting where User Equipment (UE)s communicate with a BS by using uplink/downlink channels, DUs can communicate not only with the BS but also exchange data directly with each other by establishing sidelinks as illustrated in Fig. 1.1 (a). Nonetheless, some of the devices may be located outside the coverage area of the BS, or due to low SINR they cannot maintain a connection with the BS with the desired Quality of Service (QoS). For example, if the average received SINR from the BS at an UE is less than –6 dB, then those devices are considered to be out-of-coverage [12]. However, some devices within the coverage of the cellular network can act as relaying devices via D2D communications and extend the network coverage as illustrated in the partial-network coverage scenario in Fig. 1.1 (b). Finally, in the out-of-coverage scenario shown in Fig. 1.1 (c), none of the devices can maintain a connection with a BS or a relaying device inside the coverage of a cellular network. However, such devices that are in close proximity can still communicate with each other via side links by forming a D2D network.



(a) In-network coverage scenario.



(b) Partial-network coverage scenario.



(c) Out-of-coverage scenario.

Fig. 1.1. Illustration of different D2D network (area with dashed-line) deployment scenarios with respect to the cellular network (area with solid line).

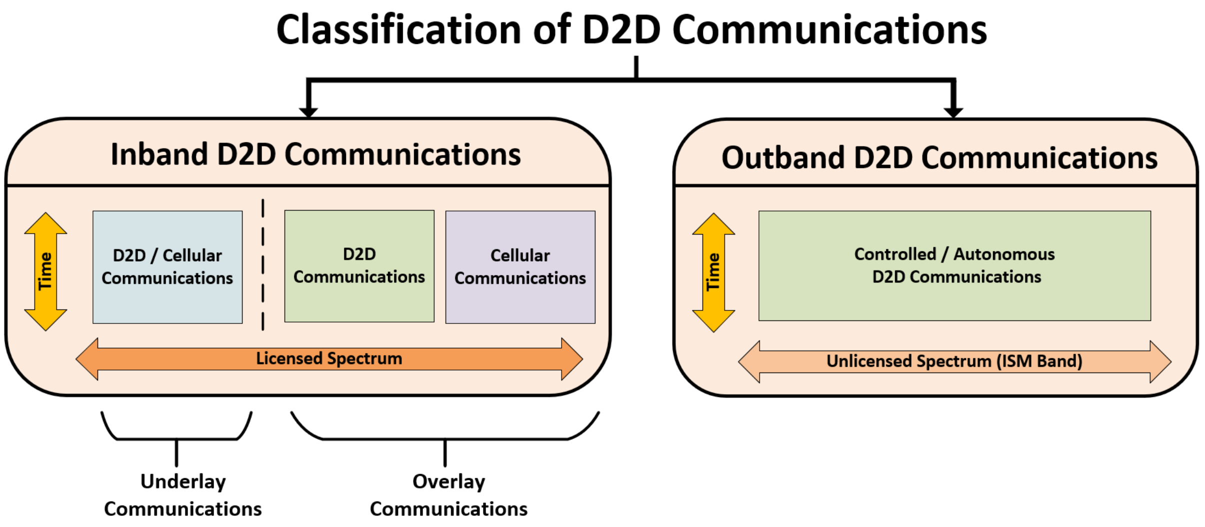

Classification of D2D Communications
Inband D2D Communications
Outband D2D Communications
Time
D2D / Cellular Communications
D2D Communications
Cellular Communications
Licensed Spectrum
Time
Controlled / Autonomous D2D Communications
Unlicensed Spectrum (ISM Band)
Underlay Communications
Overlay Communications

Fig. 1.2. Spectrum utilization of D2D communications.

D2D communications can be further classified into two groups depending on the spectrum utilization strategy as shown in Fig. 1.2 [13].

- *Inband* D2D communications: DUs operate within the licensed cellular spectrum and depending on the its utilization, inband D2D communications are further divided into two sub-classes:
  - *Inband-Underlay* D2D communications: In the licensed band, CUs are Primary Users (PUs) and DUs are Secondary Users (SUs) [14]. DUs share the same time and frequency resources with CUs to increase spectrum efficiency as long as they do not cause harmful interference to them [15–20]. In the literature, various interference mitigation techniques are studied such as mode selection for determining *uplink/downlink* or *sidelink* connection [21–23], power control [24, 25], Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) based random access schemes [26], and space diversity techniques [27]. If the QoS of CUs cannot be maintained while co-existing with DUs, D2D communications should be moved to different time/frequency resources or terminated.
  - *Inband-Overlay* D2D communications: Some portion of the licensed cellular spectrum is allocated to D2D operation depending on availability [28]. Thus, harmful interference to CUs is avoided, however, the spectrum is not efficiently utilized. In addition, D2D communications should be terminated in the case of the allocated

time and frequency resources are requested by CUs as they are the PUs in this band.

- *Outband* D2D communications: DUs can also operate outside of the licensed spectrum, e.g., in the Industrial, Scientific and Medical (ISM) band, hence the name [29]. Although the interference among D2D and cellular networks is averted similar to inband overlay D2D communications, interference within D2D networks is possible [30]. Outband D2D communications can be further divided into two sub-categories, namely, *controlled* and *autonomous* D2D communications [13]. In the *controlled* D2D communications, a BS monitors and controls the D2D communications links, whereas in *autonomous* mode, DUs are responsible to distributively optimize their communication links.

1.2 Applications of D2D Communications

While 5G networks are currently being deployed in various parts of the world, the standardization of D2D communications was initiated in 2012 by 3GPP in Release 12 under the name of Proximity Services (ProSe) also known as LTE-Direct [31]. The main focus of ProSe is the discovery and direct communication of mobile devices in close proximity for public safety applications. To this end, the LTE-A networks in the U.S. have reserved a bandwidth of 20MHz in the 700MHz frequency band for such applications [32,33]. In the event of natural disasters and emergency situations, where the cellular network service may be partially or completely disrupted, D2D communications, especially the out-of-coverage network setting, provides fast localization and intervention for mission-critical operations executed by emergency dispatchers, fire-fighters, police officers and paramedics [32]. In addition, the high data rates and ultra-low-latency due to utilization of D2D communications also improve the QoS in such applications by offering high precision geopositioning, accurate navigation, and real-time indoor mapping [34]. Finally, time-critical emergency alerts such as early tsunami and earthquake detection warnings can be effectively relayed among proximate wireless devices without the need of a cellular infrastructure.

D2D communication is a key technology to enable the Internet of Things (IoT), which describes a network of connected physical objects equipped with sensors, transceivers and software to exchange data over the internet [35]. In addition to the public safety aspects

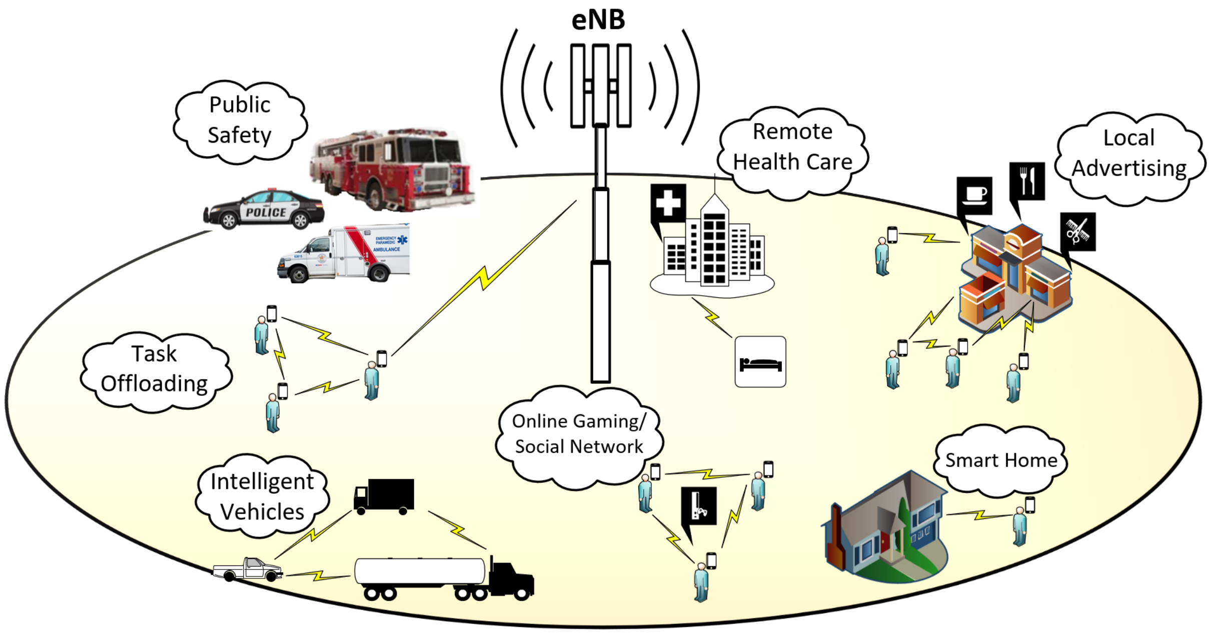


Fig. 1.3. Ubiquitous applications of D2D communications.

of D2D communications, some appealing new use cases for non-public safety applications are remote health care, smart home technologies, online gaming, local advertising, and task offloading as illustrated in Fig. 1.3. In remote health monitoring, elderly people who are living alone or patients equipped with on-body sensors can be monitored constantly without needing to be present in hospitals [13]. Similarly, smart home technologies allow residents to remotely control and manage their home appliances, heating, lighting, and security with ease by using their mobile devices [36]. Furthermore, D2D communications enable augmented reality, P2P interactive gaming, and real-time content sharing such as local advertising by offering ultra-low latency and high data rates [37–39]. However, compared to the proliferation of these applications along with their exacting computation and storage requirements, the technology on the device side has generally advanced slower, which challenges the capabilities of mobile devices as they can hardly meet such demands [40–42]. Therefore, in the past few years, the idea of *task offloading*, which allows devices with computation-intensive tasks to utilize more powerful dedicated servers and/or nearby idle mobile devices through incentive policies for task processing as illustrated in Fig. 1.4, has attracted a great deal of attention [43–45]. Hence, the computation burden on mobile devices can be significantly reduced while improving the overall energy efficiency [46, 47].

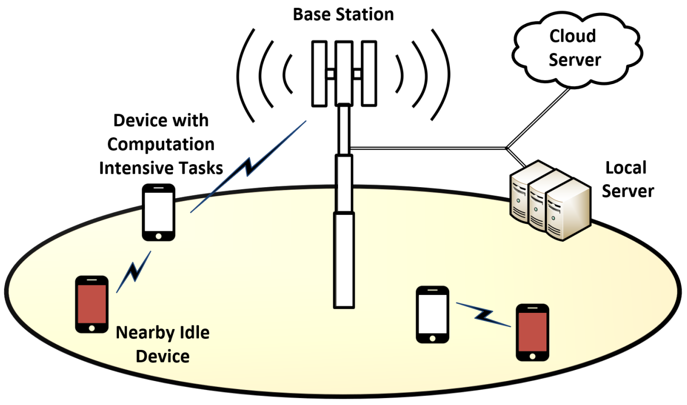


Fig. 1.4. Illustration of a D2D-aided task offloading scenario.

Moreover, in Release 14 of the 3GPP specifications, Vehicle-to-Everything (V2X) communications is considered, which allows intelligent vehicles to communicate not only with other vehicles in their vicinity but also with other smart devices in their surroundings for safer commuting. As illustrated in Fig. 1.5, intelligent vehicles and pedestrians with mobile devices can form D2D connections and communicate with each other at intersections, traffic lights and cross-walks to prevent accidents. This specific type of D2D communications forms a Vehicle-to-Pedestrian network as referred in Release 16 of the 3GPP specifications to support advanced use cases such as helping people with vision and hearing impairments for safer commuting [48]. V2X networks are expected to be further developed for enhanced autonomous driving, extended network coverage via device relaying, industrial applications and virtual reality for B5G networks [49].

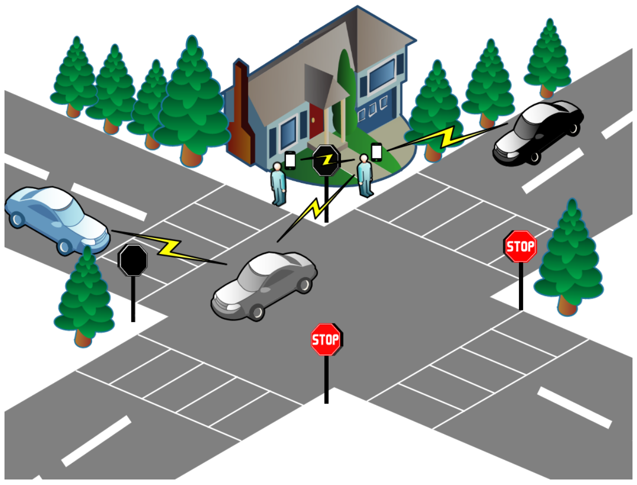


Fig. 1.5. Use-case of V2X and D2D communications in a busy intersection.

1.3 Thesis Objectives and Contributions

The main objective of this thesis is to develop efficient signal processing and optimization strategies for the implementation of reliable D2D communications as envisioned to be a part of 5G and B5G networks. In this regard, we address two main challenges towards the realization of D2D communications, namely, synchronization and resource allocation. Since synchronization is the fundamental block of high data rate communications, our first objective focuses on implementing fast and robust clock synchronization algorithms to allow the initiation of reliable data communications. After investigating the synchronization problem, we then turn our attention to the problem of energy-efficient resource allocation in a D2D-aided task offloading scenario as reducing the energy consumption on battery-driven mobile devices is crucial. Our next objective is to develop optimal resource allocation strategies to achieve energy-efficient task offloading via D2D communications. The main contributions and the research outcomes of the thesis are summarized as next.

In Chapter 3, we propose a scalable, robust and energy-efficient synchronization protocol for D2D communications, specifically for the out-of-coverage scenario. Since no common reference time is available in such a scenario, achieving and maintaining global synchronization is challenging. In comparison to distributed synchronization algorithms in the literature, such as synchronization in Wireless Sensor Networks (WSNs), in which information usually flows towards a single end-node called *sink*, our objective differs as follows. First, the main goal of synchronized D2D communications is different as data transmission is opportunistic and can be established between any proximate devices. Therefore, in D2D networks, synchronization should be achieved not with respect to a single common device but globally. However, several factors impact the synchronization performance. For example, the duplexing scheme, i.e., half-duplex or full-duplex, drastically changes the synchronization time. Full-duplex technology allows simultaneous signal transmission, however, it is not practical yet to implement on mobile devices in terms of cost and energy efficiency. Furthermore, physical factors such as clock skew, propagation delays and dynamic network size continuously degrade synchronization. For a D2D synchronization algorithm to be robust and resilient it should consider the above-mentioned challenges and mitigate their effects. Second, in distributed D2D networks, devices should be aware of the synchronization status of others in order to initiate data communications, therefore, the synchronization phase should be successfully terminated at all devices, ideally at the same time, to allow devices to communicate. To address all of the aforementioned issues, we start by developing a half-duplex timing-advance synchronization algorithm wherein each device alternates between being a transmitter and a receiver in its exchange of synchronization pulses at each clock period. Based on this algorithm, we propose a novel fully-distributed pulse-based synchronization protocol for half-duplex D2D communications. The protocol allows participating devices to become aware of the global synchronization status, so that they can complete the synchronization process simultaneously and proceed to data communications. The proposed protocol is robust against possible perturbations and performs well in comparison to an existing benchmark from the literature over a wide range of conditions such as multi-path frequency selective channels, clock skew, dynamic number of devices, and different network topologies.

After implementing the full-fledged synchronization protocol, in Chapter 4 we address the energy-efficient resource allocation problem in a D2D-aided task offloading scenario with respect to task partitioning, computation resources and transmit power. However, processing computation-intensive tasks at mobile devices might lead to Central Processing Unit (CPU) throttling to control the operating temperatures. Consequently, task processing times become random, which negatively affects the task offloading performance in terms of QoS and energy efficiency. To address this issue, we formulate the resource allocation problem to minimize the expected total energy consumption under probabilistic constraints on the processing time. However, the formulated problem is non-convex and intractable, hence, we propose two sub-optimal solution methods to efficiently solve it. The first method is based on Difference of Convex (DC) programming, where we adopt chance-constraint optimization for the probabilistic constraints on the task processing time to obtain their deterministic equivalents. Considering that DC programming is dependent on a good initial point, we propose a second method that relies only on convex programming. Simulation results demonstrate that both proposed methods significantly improve the energy efficiency when computing the same task compared to the total energy consumption when the task is completed locally. However, the second method outperforms the first in terms of energy efficiency and run-time.

Chapter 5 complements the task offloading scenario in Chapter 4 by considering multiple devices with computation-intensive tasks, which can be offloaded not only to proximate wireless devices via D2D links but also to a more powerful, central computation server, referred to as Edge Server (ES). On the one hand, the ES has more computation capability compared to the mobile devices, and hence can compute more tasks simultaneously; however, data rates on the ES links may limit the task uploading speed. On the other hand, D2D-aided task offloading framework can take advantage of close proximity to yield higher data rates and reduce the task offloading time. However, in comparison to Chapter 4, we assume that the task processing time at each device is deterministic due to the additional complexity imposed by handling the probabilistic constraints and jointly allocating the computation resources at the ES. The considered problem is non-convex and finding its global optimum is generally intractable; hence we propose two sub-optimal methods to solve it. First, by investigating the relationship between the task processing time and the total energy consumption, we show how the original problem can be relaxed into a sequence of convex subproblems whose solutions can be efficiently obtained via standard algorithms from convex optimization theory. Second, to further reduce computational complexity, we propose a low-complexity heuristic resource allocation strategy which does not require calculating gradients and Hessian matrices in the solution process. We compare both proposed methods against two benchmarks. Similar to Chapter 4, for our first benchmark, we consider the total energy consumption when the task is completed only locally, which acts as an upper-bound on the task offloading performance in terms of energy efficiency. For the second benchmark, we calculate the total energy consumption in an ideal task offloading scenario to be used as a lower-bound. Computer simulations under a wide range of conditions and parameter settings (number of fog devices, task sizes, computation resources, transit power, etc.) show that both methods significantly reduce the total energy consumption compared to the upper-bound while attaining near optimal solutions in comparison to the lower-bound.

1.4 Thesis Organization and Notations

The rest of the thesis is organized as follows. Chapter 2 includes a detailed literature survey on the technical challenges facing the realization of D2D communications, and provide background on numerical optimization. Chapter 3 presents the proposed energy-efficient synchronization protocol for D2D networks. Chapter 4 focuses on the resource allocation problem in task offloading via D2D communications subject to the probabilistic task processing times due to CPU throttling. Chapter 5 complements the task offloading scenario by considering a multi-device setting under the supervision of a centralized node, which not only orchestrates the devices but also participates in task processing to increase energy efficiency. Finally, Chapter 6 gives some concluding remarks and discusses possible future research directions to extend and develop the work conducted in this thesis.

In the thesis we use the following notations: Boldface capital letters (e.g., $\mathbf{A}$) denote matrices, boldface letters (e.g., $\mathbf{a}$) denote column vectors while $[\cdot]^\top$ represents the transpose, $\|\cdot\|_p$ is ℓ_p norm, $*$ denotes convolution, ∇ indicates the gradient operator and $(\cdot)^{-1}$ is inverse. Furthermore, sets are denoted by calligraphy letters (e.g., $\mathcal{N}$) and $|\mathcal{N}|$ represents the cardinality of sets. Finally, $\mathbb{E}[\cdot]$ is the expectation operator and $\mathbb{P}(\cdot)$ is the probability of an event.

Chapter 2

Background and Literature Review

In this chapter, we first present a comprehensive survey on the technical challenges for the realization of D2D communications, among which we focus on two crucial aspects: synchronization and task offloading. Then, we give a detailed literature review on these elements to motivate our work. Finally, we present some of the optimization techniques useful to construct the algorithms in the thesis.

2.1 Technical Challenges of D2D Communications

In spite of the numerous advantages of D2D communications, this new technology also raises several questions of its own, e.g., how and when to initiate device discovery, synchronize proximate devices, and allocate physical resources in a most energy-efficient way while not creating interference between DUs and CUs [5]. During data communications, conservation of energy and optimization of battery life of mobile devices is essential, hence, energy-efficient resource allocation strategies are crucial. Furthermore, mode selection, i.e., network assisted or standalone D2D communications [50–52], preventing eavesdropping on D2D links and maintaining data security are some of the other important factors to be taken into consideration for robust D2D network deployments [21, 53, 54].

To achieve reliable and high data rate communications in D2D networks, synchronous data transmission is essential and preferable over asynchronous communication [55–57]. Therefore, the challenges of achieving synchronization requires more attention as it is the first step and the fundamental block of implementing realistic D2D communications [58].

Depending on the different D2D network deployments, from the synchronization perspective, out-of-coverage scenario is the most challenging one since no common reference time is available [59, 60]. Even though some devices become the synchronization broadcaster in the network based on pre-determined algorithms [61], the other devices may not be aware of these devices if they leave the network, which leads to frequent initialization of leader electing algorithms, hence, drains device batteries sooner. Moreover, a synchronization algorithm should take environmental disturbances, i.e., multipath channels, into account while being scalable to accommodate dynamic device numbers as DUs can arbitrarily join or leave D2D networks.

In the literature, early contributions for realization of D2D communications considers ideal conditions while putting less emphasis on practical constraints, e.g., disregarding propagation delays and signal time of arrival [59, 62–64]. However, in multipath environments, the propagation delays yield in asynchronous clocks even though the clocks are perfectly synchronized at the beginning of data communications. In addition, considering the nature of D2D networks, a synchronization algorithm should be fast and energy efficient. Ideally, synchronization procedure should take the minimal time while consuming the least possible energy from the device's battery [65]. Hence, the devices could be ready to transmit/receive data packets without losing time whilst efficiently utilizing their battery life. To the best of our knowledge, there is no energy-efficient synchronization algorithm concerning out-of-coverage D2D networks in the literature.

In addition, duplexing mode (half-duplex vs full-duplex) puts additional constraints on communication systems. Since full-duplex is not yet a mature technology especially at the device side, half-duplex communication scheme should be considered for practical D2D network implementations [62, 63]. However, for distributed networks, where there is no common reference time, half-duplex scheme limits data transmission as the devices need to align their frames independently to minimize packet collisions. Furthermore, when there is no dedicated frequency band for synchronization (which is very likely for D2D communications operating underlaying cellular networks), then both synchronization and data communication in half-duplex scheme should be considered together.

Once synchronization is successfully achieved, DUs can proceed to data communications. However, efficient utilization of the physical spectrum while meeting the demand for high data rates and ultra-low latency is challenging due to scarce resources. Therefore, physical resources should be efficiently allocated to maintain QoS, which is desired by network operators to facilitate a profit [66]. In addition, processing computation-intensive tasks that demand ultra-low latency, e.g., online gaming, push the limits of device capabilities, which in turn, increase the total energy consumption on the device side [67]. Even though offloading these tasks to more powerful servers or proximate devices via D2D links yields energy efficiency, the allocation of limited transmit powers, task sizes and computation resources subject to strict task processing times is a difficult optimization problem [68–70].

Furthermore, processing computation-intensive tasks with strict deadlines at mobile devices is challenging since the thermal management on device-side is not advanced to handle high operating temperatures [71]. Compared to cloud services, in which data centers rely on efficient cooling technologies, mobile devices are likely to suffer CPU throttling during task offloading [72]. Since this performance reduction in device CPUs is random depending on the on-chip temperature readings, the time it takes to process tasks also becomes random, which in turn negatively impacts the overall task offloading performance [73].

In what follows we present the background information and a detailed literature survey on clock synchronization and resource allocation problems based on the aforementioned challenges to motivate our research.

2.2 Synchronization in D2D Networks

In this section, we present the technical challenges facing the implementation of a realistic synchronization algorithm for D2D communications. Specifically, we focus on the availability of a common clock, design layer options such as Physical Layer (PHY) or Medium Access Layer (MAC), and finally, transitioning from synchronization to data communication, which is the main goal of synchronization.

2.2.1 Centralized vs Distributed Synchronization

Synchronization is an essential step in establishing a connection in a digital communication system. For reliable data transmissions, all entities in a communication network must become synchronized and remain in this state as long as they operate [58,65]. In cellular networks, a fixed Access Point (AP) such as a BS regularly broadcasts a time signal so that user devices can synchronize themselves to this common time reference as in Fig. 2.1. Such centralized schemes not only offer fast synchronization but also easily maintain it by regularly correcting the device clocks, which can diverge due to clock skews. In addition, the same AP can coordinate the devices during data communications to compensate for propagation delays by adjusting their clocks, a technique known as timing-advance.

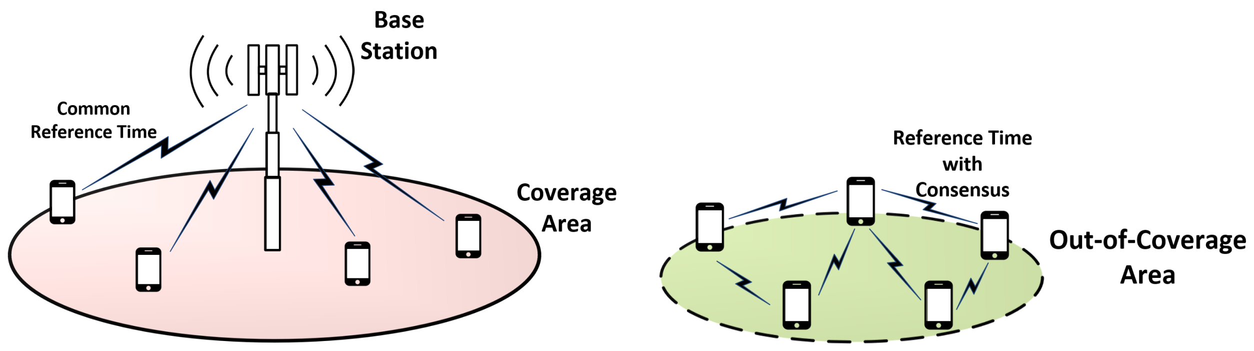


Fig. 2.1. Centralized synchronization (left) versus distributed synchronization (right), where a common reference time is broadcasted from the BS or the distributed devices, respectively.

However, in distributed systems such as WSNs or out-of-coverage D2D networks, the latter being one of the use case for D2D communications, no such fixed AP is available as shown in Fig. 2.1. In this type of scenarios, the aforementioned benefits of centralized synchronization can be retained by selecting, e.g., via an "elect-a-leader" algorithm [61], one of the devices to serve as the synchronization AP. Nonetheless, the synchronization performance depends heavily on the choice of the AP and on the quality of the channels between the AP and the rest of the devices. In addition, if the AP loses connectivity with a part of the network or leaves the network, then the AP selection process should be re-initiated.

An alternative approach is distributed synchronization, wherein devices exchange synchronization signals according to a predefined strategy, or protocol, allowing them to reach a consensus on a common reference time [59]. Although typically slower than centralized synchronization, distributed synchronization is more robust against connectivity failures and network changes due to mobility. Hence, it may be better suited for WSNs or out-of-coverage D2D networks where such conditions are prevailing [58]. Nevertheless, the technical aspects of a network determine the design of the synchronization algorithm. Specifically, in a typical WSN, information flows towards a single sink node, while data communication is low rate and sporadic with relaxed guarantees in terms of latency and reliability. In contrast, D2D communications by nature requires high data rate, low-latency and reliable communication between arbitrary devices among which the communication links are P2P. Therefore, to satisfy these requirements, distributed D2D networks must employ a reliable, fast and efficient synchronization algorithm, which should also mitigate the effect of propagation delays and multipath channels by properly using timing-advance [56].

2.2.2 Synchronization Techniques and Duplexing Scheme

The prevalent approaches for distributed synchronization can be divided into two main categories, namely: packet-based and pulse-based [58]. Comparing the two, Packet-based synchronization is a MAC layer-based approach relying on the exchange of timestamps encoded in packets [74–77]. It requires collision-free transmission of the packets on a random access channel and their subsequent successful decoding. As such it suffers from delays due to packet queuing and re-transmissions and, more importantly in the context of D2D communications, it exhibits high energy expenditure, high latency and poor scalability. Pulse-based synchronization, in contrast, is a physical layer-based approach where the timing information is encoded in the transmission time of physical layer pulses. Hence, it can successfully achieve frame synchronization with high precision while offering scalability [78]. Local clock updates are done by processing the received superposition of timing pulses transmitted by neighbouring devices. This approach, which naturally capitalizes on the broadcast nature of wireless channels, can overcome the above-mentioned limitations of packet-based approach. Thus, pulse-based is often preferred over packet-based synchronization in distributed wireless networks, especially for the initial network synchronization since the packets cannot be demodulated due to asynchronous clocks. [58, 59]. However, after successful initial synchronization, high data rate packet-based communication should be initiated.

Pulse-based synchronization is typically implemented by Distributed Phase-Locked Loops (DPLLs) [63, 79]. The performance of DPLLs is affected by the duplexing scheme employed by the devices. Full-duplex technology allows simultaneous signal transmission and reception, which is highly beneficial for synchronization as clock information is broadcasted and updated at the same time. Naturally, full-duplex communication significantly decreases the synchronization time compared to half-duplex, and has been considered by several authors [59, 64]. However, implementation of full-duplex technology, especially at mobile devices poses a number of practical issues in terms of cost, complexity and power consumption [62,63,79]. Due to the additional power required for self-interference cancellation mobile devices with a limited battery life cannot currently afford to operate in the full-duplex mode [80]. Hence, half-duplex communication is a more practical implementation choice for distributed networks and is likely to remain so in the near future.

2.2.3 Clock Skew and Propagation Delays

Random clock initializations (i.e., clock phase difference) lead to Time-Offset (TO), which is interpreted as the clock synchronization error. Nevertheless, various other factors also introduce TO, such as the quality of the crystal oscillators whose frequency may drift with temperature fluctuations. This effect, known as clock skew, not only alters the perceived rate of signal transmission and reception arbitrarily over time but also causes Carrier Frequency Offset (CFO), which may severely affect data communications if not compensated [81]. Even though specially designed synchronization signals can deal with the presence of TO and CFO in stationary networks [81], signal propagation time is another factor that imposes difficulty on synchronization by introducing additional TO. Especially for pulse-based synchronization techniques that use DPLLs, the effects of clock phases, clock skews, and propagation delays are entangled within the superimposed timing pulses. Hence, to achieve synchronization, the aforementioned effects should be estimated from the received pulses and removed by updating the device clocks accordingly [79]. However, this estimation becomes highly challenging as the received pulses are not only altered by these effects, but also by the very mechanisms used to correct them in DPLLs, which could lead to instability.

In the literature, it is often assumed that: the device clocks are frequency synchronized (i.e., there is no clock skew) [59, 63, 64, 79, 82], there are no propagation delays, and the network size (i.e., the number of devices) is static [83]. For example in [81], joint time and frequency synchronization for distributed networks is proposed, however, propagation delays are not considered. The convergence of clock skew and clock phase offset is investigated in [83] but environmental factors, such as radio propagation delays and changes in the number of connected devices, are not taken into consideration. However, signal propagation delays do exist and D2D devices can arbitrarily join or leave the network; hence, these assumptions are not valid in a realistic scenario. In [84], a synchronization method is proposed, in which clock skew and clock phases are corrected with respect to a selected reference node which does not participate in the synchronization process. Another approach is proposed in [64], where the devices first estimate the propagation delays to their neighbors and then transmit these estimates to a centralized fusion center; in turn, the center informs all the devices about the delay estimates within the network. Consequently, synchronization is achieved after pre-compensation for propagation delays is applied known as Timing-Advance (TA) timing-advance. In the absence of a centralized fusion center as, for example, in an out-of-coverage D2D network, such global time-advance compensation is impractical. Clock phases, clock skews and propagation delays in this case should be jointly estimated in a distributed manner, thereby allowing devices to synchronize their clocks individually by means of timing-advance.

2.2.4 Transitioning from Synchronization to Communications

From the energy efficiency and performance perspectives it is beneficial to stop the synchronization process at all devices at the same time and, ideally, as soon as they become synchronized. However in a distributed D2D network, the devices are not aware of the synchronization status of others and the time it takes to reach synchronization might vary for each device. If some of the devices stop their synchronization process later than others, the presence of ongoing timing pulses might trigger the synchronized devices to re-start this process [79]. Thus, in a fully distributed wireless network, the participating devices should be aware of the global synchronization status, so that they can terminate the synchronization process ideally at the same time and proceed to data communications.

Once the synchronization process is completed, some of the devices might become idle to conserve energy, while others might begin data communications. In this case, the idle devices may lose synchronization with respect to the communicating ones since there is no mechanism to inform them about their status. To prevent this, the devices must periodically re-transmit timing pulses, i.e., discovery beacons, to remain part of the network [65, 85]. However, this will cause a disturbance among the synchronized devices and increase energy consumption in the overall network. Hence, synchronization should be maintained as long as possible without exchanging timing pulses to reduce the frequency of unnecessary re-initialization.

2.3 Task Offloading via D2D Communications

In this section, D2D communication technology is reviewed as an enabler of the task offloading framework. To this end, we present both the advantages and the technical challenges of using D2D communications in task offloading.

2.3.1 Task Offloading Strategies

Cloud computing, one of the most studied task offloading strategies, allows user data to be stored and processed at remote data centers through the internet [86, 87]. However, unpredictable wireless channel conditions and high data traffic due to excessive user density limit the overall task offloading speed, which in turn, reduce the QoS [66]. Mobile Edge Computing (MEC) as a substitute to cloud computing, brings data processing at the edge of the network by allowing mobile devices to offload their tasks to ESs, which are usually connected to BSs via high-speed links [86–89]. Therefore, MEC can effectively increase energy efficiency by reducing end-to-end delays and data processing bottlenecks within the network, which enables many use-cases such as real-time video rendering, surveillance, online gaming and task offloading [90–92]. In [67], the authors consider joint task offloading and resource allocation in a MEC scenario, where a centralized mechanism allocates the available resources to minimize total overhead and energy consumption. The authors in [93] focus on total energy minimization subject to constraints on the task processing times in a ultra-dense network. In [69], the authors propose a resource allocation method for MEC aiming to minimize the maximum system delay, while in [94], energy consumption, task execution delay and cost of task offloading are jointly optimized by using queuing theory. In [70], energy-efficient resource allocation is investigated for latency-sensitive tasks by considering a multi-user offloading scenario. In [95], the authors study a similar scenario, in which offloading decisions are now modeled by using a game-theoretic approach. Moreover in [96], a distributed power allocation method is proposed, while in [97], minimization of energy consumption and average response time is investigated based on game-theory. While shedding light on the utilisation of MEC, the aforementioned studies only consider a single off-loading destination, such as the ES. However, the limited computational capability of ES (compared to the cloud) limits the energy efficiency and restrains the performance of task off-loading as data traffic increases within the network.

2.3.2 Mobile Fog Computing

As a complement to MEC and cloud computing, fog computing can exploit the full potential of distributed data processing by allowing data-generating devices to offload their tasks to nearby computation resources through incentive policies [98–101]. This strategy can significantly decrease the total energy consumption and task processing times, by leveraging the availability of all computation resources within radio proximity, referred to as *fog devices* in this context. D2D communications provide an attractive technology enabler for fog computing, by facilitating the creation of direct communication links between neighboring devices. In [102], the authors integrate D2D communications with MEC, aiming to maximize the number of devices supported by the cellular network with constraints on both communication and computation resources. Likewise in [68], the authors study computation latency minimization in the case of D2D-enabled MEC system, while in [103] the authors focus on time-average energy minimization. Nonetheless, due to the difficulties posed by the optimal allocation of communication and computation resources as the number of offloading destination increases, the above studies only consider a single D2D connection within a MEC system. However, restricting the number of D2D connections limits the potential gains in capacity and energy-efficiency of the task offloading. Therefore, a scalable approach, in which an arbitrary number of fog devices is supported by multiple D2D connections in a MEC scenario is needed.

Another factor that directly affects the energy consumption is the very nature of the task offloading scheme, i.e., whether it is binary or partial. In the former case, a task is either computed locally (i.e., on the data generating device) or offloaded entirely to a single neighboring device, while in the latter case, a task is divided into various portions for parallel computing on different devices, including the local one. In [104], binary task offloading with multiple ES is studied to jointly optimize the network access selection subject to constraints on time delays and QoS. In [105], the authors consider binary task offloading in a D2D-assisted fog computing scenario, and propose new algorithms based on branch-and-price for jointly optimizing link scheduling, channel assignment and power control. In [103], the authors propose an optimization framework based on binary offloading in order to minimize the time-average energy consumption for execution of all user tasks while taking into account key incentive constraints. Likewise in [106], the overall system utility is maximized with respect to binary offloading decisions by developing a pricing game-based algorithm.

In contrast to the binary scheme, partial task offloading can take advantage of parallel computation towards increasing the energy efficiency. In [70], both partial and binary task offloading schemes are investigated and it is shown that partial offloading is beneficial in terms of energy efficiency. In [107], partial task offloading in MEC is considered, where the aim is to minimize the total energy consumption by jointly optimizing the transmit power, computation speed and task partitioning. In [108], the authors consider a cooperative partial task offloading scheme with both cloud and MEC computing, wherein task partitioning decisions are made by minimizing the end-to-end delay. While offering considerable insights into the partial task offloading, these works do not take advantage of D2D-aided fog-computing to further improve energy-efficiency and reduce delay. In contrast, references [109–111] explicitly focus on partial task offloading with the help of D2D communications and fog-computing. In [109], a hybrid D2D-aided fog and cloud computing scenario with a single task offloading device is investigated, for which the total energy consumption is minimized subject to constraints on task processing time. Whereas the work in [110] seeks to find an optimal network partition in a multi-device D2D-aided fog computing scenario by minimizing the overall energy consumption at the mobile terminal side. Similarly in [111], the authors minimize the total energy consumption under delay constraints, by jointly optimizing of the task scheduling and computation resources. Although these works achieve significant improvement in energy-efficiency by using D2D-aided fog-computing, they do not consider the allocation of transmit powers and utilization of an ES, which could further enhance the performance of the task offloading.

2.3.3 Effect of CPU Throttling on Task Processing Time

In mobile devices, Dynamic Thermal Management (DTM) schemes control the on-chip temperature by lowering the voltage and frequency of the CPU to prevent damage in the case of high temperature [71, 112, 113]. Ideally, devices allocate the highest available CPU frequency, measured in cycles per second, to perform a given task within a minimum amount of time [114]. Due to DTM, however, significant yet unpredictable fluctuations in allocated CPU frequency do occur [73]. Since real-time applications require low latency and strict processing time, a random reduction in CPU frequency negatively impacts task offloading. Consequently, to optimize fog computing performance subject to this type of uncertainty, allocation of computation resources should be treated as a probabilistic optimization problem rather than a deterministic one.

2.4 Background on Numerical Optimization

In this section, we present some of the numerical optimization techniques that are used as a basis to develop the proposed methods and the algorithms given in the thesis.

2.4.1 Convex vs Non-Convex Optimization

An optimization problem is convex if the feasible set, i.e., the set of all possible points that satisfy all constraints, is a convex set and the objective function is a convex function [115]. Specifically, a set $\mathcal{S} \in \mathbb{R}^n$ is convex, if the straight line segment connecting any two points $x \in \mathcal{S}$ and $y \in \mathcal{S}$ is also in the set $\mathcal{S}$, i.e., $\alpha x + (1-\alpha)y \in \mathcal{S}, \forall \alpha \in [0,1]$. Similarly, the function f is convex if its domain $\mathcal{S}$ is a convex set and any two points such as $x \in \mathcal{S}$ and $y \in \mathcal{S}$ satisfies the inequality:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \forall \alpha \in [0,1] \tag{2.1}$$

The standard form of a convex minimization problem with equality and inequality constraints is given below:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{2.2a}$$

$$\text{s.t.} \quad h_i(\mathbf{x}) = 0, \quad i = 1, ..., m \tag{2.2b}$$

$$g_i(\mathbf{x}) \leq 0, \quad i = m+1, ..., k \tag{2.2c}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$ contains the decision variables, the functions $f(\mathbf{x})$ and $g_i(\mathbf{x}), i = 1, ..., k$ are convex and $h_i(\mathbf{x}), i = 1, ..., m$ are affine.

In convex optimization, a local optimum e.g., $\mathbf{x}^\star$ that minimizes (2.2a) and satisfies constraints (2.2b) and (2.2c), is also the global optimum, which can be effectively obtained by using any algorithm from the convex optimization theory [115,116]. Whereas if the objective function or at least one of the constraints is a non-convex function, the problem becomes a non-convex program. Hence, obtaining the global solution for such problem is often impractical or not possible in polynomial time due to computation complexity [117]. However, convex programming can be still useful to find an initialization for local optimization of a non-convex problem by approximating it as a convex problem [115]. Hence, recognizing and formulating non-convex problems as convex problem is a useful tool in non-convex optimization.

2.4.2 Difference of Convex Programming

DC programming is an important tool in non-convex optimization as it provides an efficient and scalable method for smooth/non-smooth non-convex programming [118]. Although any polynomial Difference of Convex Functions (DCF)s with a desired accuracy, the obtained decomposition is not unique and there is no general procedure to construct the optimum DCFs that lead the global solution [119, 120]. However, once the DCFs are available, a non-convex program can be written as a DC program as follows:

$$\min_{\mathbf{x}} \quad f_0(\mathbf{x}) \triangleq h_0(\mathbf{x}) - g_0(\mathbf{x}) \tag{2.3a}$$

$$\text{s.t.} \quad f_i(\mathbf{x}) \triangleq h_i(\mathbf{x}) - g_i(\mathbf{x}) = 0, \quad i = 1, ..., m \tag{2.3b}$$

$$f_i(\mathbf{x}) \triangleq h_i(\mathbf{x}) - g_i(\mathbf{x}) \leq 0, \quad i = m+1, ..., k \tag{2.3c}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$ contains the decision variables, $h_i(\mathbf{x})$ and $g_i(\mathbf{x}), i = 0, 1, ..., k$ are convex functions, hence, the non-convex functions $f_i(\mathbf{x}), i = 0, 1, ..., k$ are a DCFs [119–122]. Then, problem (2.3a) can be solved by using the DC algorithm (DCA), which employs the convex analysis to the two convex parts of the decomposed functions [123].

If the functions $h_i(\mathbf{x})$ and $g_i(\mathbf{x})$ $i = 0, 1, ..., k$ are differentiable, then DCA reduces to Convex-Concave Procedure (CCP) algorithm, which can iteratively solve problem (2.3a) by using the convex program as follows [122]:

Algorithm 2.1 Convex-Concave Procedure

```
1: input Set $k = 0$, initialize $\mathbf{x}^{(0)}$
2: repeat
3:     $\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\text{argmin}} \quad f_0(\mathbf{x}) \triangleq h_0(\mathbf{x}) - g_0(\mathbf{x}^{(k)}) - \nabla g_0(\mathbf{x})^\top(\mathbf{x} - \mathbf{x}^{(k)})$
4:     $\text{s.t.} \quad f_i(\mathbf{x}) \triangleq h_i(\mathbf{x}) - g_i(\mathbf{x}^{(k)}) - \nabla g_i(\mathbf{x})^\top(\mathbf{x} - \mathbf{x}^{(k)}) = 0, \quad i = 1, ..., m$
5:     $f_i(\mathbf{x}) \triangleq h_i(\mathbf{x}) - g_i(\mathbf{x}^{(k)}) - \nabla g_i(\mathbf{x})^\top(\mathbf{x} - \mathbf{x}^{(k)}) \leq 0, \quad i = m+1, ..., k$
6:     $k \leftarrow k + 1$
7: until $|f_0(\mathbf{x}^{(k+1)}) - f_0(\mathbf{x}^{(k)})| > \epsilon$ or $k \leq k^{\max}$
8: output $\mathbf{x}^{(k)}$
```

where k is the iteration number. Similar to many iterative algorithms, there must be a stopping condition, which can be a tolerance value to control the performance variations between the consecutive iterations and accordingly make a decision, or simply a maximum iteration number to terminate the algorithm. Hence, one stopping condition can be set as $|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})| < \epsilon$ with ϵ being a tolerance value or $k^{\max}$ being the maximum allowed iteration number. The main idea is to linearize the concave parts of DCFs, i.e., $-g_i(\mathbf{x}), i = 0, 1, ..., k$, hence, convexify the functions at the kth iteration. Therefore, the non-convex program (2.3a) can be solved as a sequence of convex programs [121]. However, the performance of CCP algorithm is sensitive to the choice of the initial point, i.e., $\mathbf{x}^{(0)}$, as well as the decomposition of non-convex functions, i.e., the construction of DCFs.

2.4.3 Reducing Constrained DC Program with Penalty Theory

Consider the following DC program with equality and inequality constraints:

$$\min_{\mathbf{x}} \quad f_0(\mathbf{x}) \triangleq h_0(\mathbf{x}) - g_0(\mathbf{x}) \tag{2.4a}$$

$$\text{s.t.} \quad f_i(\mathbf{x}) \triangleq h_i(\mathbf{x}) - g_i(\mathbf{x}) \leq 0, \quad i \in \mathcal{I} = \{1, ..., m\} \tag{2.4b}$$

$$f_i(\mathbf{x}) \triangleq h_i(\mathbf{x}) - g_i(\mathbf{x}) = 0, \quad i \in \mathcal{E} = \{m+1, ...k\} \tag{2.4c}$$

which can be reduced into an unconstrained DC program with a penalty parameter $\sigma \geq 0$ by incorporating the constraints into the objective function as follows: [119]:

$$\begin{aligned} F_\sigma(\mathbf{x}) &\triangleq f_0(\mathbf{x}) + \sigma \max\{0, f_i(\mathbf{x}), i \in \mathcal{I}\} + \sigma \sum_{i \in \mathcal{E}} |f_i(\mathbf{x})| \\ &= H_\sigma(\mathbf{x}) - G_\sigma(\mathbf{x}) \end{aligned} \tag{2.5}$$

where $F_\sigma(\mathbf{x})$ is the penalized objective function, $H_\sigma(\mathbf{x})$ and $G_\sigma(\mathbf{x})$ are convex functions given as:

$$\begin{aligned} H_\sigma(\mathbf{x}) = h_0(\mathbf{x}) + \sigma \max\Bigg\{ \sum_{j \in \mathcal{I}} g_j(\mathbf{x}); \Bigg[h_i(\mathbf{x}) + \sum_{j \in \mathcal{I}}^{j \neq i} g_j(\mathbf{x}) \Bigg], i \in \mathcal{I} \Bigg\} \\ + 2\sigma \sum_{i \in \mathcal{E}} \max\{h_i(\mathbf{x}); g_i(\mathbf{x})\} \end{aligned} \tag{2.6}$$

$$G_\sigma(\mathbf{x}) = g_0(\mathbf{x}) + \sigma \Bigg[\sum_{i \in \mathcal{I}} g_i(\mathbf{x}) + \sum_{j \in \mathcal{E}} \big(h_j(\mathbf{x}) + g_j(\mathbf{x}) \big) \Bigg] \tag{2.7}$$

Then, depending on the properties of $H_\sigma(\mathbf{x})$ and $G_\sigma(\mathbf{x})$, i.e., differentiable, the new penalized objective function $F_\sigma(\mathbf{x})$ can be iteratively minimized similar to Algorithm 2.1 as a sequence of convex problems by using any algorithms from the convex optimization theory as follows:

$$\mathbf{x}^{(k+1)} \in \underset{\mathbf{x}}{\operatorname{argmin}} \quad F_\sigma(\mathbf{x}) \triangleq H_\sigma(\mathbf{x}) - G_\sigma(\mathbf{x}^{(k)}) - \nabla G_\sigma(\mathbf{x})^\top(\mathbf{x} - \mathbf{x}^{(k)}) \tag{2.8}$$

2.4.4 Chance-Constraint Optimization

Chance-constraint optimization focuses on solving optimization problems subject to different uncertainties imposed as probabilistic constraints. Therefore, this type of optimization naturally arises in economics, finance, weather forecast, and engineering to ensure that the probability of an anticipated event is above a certain threshold. The standard form of a single chance constraints program is formulated as follows [124]:

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{2.9a}$$

$$\text{s.t.} \quad \mathbb{P}(g_i(\mathbf{x}, \xi) \leq 0) \geq \gamma, \quad i = 1, ..., k \tag{2.9b}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$, ξ is a random variable following some distribution and $\gamma \in [0, 1]$ is reliability level. Hence, constraint (2.9b), which can be interpreted as a *reliability constraint*, ensures that the probability of realizing function $g_i(\mathbf{x}, \xi)$ for the given values is smaller or equal to zero at γ percent of the time.

Major difficulty arises when computing the probability $p(x) = \mathbb{P}(g_i(\mathbf{x}, \xi) \leq 0)$ based on the distribution of ξ. However, depending on the constraint functions $g_i(\mathbf{x})$ as well as the distribution of ξ, the probability $p(x)$ can be calculated by using the Cumulative Distribution Function (CDF) of ξ, i.e., $F_\xi(\cdot)$. Hence, the exact analytical form of constraint (2.9b) can be obtained. Then, depending on the nature of the deterministic equivalent of constraint (2.9b) and the overall problem, the equivalent deterministic optimization problem, i.e., convex vs non-convex, can be solved by using the appropriate methods.

2.5 Summary

In this chapter, we presented the technical challenges and related studies from recent literature in regards to the implementation of realistic D2D communications as conceptualized to be part of future generation wireless networks. Motivated by the aforementioned missing key points, in the next chapter, we propose a distributed synchronization protocol to be the basis of reliable data communications in D2D networks.

Chapter 3

A Distributed Synchronization Protocol for D2D Networks

In this chapter[1], we focus on a clock synchronization problem in D2D networks. To this end, we propose a novel distributed synchronization protocol, which not only achieves fast and robust synchronization by taking realistic conditions, i.e., propagation delays, clock skews, and dynamic network sizes, into account but also allows devices to initiate data communication.

3.1 Introduction

In distributed D2D communications, no common reference time is available and the devices must employ distributed synchronization techniques. In this context, pulse-based synchronization, which can be implemented by DPLLs is preferred due to its scalability. However, several factors degrade the performance of pulse-based synchronization, such as duplexing scheme, clock skew and propagation delays. Furthermore, in distributed networks, devices should be aware of the synchronization status of others, and terminate the synchronization process ideally at the same time to initiate data communications. Otherwise, ongoing synchronization signals may trigger the re-initiation of the synchronization process at some of the synchronized devices, which may lead to an oscillating effect, where the distributed devices cannot leave the synchronization phase.

[1]Parts of the work presented in this chapter have been published in [57, 79, 125].

Motivated by these prevailing issues and the technical challenges given in Section 2.2 of Chapter 2 regarding the need for an efficient synchronization algorithm for D2D communications, we now focus on the development of a fast, scalable and robust distributed synchronization protocol by considering out-of-coverage D2D networks. First, we propose a timing-advance synchronization algorithm wherein each device alternates between the transmitter and receiver modes in its exchange of synchronization pulses at each clock period. Based on this algorithm, we then develop a fully-distributed pulse-based synchronization protocol for half-duplex D2D communications using the channel models for 5G networks. Specifically, our main contributions in this chapter are summarized as follows:

- We propose an online estimation technique which jointly tracks the synchronization errors due to clock phases, clock skews and propagation delays in multipath channels. We incorporate the obtained estimates in the conventional DPLLs clock update and propose a half-duplex timing-advance synchronization algorithm which compensates for *all* these effects in a distributed manner. In particular, we show analytically that the proposed compensation mechanism contributes to reducing the synchronization error at each iteration.

- We conceive a distributed synchronization protocol in the form of a state diagram, which can be easily implemented on each device. In this protocol, the participating devices acquire the synchronization status of others by a coordinated exchange of pulses, and terminate this process as soon as the overall network is synchronized. Our proposed protocol also allows the devices already in operation to detect the presence of new devices joining the network at any time, and to re-synchronize themselves by including the new ones.

- After the network is synchronized, the devices can either initiate data communication or stay idle and only operate as a receiver to conserve power. At this point, to maintain synchronization in the network without exchanging timing pulses, the devices predict their relative clock time based on estimated values of the synchronization parameters. Thus, they can preserve the already achieved synchronization without unnecessarily re-initiating this process.

- The complete integrated protocol is evaluated by means of computer simulations based on 5G channel models and under different conditions of operation, i.e., clock skew, number of devices, and network topologies, including full mesh and partial mesh. Our extended results show that the proposed protocol offers better synchronization performance than a benchmark approach from the current literature [63], even for partial-mesh topology.

3.2 System Model and Problem Formulation

In this section, we first present the overall system model including the network setup, the communication scheme and the clock model. Then, we formally define the clock synchronization problem in out-of-coverage D2D networks by considering the effects of clock skew, multipath channels and duplexing scheme.

3.2.1 Network Setup and Clock Model

We consider a distributed overlaying D2D network, which can be fully or partially connected, with J wireless devices indexed by $j \in \mathcal{J} = \{1, ..., J\}$ as illustrated in Fig. 3.1. We assume that the devices may join or leave the network at any time and that they do not have any information about the network, such as the number of nearby devices or their locations. Since there is no BS to provide a common timing reference, the devices synchronize their clocks in a fully distributed manner by exchanging timing pulses over radio frequencies at the physical layer.

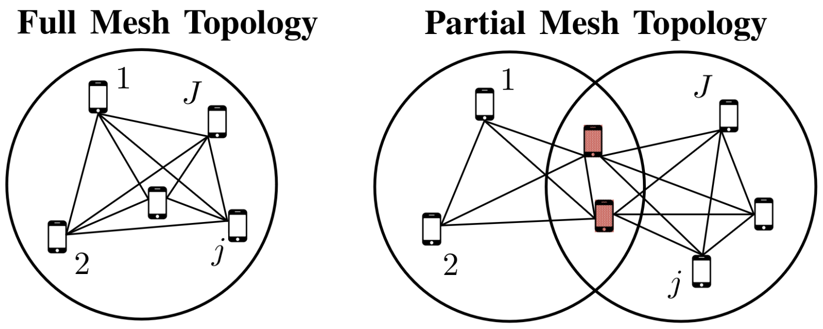


Fig. 3.1. Fully or partially connected D2D networks. In the partial mesh topology, common devices (shown in red), act as a relaying node during synchronization.

The physical clock of the jth device is modeled as $t_j(t) = \alpha_j t + \theta_j$ [126], where t is the universal time, α_j is the clock skew, and $\theta_j \in [0, T_0)$ is the clock phase with T_0 being the clock period. A discrete logical clock is obtained by uniformly sampling the physical clock at times $t = \nu T_0$, that is:

$$t_j[\nu] = t_j(\nu T_0) = \alpha_j \nu T_0 + \theta_j \tag{3.1}$$

where $\nu \in \mathbb{N}$ is the discrete-time index. We refer to time $t_j[\nu]$ as the νth *clock tick* of the jth device. It is convenient to partition the universal time axis into a sequence of non-overlapping time slots $[\nu T_0, (\nu+1)T_0)$. In practice, α_j differs from 1 by a very small amount, on the order of a few parts per million (ppm) [127], and it is therefore safe to assume that each time slot contains a single clock tick $t_j[\nu]$ as shown in Fig. 3.2. In the ideal case of no propagation delay, we define the TO between the ith and the jth devices as the minimum clock tick difference, that is:

$$\Delta t_{ij}[\nu] = \min_{\eta \in \mathcal{V}_\nu} \big| t_i[\eta] - t_j[\nu] \big| \tag{3.2}$$

where for convenience, we define the set $\mathcal{V}_\nu = \{\nu, \nu \pm 1\}$. In effect, $\Delta t_{ij}[\nu]$ can be interpreted (for this ideal case) as the synchronization error between the devices i and j.

3.2.2 Half-duplex Signaling Model

We assume half-duplex communication, where a device can only transmit or receive at any given time. We define the transceiver mode of the jth device at the νth clock tick as $M_\nu^j \in \{\mathrm{TX}, \mathrm{RX}\}$ indicating whether the device operates in transmitter mode (TX) or in receiver mode (RX), as depicted in Fig. 3.2. We let $\mathcal{T}_\nu$ and $\mathcal{R}_\nu$ denote the mutually exclusive index sets of transmitter and receiver devices at the νth clock tick, respectively.

If, at a given clock tick ν, a device operates as a transmitter, i.e., $i \in \mathcal{T}_\nu$, it broadcasts a time-shifted synchronization signal $x(t - t_i[\nu])$ with $x(t)$ defined as:

$$x(t) = \sum_{n=0}^{N_s-1} s[n] g(t - nT_p) \tag{3.3}$$

where $g(t) \in \mathbb{R}$ is a normalized baseband pulse and T_p denotes the pulse spacing with $N_s T_p \ll T_0$. In (3.3), $s[n]$ is a synchronization sequence of length $N_s = 2N$ constructed by concatenating two Zadoff-Chu (ZC) sequences of length N with root indices u and $-u$ [64,81], that is:

$$s[n] = \begin{cases} e^{\mathrm{j}\frac{\pi}{N}un^2}, & 0 \leq n \leq N-1 \\ e^{-\mathrm{j}\frac{\pi}{N}u(n-N)^2}, & N \leq n \leq 2N-1 \end{cases} \tag{3.4}$$

where u and N are coprime, and $\mathrm{j} = \sqrt{-1}$. By constructing the synchronization sequences as in (3.4), the effect of CFO is decoupled from TO estimation [64, 81].

A receiver device $j \in \mathcal{R}_\nu$ listens for broadcasted synchronization signals over the *reception period* $[t_j[\nu] - \frac{T_0}{2}, t_j[\nu] + \frac{T_0}{2})$, centered at its own clock tick $t_j[\nu]$, as illustrated in Fig. 3.2. The received signal at the jth device is:

$$y_j(t) = \sum_{\eta \in \mathcal{V}_\nu} \sum_{i \in \mathcal{T}_\eta} x(t - t_i[\eta]) * h_{ij}(t) + w_j(t) \tag{3.5}$$

where $h_{ij}(t) = \sum_{p \in \mathcal{P}} \rho_{ijp} \delta(t - \tau_{ijp})$ is the impulse response of the multipath channel between the ith and jth device, $p \in \mathcal{P} = \{1, \cdots, P\}$ is the path index, P is the number of resolvable paths, assumed to be the same for all devices, and $\delta(\cdot)$ is the Dirac delta function. Additionally, $\rho_{ijp} \in \mathbb{C}$ and $\tau_{ijp} \in \mathbb{R}_+$ are the complex gain and propagation delay, respectively, of the pth path, while operator $*$ denotes convolution and $w_j(t)$ is an additive noise term. Note that depending on the clock phase of the receiver device, the received signal may contain signal contributions not only from the νth clock tick but also from the adjacent ones, i.e., $(\nu \pm 1)$th. Thus, the outer summation in (3.5) takes all possible signal contributions into account; however, it does not span beyond the $(\nu - 1)$th time slot as the propagation delays are assumed to be much smaller than the clock period, that is, $\tau_{ijp} \ll T_0$.

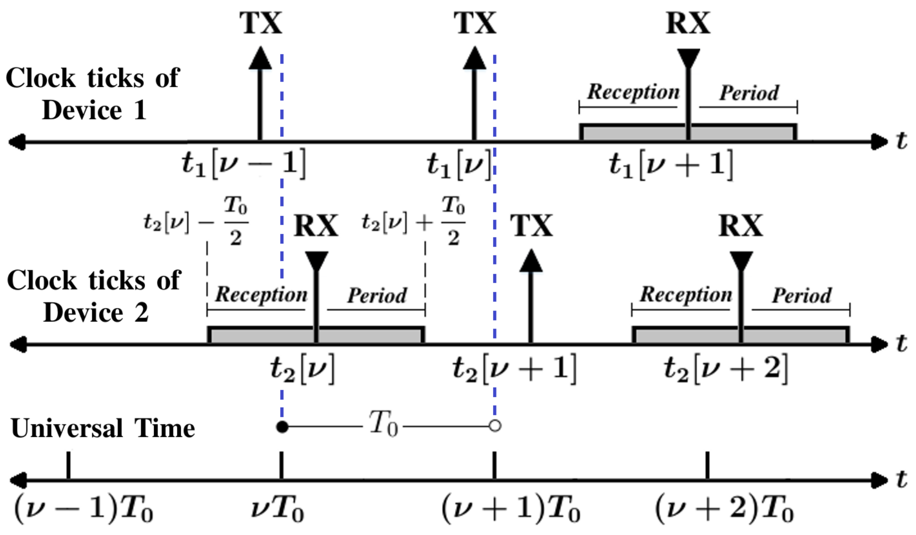


Fig. 3.2. Clock ticks of two devices relative to the partitioned universal time axis. At the νth clock tick, device 1 is a transmitter as denoted by TX (upward arrow) whereas device 2 is a receiver as denoted by RX (downward arrow).

Finally, the jth receiver device samples (3.5) at time instances kT_s during the reception period, where $T_s \ll T_0$ is the sampling period, $k \in \mathcal{K} = \{-K, \cdots, -1, 0, 1, \cdots, K\}$ is the discrete-time index, and $K = \lfloor \frac{T_0}{2T_s} \rfloor$. The resulting sampled signal at the νth clock tick is expressed as:

$$\begin{aligned} y_j[k;\nu] &= y_j(kT_s + t_j[\nu]) \\ &= \sum_{\eta \in \mathcal{V}_\nu} \sum_{i \in \mathcal{T}_\eta} \sum_{p \in \mathcal{P}} \rho_{ijp} x(kT_s + t_j[\nu] - t_i[\eta] - \tau_{ijp}) + w_j[k] \end{aligned} \tag{3.6}$$

where $w_j[k]$ is the discrete-time noise process.

3.2.3 DPLL Clock Update

At the νth clock tick, the jth receiver device cross-correlates (3.6) with the two distinct parts of the synchronization signal (available locally) to decouple the effect of CFO in TO estimation [64]:

$$R_{y_j x_\pm}[l, \nu] = \sum_{k \in \mathcal{K}} y_j[k; \nu] x_\pm[k - l]^* \tag{3.7}$$

where l is the integer lag and superscript * denotes complex conjugation. The first correlation is with $x_+[k] = \sum_{n=0}^{N-1} s[n] g(kT_s - nT_p)$, which is obtained from (3.3) by retaining pulses with index $0 \le n \le N - 1$ and sampling every T_s, while the second correlation is with $x_-[k] = \sum_{N}^{2N-1} s[n] g(kT_s - nT_p)$, which is obtained in a similar way but for $N \le n \le 2N - 1$. Hence, the subscripts $\pm$ in the signals $x_\pm[k]$ indicate which ZC root index (i.e., $+u$ or $-u$) is used to construct them.

The jth device then uses a weighted average across lags l to obtain two different preliminary TO estimates [59]:

$$q_j^\pm[\nu] = \frac{\sum_l l\, T_s |R_{y_j x_\pm}[l, \nu]|^2}{\sum_l |R_{y_j x_\pm}[l, \nu]|^2}. \tag{3.8}$$

Here, $q_j^+[\nu]$ provides an estimate of the average TO seen by the jth device, while $q_j^-[\nu]$ provides a similar estimate but with a positive offset of NT_p due to the definition of $x_-[k]$ in (3.7). Then, the jth device combines the estimates in (3.8) to obtain the desired weighted average TO estimate as:

$$\widehat{\Delta t}_j[\nu] = \frac{1}{2}\big(q_j^+[\nu] + q_j^-[\nu] - NT_p\big). \tag{3.9}$$

In the following, this quantity will be interpreted as the synchronization error experienced by the jth receiver at the νth clock tick. Hence, the jth device updates its clock tick for the $(\nu+1)$th time slot as dictated by DPLLs [58]:

$$t_j[\nu+1] = t_j[\nu] + \alpha_j T_0 + \epsilon \widehat{\Delta t}_j[\nu], \ j \in \mathcal{R}_\nu \tag{3.10}$$

where $\epsilon > 0$ is a scaling parameter. We note that by using the DPLLs clock update, the devices implicitly change their clock phases, which makes the clock phase time-variant, i.e., $\theta_j \equiv \theta_j[\nu]$. Therefore, to emphasize this point, we re-write the discrete logical clock in (3.1) as $t_j[\nu] = \alpha_j \nu T_0 + \theta_j[\nu], \ \forall j \in \mathcal{J}$. In contrast to (3.10), for the ith transmitter device, no correction is made and the clock tick is updated as:

$$t_i[\nu+1] = t_i[\nu] + \alpha_i T_0, \ i \in \mathcal{T}_\nu. \tag{3.11}$$

3.2.4 Problem Formulation

When devices join the network, unpredictable differences in clock phases, clock skews and propagation delays lead to synchronization errors. To further elaborate on this point, let us analyze the weighted average TO expression in (3.9). Based on the signal model in (3.6), this can be approximated as [58]:

$$\begin{aligned}\widehat{\Delta t}_j[\nu] \approx & \sum_{\eta\in\mathcal{V}_\nu} \sum_{(i,p)\in\mathcal{D}_j^{\nu,\eta}} \mu_{ijp}(t_i[\eta] + \tau_{ijp}) - t_j[\nu] \\ = & \underbrace{\sum_{\eta\in\mathcal{V}_\nu} \sum_{(i,p)\in\mathcal{D}_j^{\nu,\eta}} \mu_{ijp}\big(t_i[\eta] - t_j[\nu]\big)}_{\overline{\Delta t}_j[\nu]} + \underbrace{\sum_{\eta\in\mathcal{V}_\nu} \sum_{(i,p)\in\mathcal{D}_j^{\nu,\eta}} \mu_{ijp}\tau_{ijp}}_{\beta_j[\nu]}\end{aligned} \tag{3.12}$$

where $\mathcal{D}_j^{\nu,\eta} = \big\{(i,p) \in \mathcal{T}_\eta \times \mathcal{P} : \big|t_i[\eta] + \tau_{ijp} - t_j[\nu]\big| \leq \frac{T_0}{2}\big\}$ is the set of pairs formed by the index of transmitter devices and the path indices contributing to the received signal of the jth receiver device during the reception period centered at its νth clock tick. In (3.12), μ_{ijp} is the normalized channel gain of the pth path between the ith and jth devices given by $\mu_{ijp} = |\rho_{ijp}|\big(\sum_{\eta\in\mathcal{V}_\nu}\sum_{(i,p)\in\mathcal{D}_j^{\nu,\eta}} |\rho_{ijp}|\big)^{-1}$. Moreover, $\widehat{\Delta t}_j[\nu]$ can be written as a sum of two terms: the first term $\overline{\Delta t}_j[\nu]$, is the weighted average of clock tick differences between the contributing transmitters and the jth receiver, which includes the effects of the relative clock phases and clock skews. The second term $\beta_j[\nu]$ is a weighted average of the propagation delays seen from the jth device; we will refer to this term as *bias*.

Using (3.12), the DPLL clock update in (3.10) becomes:

$$t_j[\nu+1] \approx t_j[\nu] + \alpha_j T_0 + \epsilon\big(\overline{\Delta t}_j[\nu] + \beta_j[\nu]\big),\ j \in \mathcal{R}_\nu. \tag{3.13}$$

Even if the device clocks were perfectly aligned initially, i.e., $t_i[0] = t_j[0]\ \forall i,j \in \mathcal{J}$, and consequently $\overline{\Delta t}_j[0] = 0$, the clocks might start deviating from each other as ν increases due to differences in clock skews, i.e., $\alpha_i \neq \alpha_j$, so that $\overline{\Delta t}_j[\nu] \neq 0$ for $\nu > 0$ in general. Furthermore, the propagation delays, which are always positive by nature, introduce an additional error due to the bias $\beta_j[\nu]$. Hence, our first objective in this work is to reduce the effects of clock phases, clock skews and bias by means of distributed timing-advance synchronization. Ultimately, for ν sufficiently large, the maximum synchronization error for the overall network in the practical case with propagation delays should not exceed a pre-defined threshold λ_{sync}, that is:

$$\left(\max_{\substack{i,j\in\mathcal{J}\\ \eta\in\mathcal{V}_\nu}} \big|t_i[\eta] + \tau_{ij1} - t_j[\nu]\big|\right) \leq \lambda_{\text{sync}} \tag{3.14}$$

where τ_{ij1} is the delay of the first path and the maximum is over all $i,j \in \mathcal{J}$ as well as $\eta \in \mathcal{V}_\nu$.

In distributed networks, the devices are not aware of the synchronization status of others and they do not generally experience the same synchronization error. Therefore, the devices cannot stop the synchronization process simultaneously by just relying on their own error estimates, as given by (3.9). In fact, if some devices stop this process earlier than others, they might become asynchronous with respect to the remaining devices still running DPLLs; while if some devices stop later than others, the presence of ongoing timing pulses might trigger the synchronized devices to re-start the process. Hence, our second objective aims for distributed coordination among the devices to let them be aware of the overall network synchronization status and, ideally, terminate the synchronization process at the same time.

After synchronization, some of the devices may become idle to conserve energy instead of immediately initiating data communication, in which case the idle devices may become asynchronous with the rest of the network. To prevent this, they must periodically re-initiate the synchronization process. However, frequent re-initialization will lead to a disturbance among the synchronized devices and increased energy consumption in the overall network. Thus, our third and final objective is to maintain synchronization as long as possible without exchanging timing pulses to reduce the frequency of re-initialization.

3.3 Timing-Advance Synchronization Algorithm

In this section, we introduce a timing-advance synchronization algorithm that will later be used to develop a full-fledged synchronization protocol. First, we introduce a half-duplex method, whereby each device alternates between the transmitter and receiver modes in their exchange of timing pulses at each clock tick. Second, we modify the DPLLs clock update to achieve timing-advance synchronization in a distributed manner. Third, we present an online bias estimation technique, which will be incorporated into the modified DPLLs. Finally, the overall algorithm is given from the perspective of a single device.

3.3.1 Alternating Transceiver Mode

In distributed half-duplex synchronization, some devices broadcast timing pulses, while the others listen for the broadcasted signals during their own reception period to update their clock. However, if the devices randomly decide their transceiver mode at each clock tick as in [63], then the set $\mathcal{D}_j^{\nu,\eta}$ of transmitter devices and path indices seen by the jth receiver device, as defined in (3.12), will evolve unpredictably over time. In turn, this will result in arbitrary fluctuations in the weighted average TO estimates $\widehat{\Delta t}_j[\nu]$ [125].

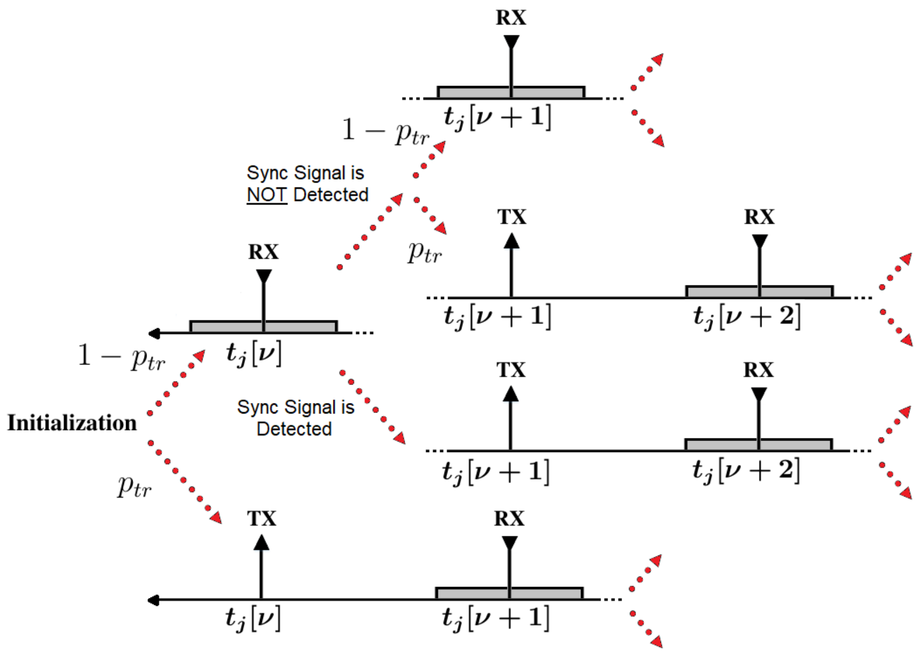
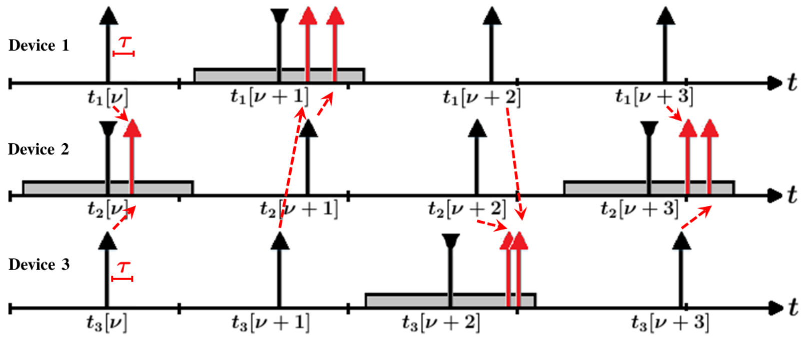


Fig. 3.3. Concept of the alternating transceiver mode.

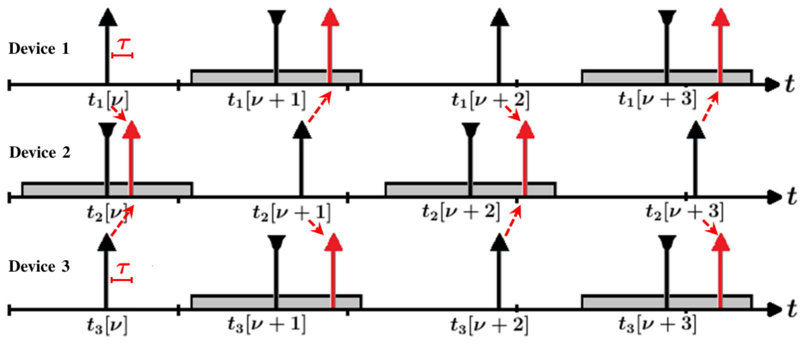
To eliminate this source of randomness in the set $\mathcal{D}_j^{\nu,\eta}$, we propose a method called *alternating transceiver mode*, which enables devices to determine their transceiver mode independently in a systematic manner. The underlying concept of the method is illustrated in Fig. 3.3. When a device first joins the network at the νth clock tick, it randomly initializes its transceiver mode, where the probability of being a transmitter, denoted as $p_{\text{tr}} \in (0, 1)$, is pre-determined and the same for all devices. If a device operates as a transmitter, it broadcasts its synchronization signal and then for the next clock tick, it changes its mode to become a receiver. If instead the device operates as a receiver, it listens for broadcast synchronization signals and attempts to determine whether or not such a signal is present during its reception period. This detection is performed by comparing the cross-correlation in (3.7) to a threshold value, as further explained in Section 3.4. In the case of signal detection (whose probability depends on several factors, e.g., the number of devices, p_{tr}, and the Signal-to-Noise Ratio (SNR)) the device alternates its mode at the next clock tick to operate as a transmitter; otherwise, it randomly re-determines its transceiver mode based on p_{tr} as above.

To illustrate the behavior of the proposed alternating transceiver mode, we compare it in Fig. 3.4 to the random transceiver mode proposed in [63], where its main goal is to randomly determine the transceiver mode of devices at *each* clock tick based on being a transmitter with probability p_{tr} or a receiver with probability $1 - p_{tr}$. As seen in Fig. 3.4 (a), in the case of selecting the transceiver mode randomly, the average TO seen from the receivers may fluctuate greatly from one clock tick to the next, which, in turn, results in increased synchronization errors. In sharp contrast, with the proposed method the average TO estimates and clock updates evolve over time in an orderly manner (cf. Fig. 3.4 (b)) which facilitates fast convergence with a deterministic bias.

Based on the diagram in Fig. 3.3 and assuming that the probability of signal detection for a receiver is high, the devices will cluster themselves into TX and RX groups and alternate between them at each clock tick, which can be also seen in Fig. 3.4 (b). We elaborate this behavior in our simulations under representative conditions of operation for LTE [128], as further discussed in Section 3.5. Hence, the synchronization signals are exchanged between the same groups of transmitter and receiver devices, which has two important implications. First, the clocks of each device are corrected at every two clock ticks by using DPLLs as given in (3.10). Consequently, clock skew can only alter the device clocks for no more than one clock tick before being compensated, which helps speed up synchronization. Second, in the case of time-invariant channels, the bias term in (3.12) becomes constant, i.e., $\beta_j[\nu] = \beta_j[\nu+2]$, since the jth receiver is always affected by the same combination of transmitters and propagation paths, i.e., $\bigcup_\eta \mathcal{D}_j^{\nu,\eta}$ remains constant as ν is incremented to $\nu + 2$.



(a) Random transceiver mode.



(b) Alternating transceiver mode.

Fig. 3.4. Comparison between the random transceiver mode and the proposed alternating transceiver mode for pulse-coupled clock synchronization in the presence of propagation delays with no clock skew. A network of 3 devices is assumed, located at equal distances from each other. The propagation delay from one device to any other device is τ. Hence, the broadcasted beacons are detected with a delay as depicted with dashed-arrows within the receiver's reception period (depicted in gray). Note that at the ν^{th} clock tick Device 2 is a receiver while the other two devices are transmitters.

3.3.2 Modified DPLLs

Timing-advance synchronization can reduce the effect of propagation delays. Under the alternating transceiver mode, the receiver devices expect signals from the same group of transmitters at every two clock ticks. Hence, we let the receiver devices take proactive actions by modifying the DPLLs clock update in (3.10) as follows:

$$t_j[\nu+1] = t_j[\nu] + \alpha_j T_0 + \epsilon \widehat{\Delta t}_j[\nu] - 2\widehat{\beta}_j[\nu],\ \forall j \in \mathcal{R}_\nu \tag{3.15}$$

where $\widehat{\beta}_j[\nu]$ is an estimate of the bias term in (3.12). Meanwhile the clock update of transmitter devices remains unchanged, as already given by (3.11).

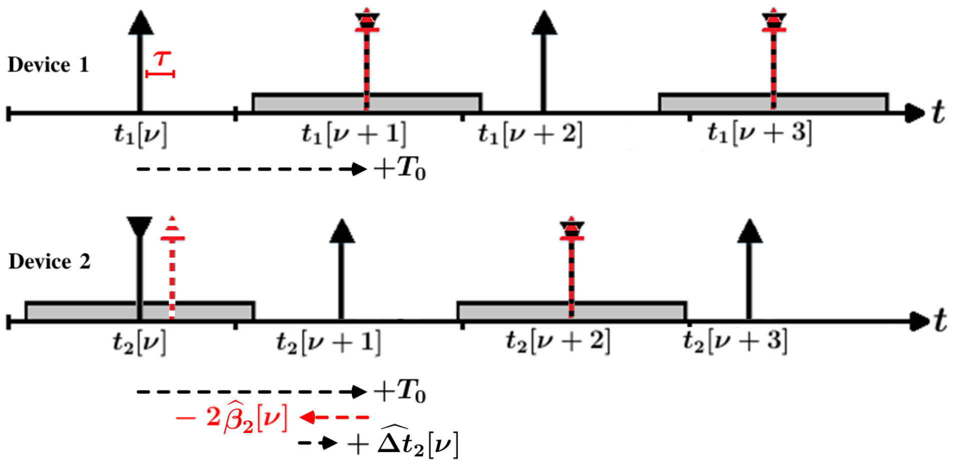


Fig. 3.5. Timing-advance synchronization in a distributed manner with propagation delay τ (upward dashed arrow represents the arrival time of a synchronization signal).

To motivate the introduction of the term $-2\widehat{\beta}_j[\nu]$ in (3.15), we consider a simple case wherein two devices, labeled as D1 and D2, exchange synchronization signals under the alternating transceiver mode over a single path channel with delay $\tau > 0$. Assume that at the νth clock tick, D1 and D2 operate as transmitter and receiver, respectively, and that their clocks are perfectly aligned without clock skew, i.e., $t_1[\nu] = t_2[\nu]$ and $\alpha_1 = \alpha_2 = 1$ as illustrated in Fig. 3.5. After broadcasting its signal, D1 updates its clock based on (3.11) as $t_1[\nu+1] = t_1[\nu] + T_0$, and then switches to the receiver mode. Meanwhile, due to the propagation delay, the corresponding synchronization error at D2 is $\widehat{\Delta t}_2[\nu] = \tau$. Assuming that D2 has a perfect bias estimate, i.e., $\widehat{\beta}_2[\nu] = \tau$, it updates its clock based on the modified DPLLs (3.15) as $t_2[\nu+1] = t_2[\nu] + T_0 - \tau$, where ϵ is set to 1 for simplicity. At the $(\nu+1)$th clock tick, D2 which now operates as a transmitter, broadcasts its synchronization signal. Due to the additional term $-2\widehat{\beta}_j[\nu] = -2\tau$, the synchronization error at D1, which now operates as a receiver, will be $\widehat{\Delta t}_1[\nu+1] = 0$.

More generally, by employing the modified DPLLs along with alternating transceiver mode, clocks of the receivers will be advanced in time with respect to the clocks of the transmitters at every tick. Therefore, the devices can achieve timing-advance synchronization in a distributed manner.

3.3.3 Estimation of the Bias

Application of the modified DPLLs in (3.15) requires an estimate of the bias term $\beta_j[\nu]$. For estimating this term, the jth device can only rely on its TO estimate $\widehat{\Delta t}_j[\nu]$ from (3.9), which includes contributions from both $\overline{\Delta t}_j[\nu]$ and $\beta_j[\nu]$, as seen from (3.12). Furthermore, as each device updates its clock based on $\widehat{\Delta t}_j[\nu]$ by using the recursive DPLLs in (3.15), future TO estimations will be affected by previous clock corrections. Hence, conventional time averaging techniques applied to $\widehat{\Delta t}_j[\nu]$ might fail to yield an unbiased estimate. Besides, the averaging time required by conventional techniques to achieve the desired accuracy may negatively impact the overall synchronization time. Therefore, estimation of $\beta_j[\nu]$ becomes challenging.

Alternatively, we can capitalize on the fact that, by using the alternating transceiver mode, the bias seen from a receiver device over time-invariant channels becomes constant. By exploiting this special property of $\widehat{\beta}_j[\nu]$, the jth receiver device may try to isolate it from $\widehat{\Delta t}_j[\nu]$ while updating its clock based on (3.15). Since by using DPLLs, $\overline{\Delta t}_j[\nu]$ tends to 0 as ν increases in the absence of propagation delays [58], estimation of the bias can be achieved by seeking to iteratively reduce the absolute synchronization error, i.e., $|\widehat{\Delta t}_j[\nu+2]| \leq |\widehat{\Delta t}_j[\nu]|$. To this end, we therefore propose the following online bias estimation technique:

$$\widehat{\beta}_j[\nu] = \widehat{\beta}_j[\nu-2] + \gamma_j \mathrm{sgn}(\widehat{\Delta t}_j[\nu]) \tag{3.16}$$

where $\mathrm{sgn}(\cdot)$ is the sign function, i.e., the jth receiver device applies corrections to its bias estimate with a small step size $\gamma_j \in \mathbb{R}_+$ based on the sign of its current TO estimate. Specifically, if $\widehat{\Delta t}_j[\nu] > 0$ (< 0), then the weighted average of the signal contributions from transmitters to the jth device is advanced (is lagging) in time with respect to its own clock tick. Hence, the jth device increases (decreases) its previous estimate $\widehat{\beta}_j[\nu-2]$ by γ_j in order to reduce its future TO estimate $|\widehat{\Delta t}_j[\nu+2]|$. When a device joins the network at tick ν_0, it may initialize its bias estimate $\widehat{\beta}_j[\nu_0] = \widehat{\beta}^{\mathrm{init}}$ based on the expected physical delay in the network. For instance, the initial value can be set to $\widehat{\beta}^{\mathrm{init}} = \frac{d}{c}$, where d is the average distance between the devices and c is the speed of light.

In Appendix A, we analytically show that the application of (3.16) in a multi-device network leads to a reduction of the synchronization error at each device as ν increases. In practice, we have found that this technique can be improved further by using a dynamic step size instead of a fixed one, that is, $\gamma_j[\nu] = a\gamma_j[\nu-2] + b$, where $a \in (0,1)$ and $b \in \mathbb{R}_+$ is a constant increment.

3.3.4 Proposed Algorithm

Each device independently implements the above procedures for distributed timing-advance synchronization after joining the network at the ν_0th clock tick. In Algorithm 3.1, these procedures are summarized and presented from the perspective of the jth device, assuming without loss in generality that it joins the network at the $\nu_0 = 0$th clock tick.

Algorithm 3.1 Timing-Advance Synchronization

```
1: Initialize $M^j_{\nu_0}$ (based on $p_{\text{tr}}$) and $\widehat{\beta}_j[\nu_0] = \widehat{\beta}^{\text{init}}$
2: for $\nu = 0, 1, 2, ...$ do
3:     if $M^j_\nu = \text{TX}$ then
4:         Broadcast the synchronization signal (3.3)
5:         Advance the clock as in (3.11)
6:         Become receiver: $M^j_{\nu+1} = \text{RX}$
7:     else
8:         if Synchronization signal is detected then
9:             Estimate the average TO (3.9)
10:            Update the clock based on modified DPLLs (3.15)
11:            Update the bias estimate (3.16)
12:            Become transmitter: $M^j_{\nu+1} = \text{TX}$
13:        else
14:            $M^j_{\nu+1} = \begin{cases} \text{TX}, & \text{with } p_{\text{tr}} \\ \text{RX}, & \text{with } 1 - p_{\text{tr}} \end{cases}$
15:            Advance the clock as in (3.11)
16:        end
17:    end
18: end
```

In Appendix B we study the rate of decrease of TO when Algorithm 3.1 is applied to a simplified scenario consisting of two devices. Furthermore, in Appendix C, we derive the expected value of TO when the devices run Algorithm 3.1 in the same simplified scenario. Therefore, based on the effect of various parameters such as γ_j, p_{tr} and ϵ used in the algorithm, we can choose the values that yield the best performance in terms of error reduction.

3.4 Proposed Synchronization Protocol

Based on the timing-advance synchronization algorithm, we first propose a synchronization protocol and present it over a state-transition diagram and then we give its complexity analysis. The underlying idea of the protocol is to create coordination among the participating devices to ensure a simultaneous termination of the synchronization process and let them initiate data communication. Therefore, to coordinate the devices in a distributed manner, we propose to use two different synchronization signals; the first signal is utilized for error reduction, whereas the presence of the second signal is a declaration of the synchronization status of a device to the network.

We construct these two signals based on (3.3)-(3.4), which can be distinguished by their ZC index u. Specifically, all the transmitter devices broadcast the first signal by using the ZC index $u = u_1$ to decrease their synchronization error below a threshold. Once they achieve this, the devices start broadcasting the second signal by switching to the ZC index $u = u_2$. If a receiver device detects only the second signal but not the first one, then it knows that all the contributing devices are synchronized. In other words, the use of the ZC root index u_2 by a device serves as a declaration to the rest of the network that it has deemed itself synchronized.

Since the devices use two synchronization signals, the cross-correlated received signal at a device can be one of four kinds:

- It contains no synchronization signals (this may occur when there are no transmitter devices).
- It only contains synchronization signals using root index u_1 (this occurs when all transmitter devices try to reduce their own synchronization errors);

- It only contains synchronization signals using root index u_2 (this occurs when all the transmitter devices deem themselves to be synchronized).
- It contains a mix of signals using u_1 and u_2 (this occurs when some devices are still reducing their errors while others are deem themselves to be synchronized).

Therefore, a receiver device should be able to reliably detect the presence or absence of the ZC root index u_1 or u_2 within the received signal by performing the cross-correlation in (3.7), which we re-define as $R_{y_j x_\pm^u}[l,\nu] = \sum_{k\in\mathcal{K}} y_j[k;\nu] x_\pm^u[k-l]^*$ to emphasize u.

Considering the properties of ZC sequences, auto-correlation of two synchronization signals that have the same root index yields a peak value N at lag l, i.e., $|R_{x_\pm^{u_1} x_\pm^{u_1}}[l,\nu]| = N$ [129]; however, cross-correlation of such two signals with different root indices yields $|R_{x_\pm^{u_1} x_\pm^{u_2}}[l,\nu]| = \sqrt{N}$ [130]. Hence, after cross-correlating the received signal in (3.7) with both $x_\pm^{u_1}[k]$ and $x_\pm^{u_2}[k]$ at the the νth clock tick, the jth receiver device calculates the following decision statistic for each $u \in \{u_1, u_2\}$:

$$\psi(\nu,u) = \max_l(|R^u_{y_j x_\pm}[l,\nu]|) \tag{3.17}$$

and compares it to a detection threshold λ_{det}. Note that in LTE, ZC sequence length N is 839 [130], and considering the typical noise levels, we set this detection threshold as $\lambda_{\text{det}} = \frac{N}{2}$ in our work. Accordingly, the receiver makes one of the following four decisions:

$$\begin{aligned}
\mathrm{D}_\nu^{00} &: \ \psi(\nu,u_1) < \lambda_{\text{det}} \ \wedge \ \psi(\nu,u_2) < \lambda_{\text{det}} \\
\mathrm{D}_\nu^{10} &: \ \psi(\nu,u_1) \geq \lambda_{\text{det}} \ \wedge \ \psi(\nu,u_2) < \lambda_{\text{det}} \\
\mathrm{D}_\nu^{01} &: \ \psi(\nu,u_1) < \lambda_{\text{det}} \ \wedge \ \psi(\nu,u_2) \geq \lambda_{\text{det}} \\
\mathrm{D}_\nu^{11} &: \ \psi(\nu,u_1) \geq \lambda_{\text{det}} \ \wedge \ \psi(\nu,u_2) \geq \lambda_{\text{det}}
\end{aligned} \tag{3.18}$$

where $\wedge$ is the logical conjunction and the superscripts of the decisions indicate the presence, i.e., 1, or the absence, i.e., 0, of the ZC root indices u_1 and u_2, respectively. Decision D_ν^{00} corresponds to the case where no synchronization signal is detected. In response, the device re-establishes its transceiver mode for the next tick based on p_{tr} and advances its clock by one clock period. For any of the other three decisions, the receiver device forms its final TO estimate as follows:

$$\widehat{\Delta t}_j[\nu] = \begin{cases} \widehat{\Delta t}_j^{u_1}[\nu] & , \ \mathrm{D}_\nu^{10} \\ \widehat{\Delta t}_j^{u_2}[\nu] & , \ \mathrm{D}_\nu^{01} \\ \frac{1}{2}\big(\widehat{\Delta t}_j^{u_1}[\nu] + \widehat{\Delta t}_j^{u_2}[\nu]\big) & , \ \mathrm{D}_\nu^{11} \end{cases} \tag{3.19}$$

In the above, the superscript to the weighted average TO estimate $\widehat{\Delta t}_j^{u_1}[\nu]$ or $\widehat{\Delta t}_j^{u_2}[\nu]$ indicates from which ZC root index it is obtained. In the case of D_ν^{11}, the jth device forms its final TO estimate by using the arithmetic mean of its individual TO estimates $\widehat{\Delta t}_j^{u_1}[\nu]$ and $\widehat{\Delta t}_j^{u_2}[\nu]$, since it is consistent with the definition of (3.9) as a weighted average.

In what follows we describe the proposed protocol over a state-transition diagram as given in Fig. 3.6 from the point of view of the jth device.

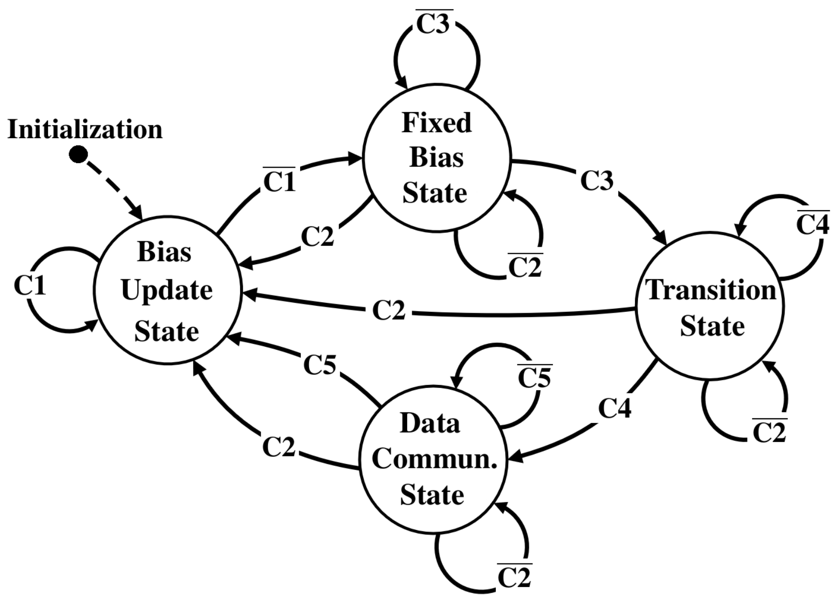


Fig. 3.6. State-transition diagram of the proposed protocol. Transitions happen at each clock tick under different conditions; the conditions and their complements are denoted by **C**i and $\overline{\textbf{C}}$i, respectively, where i $\in \{1, 2, ..., 5\}$.

Initialization: When the jth device joins the network, it stores its initial transceiver mode $M_{\nu_0}^j$. Furthermore, the jth device initializes Γ_j, ξ_j and ζ_j, which will be used as error level, stopping and clock skew control counters, respectively, as:

$$\Gamma_j = 0, \ \xi_j = 0 \text{ and } \zeta_j = 0 \tag{3.20}$$

The purpose of using the counters throughout the protocol is to provide a controlled evolution between the states by comparing them to pre-defined thresholds.

3.4.1 Bias Update State

In this state, the jth device tries to iteratively estimate its bias to reduce its synchronization error while updating its clock. Therefore, the device runs Algorithm 3.1 with $u = u_1$ as long as the following condition is satisfied:

C 1: $|\widehat{\Delta t}_j[\nu]| \leq |\Delta t_j|_{\min} \vee |\Delta t_j|_{\min} > \lambda_{\text{sync}}$
where $\vee$ is the logical disjunction, $|\Delta t_j|_{\min}$ is the smallest value of the weighted average TO encountered up to the νth iteration and λ_{sync} is the pre-defined synchronization threshold. Note that $|\Delta t_j|_{\min}$ can be less than λ_{sync}. Hence, if the synchronization error stays above the desired threshold or it keeps decreasing in absolute value, then the device does not change its state.

3.4.2 Fixed Bias State

In this state, the absolute value of the synchronization error of the device stops decreasing. Therefore, the device stops updating its bias estimate and fixes it to the one that yields the smallest error $|\Delta t_j|_{\min}$ as:

$$\widehat{\beta}_j[\nu + 2] = \widehat{\beta}_j[\nu] \tag{3.21}$$

From then on, the device only uses this bias estimate in Algorithm 3.1, i.e., (3.16) is replaced by (3.21). Meanwhile, the device increases its error level counter Γ_j but only when it operates as a receiver device:

$$\Gamma_j = \Gamma_j + 1\ , \text{ if } M_\nu^j = \text{RX} \tag{3.22}$$

In order for a device to leave this state, there are two conditions. The first condition captures the effect of a perturbation in the system, e.g., a new device joining the network, which can be detected by checking for a sudden change in the weighted average TO estimate as follows:

C 2: $\left||\widehat{\Delta t}_j[\nu]| - |\Delta t_j|_{\min}\right| > \lambda_{\text{sync}}$
In this way, the proposed protocol can operate with dynamic network size.

Whereas the second condition indicates whether or not the device deems itself synchronized. Specifically, the decrease in the weighted average TO of the jth device, i.e., $|\widehat{\Delta t}_j[\nu]|$, might be temporarily due to noise in the TO estimate or some devices leaving the network, hence, the device should not consider itself to be synchronized immediately. Instead, the device observes the change in $|\widehat{\Delta t}_j[\nu]|$ by using condition $\overline{\mathbf{C\,2}}$ and assumes synchronization only after this condition is satisfied for λ_{cons} consecutive clock ticks, that is:

C 3: $\Gamma_j \geq \lambda_{\text{cons}}$

3.4.3 Transition State

In this state the device declares its synchronization status to the network. Although the device still continues to run Algorithm 3.1, it changes its ZC index as follows:

$$u \leftarrow u_2 \text{ if } M_\nu^j = \text{TX} \tag{3.23}$$

Therefore, if the device in this state operates as a receiver at the νth clock tick, i.e., $M_\nu^j = \text{RX}$, and detects the presence of the second synchronization signal but not the first one, it assumes that other devices are also synchronized, hence, it increases its stopping counter ξ_j as:

$$\xi_j = \begin{cases} \xi_j + 1 \,, & \mathrm{D}_\nu^{01} \;\vee\; (\mathrm{D}_\nu^{00} \;\wedge\; \xi_j > 0) \\ 0 \,, & \mathrm{D}_\nu^{10} \;\vee\; \mathrm{D}_\nu^{11} \end{cases} \tag{3.24}$$

Note that when $(\mathrm{D}_\nu^{00} \wedge \xi_j > 0)$ is satisfied, the receiver device does not detect any signals, i.e., D_ν^{00}, yet it has a non-zero stopping counter. Since the device is in the transition state, this can be interpreted in two ways: either the previously detected devices left the network, or they switched to *Data Communication State*, where they essentially stop their synchronization process. In both cases, the device continues to increase its counter.

In contrast, a device operating as a transmitter at the νth clock tick, i.e., $M_\nu^j = \text{TX}$, cannot detect any signals, therefore, cannot make any decisions about the presence of the synchronization signals. Hence, we propose to incorporate the initial transceiver mode $M_{\nu_0}^j$ to the decision process. Specifically, when a device operates as a transmitter at the νth clock tick and its initial mode was also transmitter, i.e., $M_\nu^j = M_{\nu_0}^j = \text{TX}$, then it increases its stopping counter. However, a transmitter device still needs to obtain decision statistics at the νth clock tick (3.17), therefore, we propose to rely on the decision statistic from the $(\nu-1)$th clock tick, where the device was a receiver.

If $M_{\nu}^{j} = \mathrm{TX}$:

$$\xi_j = \begin{cases} \xi_j + 1 \ , \ (M_{\nu_0}^{j} = \mathrm{TX}) \wedge \mathrm{D}_{\nu-1}^{01} \\ \quad 0 \quad , \ (M_{\nu_0}^{j} = \mathrm{TX}) \wedge (\mathrm{D}_{\nu-1}^{10} \vee \mathrm{D}_{\nu-1}^{11}) \end{cases} \tag{3.25}$$

Similar to the previous state, perturbations must be detected, i.e., condition **C 2**. However, since not every device necessarily has the same synchronization error, there might be some devices that are still trying to reduce their error by broadcasting the first signal with ZC index $u = u_1$ while not triggering condition **C 2**. In this case, when the jth device in this state detects the first synchronization signal, it resets its stopping counter, i.e., sets $\xi_j = 0$. Therefore, it waits for the other devices to first synchronize themselves, then switch to the transition state, where they finally broadcast the second synchronization signal.

Note that when proceeding to the communication state, the devices should not lose synchronization with respect to each other. Specifically, this is important to avoid re-initiating the synchronization process for timing-advance communication, which will be explained later. Therefore, the next condition not only allows devices to terminate the synchronization process, ideally at the same time, but also enables them to initiate timing-advance data communication in a distributed manner.

C 4: C4.1 $\vee$ **C4.2**

C4.1: $(M_{\nu}^{j} = M_{\nu_0}^{j} = \mathrm{RX}) \wedge \big((\xi_j > \lambda_{\text{stop}}) \vee (\mathrm{D}_{\nu}^{00} \ \wedge \ \xi_j > 0)\big)$

C4.2: $(M_{\nu}^{j} = M_{\nu_0}^{j} = \mathrm{TX}) \wedge (\xi_j > \lambda_{\text{stop}})$

The device switches to the data communication state when its stopping counter exceeds the pre-defined stopping threshold λ_{stop} if and only if its current transceiver mode at the νth clock tick, i.e., M_{ν}^{j}, is the same as its initial transceiver mode, i.e., $M_{\nu_0}^{j}$. Specifically, if the device is a receiver (transmitter) and the first (second) sub-condition **C4.1** (**C4.2**) is satisfied, the device stops its synchronization process as a receiver (transmitter). Hence, the device is aware of whether or not its clock is advanced or regressed with respect to others when the synchronization process stops.

Note that in the case of no signal detection with a non-zero stopping counter, the device can safely assume that the contributing transmitter devices have already switched to the data communication state or left the network, hence, it can proceed to the next state.

3.4.4 Data Communication State

In this state, the device can conserve energy by only operating as a receiver until it initiates data communication. Therefore, the device sets its transceiver mode to receiver:

$$M_{\nu}^{j} \leftarrow \mathrm{RX} \tag{3.26}$$

and updates its clock similarly to (3.11). However, if the device anticipates data communication, then it relies on how condition **C 4** was satisfied, i.e., whether or not its clock is advanced or regressed at the termination of synchronization process. Specifically, $M_{\nu_0}^{j} = \mathrm{RX}$ means that the device is in the RX group which is suitable for data reception, while $M_{\nu_0}^{j} = \mathrm{TX}$ that the device in the TX group which is suitable for data transmission. On the contrary, if the device is in the group that does not match the anticipated communication type, i.e., $M_{\nu_0}^{j} = \mathrm{RX}$ but the device has data to transmit or $M_{\nu_0}^{j} = \mathrm{TX}$ but it expects to receive data, then any two devices in this case would need to re-initiate their own synchronization process for proper timing-advance, which would cause a network-wide perturbation. In order to avoid this, the proposed protocol allows such devices to re-arrange their clocks for proper timing-advance without actively transmitting and receiving synchronization signals as follows:

$$t_j[\nu+1] = \begin{cases} t_j[\nu] + \alpha_j T_0 - \widehat{\beta}_j[\nu], & M_{\nu_0}^{j} = \mathrm{RX} \,\wedge\, B_j = 1 \\ t_j[\nu] + \alpha_j T_0 + \widehat{\beta}_j[\nu], & M_{\nu_0}^{j} = \mathrm{TX} \,\wedge\, B_j = 0 \end{cases} \tag{3.27}$$

where $B_j \in \{0, 1\}$ indicates that the jth device anticipates data transmission or reception, when it is one or zero, respectively. By applying (3.27), the jth device uses its own bias estimate to adjust its clock to approach the vicinity of the clocks of the opposite group in time as shown in Fig. 3.7. Note that if the device applies (3.27), then it can simply revert this procedure to return its original TX or RX group after stopping data communication.

Overall, (3.27) is useful in terms of extending the achieved synchronization by allowing devices to use timing-advance communication without unnecessarily exchanging timing pulses. Therefore, for the proposed protocol, clock-skew is the only factor affecting re-initialization of the synchronization process. Indeed, in a practical system, the presence of clock skew imposes an upper bound on how long devices can stay synchronized. Since in the communication state, the jth device does not apply the DPLL clock update in (3.15), its clock might diverge from the clocks of other devices after some period of time due to clock skew. To

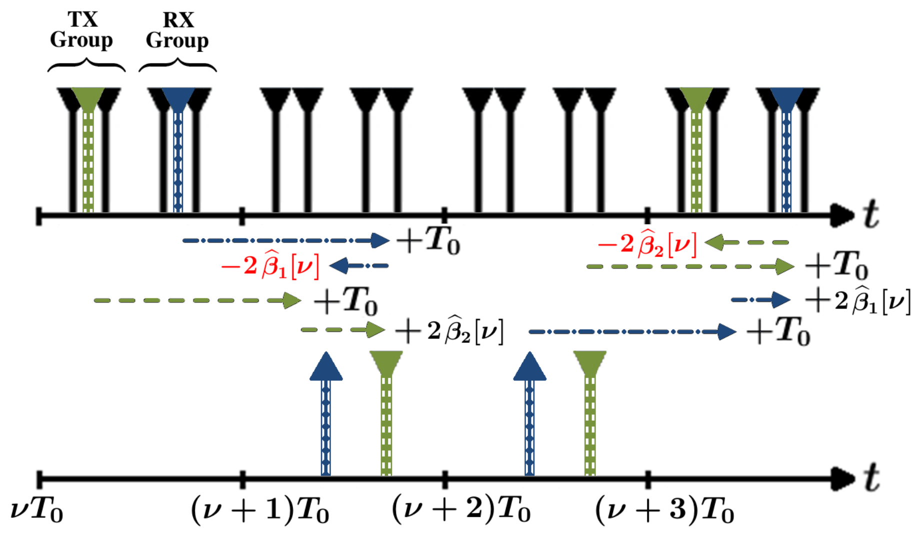

TX Group
RX Group
t
$+T_0$
$-2\widehat{\beta}_1[\nu]$
$-2\widehat{\beta}_2[\nu]$
$+T_0$
$+T_0$
$+2\widehat{\beta}_1[\nu]$
$+2\widehat{\beta}_2[\nu]$
$+T_0$
t
νT_0
$(\nu+1)T_0$
$(\nu+2)T_0$
$(\nu+3)T_0$

Fig. 3.7. After synchronization, the devices are aware the group they are in, i.e. TX or RX group. Hence, they can re-arrange their clocks by only using their own bias estimate for timing-advance communication.

prevent this, the device keeps track of how long it stays in this state by increasing its clock skew control counter at each clock tick as $\zeta_j \leftarrow \zeta_j + 1$, which is then compared to clock skew control threshold λ_{skew} as follows:

C 5: $\zeta_j \geq \lambda_{\text{skew}}$

If ζ_j exceeds the allowed limit, then the device re-initiates its synchronization process by resetting its variables as follows:

$$\Gamma_j = 0,\ \ \xi_j = 0 \text{ and } \zeta_j = 0 \tag{3.28}$$

$$|\Delta t_j|_{\min} = \widehat{\Delta t}_j[\nu] \tag{3.29}$$

$$\widehat{\beta}_j[\nu] = \widehat{\beta}^{\text{init}} \tag{3.30}$$

$$u \leftarrow u_1 \tag{3.31}$$

$$M_\nu^j \leftarrow M_{\nu_0}^j \tag{3.32}$$

We note that the condition **C5** is different than **Initialization**, where the devices are setting all their variables to 0 and determining $M_{\nu_0}^j$ randomly. However, in **C5**, the jth device sets its current transceiver mode to its initial transceiver mode $M_{\nu_0}^j$. Hence, if the devices switch to the bias update state due to the condition **C5**, then re-synchronization should take much less time since the effect of clock skew cannot cause drastic deviations on the device clocks.

3.4.5 Complexity Analysis

To derive the complexity analysis of the proposed protocol, we first analyze the required operations in Algorithm 3.1. When a device operates as a transmitter at the νth clock tick, i.e., $M_\nu^j = \text{TX}$, constructing the synchronization signal as in (3.3) requires $\mathcal{O}(2N)$ operations. Then, the clock update in (3.11) only takes one multiplication and one addition, hence, the complexity is relatively small. If a device operates as a receiver at the νth clock tick, i.e., $M_\nu^j = \text{RX}$, it first estimates the average TO in (3.9), which requires computing a cross-correlation as in (3.7), and then forming two different preliminary TO estimates as given by (3.8). Hence, the total operations needed for (3.7) have $\mathcal{O}(2K)$ complexity related to complex multiplications, whereas each preliminary TO estimate, i.e., $q_j^{\pm}[\nu]$, in (3.8) takes $\mathcal{O}(8K^2)$ operations. Therefore, the overall computation complexity of Algorithm 3.1 at the νth clock tick is $\mathcal{O}(16K^2 + 2N)$. Now considering the overall computation complexity of the proposed protocol, forming the final average TO estimate in (3.19) doubles the amount of operations due to utilization of two distinct ZC sequences, hence, the computational complexity at the νth clock tick becomes $\mathcal{O}(32K^2 + 2N)$. However, the rest of the computation complexity of the proposed protocol is relatively small since it only requires memory registers and comparisons between the assigned values.

3.5 Simulation Results

In this section, the proposed synchronization protocol is evaluated by means of computer simulations based on METIS 5G channel models [131] and under different conditions of operation, i.e., clock skew, number of devices, and network topologies, including full mesh and partial mesh.

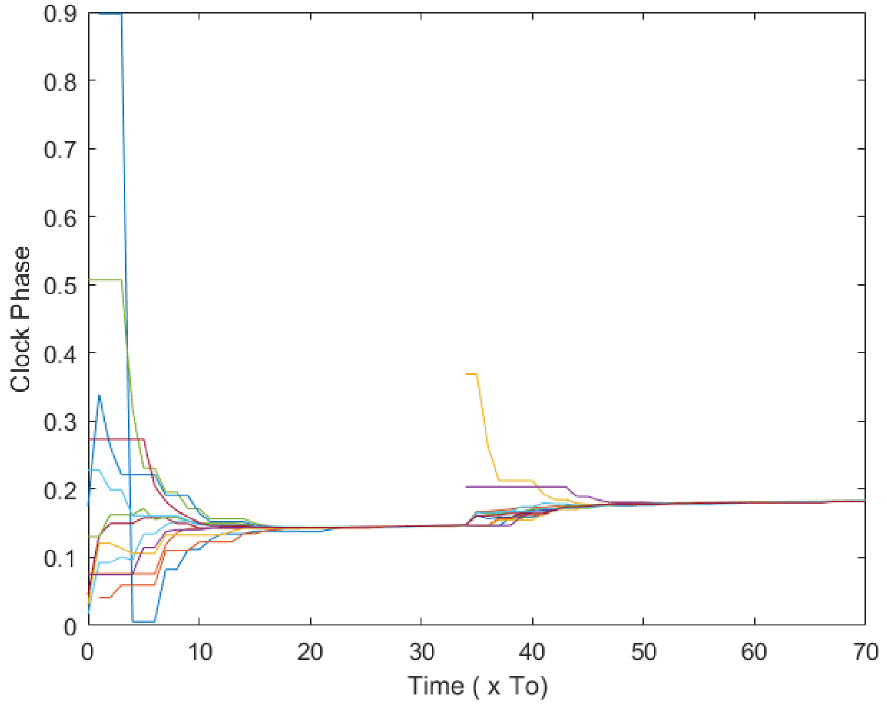
The complete synchronization protocol for a dense stationary D2D network is implemented in Matlab. Unless otherwise is specified, we consider a network of 14 devices, where 12 of them initialize at $\nu = 0$ with 2 more devices joining the network at $\nu = 33$. We use a channel model according to the Manhattan grid scenario in [131] with Rician fading for the line of sight path (if present) and Rayleigh fading for the other paths. The SNR at the input of the correlator is fixed at 15dB. The rest of the system parameters are chosen accordingly from [12] and given in Table 3.1.

Table 3.1. System parameters

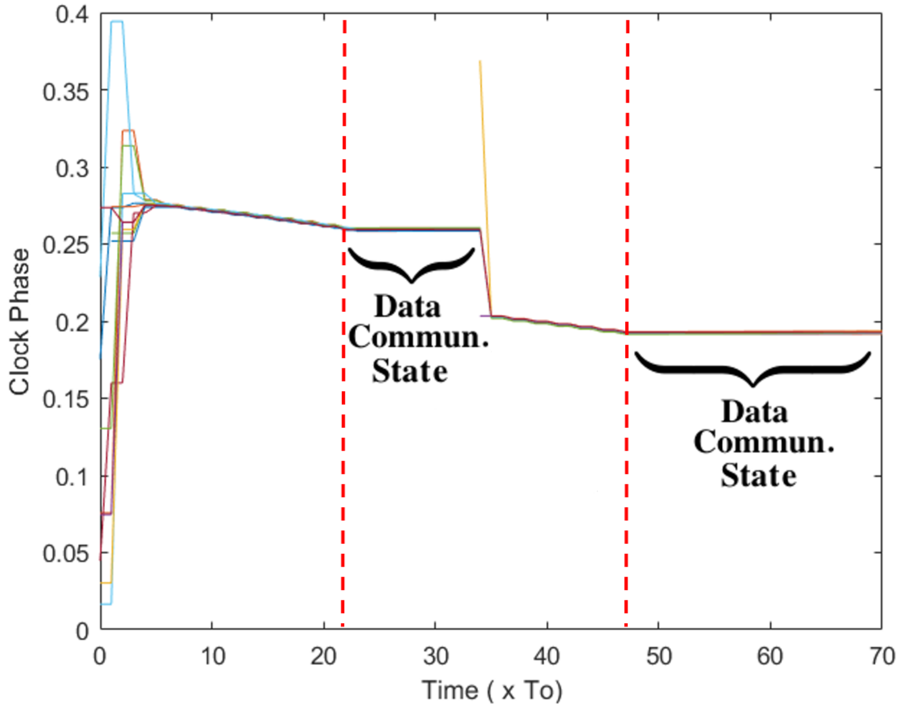
Parameter Description	**Symbol**	**Value**
Number of Devices	J	$2, 5, 8, 14$
Scaling Term of DPLL	ϵ	1
Zadoff-Chu Index	u_1, u_2	$7, 13$
Zadoff-Chu Sequence Length	N	839
Clock Period	T_0	1 ms
Pulse Spacing	T_p	0.1 μs
Sampling Period	T_s	3 ns
Transmit and Reception Powers	$P_{\text{TX}}, P_{\text{RX}}$	23 dBm, 8 dBm
Maximum Network Distance	d	500 m
Operating Frequency	f	2 GHz
Bias Estimate Initialization	$\widehat{\beta}^{\text{init}}$	.86 μs
Step Size Initialization	γ^{init}	33 ns
Step Size Slope	a	.98
Step Size Increment	b	3 ns
Probability of Being a Transmitter	p_{tr}	$\{0.1, 0.5, 0.9\}$
Signal Detection Threshold	λ_{det}	$\frac{N}{2}$
Synchronization Error Threshold	λ_{sync}	1.5 μs
Consecutive Clock Tick Threshold	λ_{cons}	2
Clock Skew Control Threshold	λ_{skew}	10
Stopping Threshold	λ_{stop}	2
Number of Resolvable Paths	P	4
Rayleigh Fading Scaling Parameter	-	1
Rician Fading Noncentrality Parameter	-	1
Rician Fading Scaling Parameter	-	1

3.5.1 Clock Phase Convergence

First, we study how fast our protocol reduces the synchronization error compared to [63] under the exact same network conditions. In Fig. 3.8 (a) we show a single realization of the clock phase evolution for the algorithm in [63] where the devices randomly choose their transceiver mode based on $p_{\text{tr}} = 0.5$. Similarly in Fig. 3.8 (b), the devices employ the proposed protocol with $p_{\text{tr}} = 0.5$. We note that the algorithm in [63] transmits synchronization signals for the duration of the experiment, while the proposed protocol only transmits until *Data Communication State* as illustrated in Fig. 3.8 (b).



(a) Random transceiver mode (RTM) [63] ($p_{\rm tr} = 0.5$).



(b) Proposed synchronization protocol ($p_{\rm tr} = 0.5$).

Fig. 3.8. A single realization of clock phase evolution with a perturbation at the 33rd clock tick.

As we can see, during these states, the proposed protocol not only reduces the difference in relative clock phases faster but also achieves a smaller deviation. Furthermore, our protocol quickly reacts to the perturbation that occurs at $\nu = 33$ when new devices join the network, and compensates for it better than [63].

Importantly, by using the proposed protocol, the devices are aware of the global synchronization status which allows them to terminate the synchronization process simultaneously at $\nu = 22$ and proceed to data communication state as shown in Fig. 3.8 (b). However, when using the algorithm in [63], the devices are not aware of the synchronization status of others, hence, they cannot simultaneously terminate this process and proceed to data communication as intended.

3.5.2 Synchronization Performance before Data Communication

To quantify the synchronization performance, we introduce three synchronization performance metrics: maximum, minimum and average synchronization errors at a given clock tick ν, which take propagation delays into account. These metrics are defined as follows:

$$\Delta t_{\text{sync}}^{max}[\nu] = \max_{(\eta,i,j)\in\mathcal{S}_\nu} \left|t_i[\eta] + \tau_{ij1} - t_j[\nu]\right| \tag{3.33}$$

$$\Delta t_{\text{sync}}^{min}[\nu] = \min_{(\eta,i,j)\in\mathcal{S}_\nu} \left|t_i[\eta] + \tau_{ij1} - t_j[\nu]\right| \tag{3.34}$$

$$\Delta t_{\text{sync}}^{avg}[\nu] = \max_{j\in\mathcal{R}_\nu} \left|\frac{1}{|\mathcal{A}_j^\nu|} \sum_{(\eta,i)\in\mathcal{A}_j^\nu} (t_i[\eta] + \tau_{ij1}) - t_j[\nu]\right| \tag{3.35}$$

In (3.33) and (3.34), $\mathcal{S}_\nu = \bigcup_{\eta\in\mathcal{V}_\nu} \left\{(\eta,i,j) : \ i \in \mathcal{T}_\eta, \ j \in \mathcal{R}_\nu, \left|t_i[\eta] + \tau_{ij1} - t_j[\nu]\right| \leq \frac{T_0}{2}\right\}$ is the set of triplets containing the indices of transmitters from the ηth clock tick that are contributing to the received signal of the jth device at the νth clock tick. Furthermore in (3.35), we consider the maximum of the average synchronization error, hence, the set $\mathcal{A}_j^\nu = \bigcup_{\eta\in\mathcal{V}_\nu} \left\{(\eta,i) : i \in \mathcal{T}_\eta, \left|t_i[\eta] + \tau_{ij1} - t_j[\nu]\right| \leq \frac{T_0}{2}\right\}$ includes the index of the transmitter devices from the ηth clock tick detected by the jth device at the νth clock tick.

In Fig. 3.9 we compare the synchronization performance of the proposed protocol to [63] using these metrics. Since there is no stopping condition in [63], we set $\lambda_{\text{cons}} = \infty$ in the proposed protocol for a fair comparison, hence, the devices only operate in *Bias Update State* and *Fixed Bias State*. Furthermore, in [63], it is stated that the lower the p_{tr}, the better the synchronization performance. We confirmed this observation by trying several values of $p_{\text{tr}} \in \{0.1, 0.5, 0.9\}$ and we found that $p_{\text{tr}} = 0.1$ indeed yields the best performance. So, in Fig. 3.9 we use $p_{\text{tr}} = 0.1$ for the algorithm in [63]. We see that the proposed protocol reduces the synchronization error and reaches the steady-state much faster, while outperforming [63] in all performance metrics, i.e., (3.33), (3.34), (3.35). In addition, the proposed protocol adapts to the change in the network size, which occurs at $\nu = 33$, more rapidly. Furthermore, we can also observe that there is no fluctuation in the synchronization error as the devices switch from *Bias Update State* to *Fixed Bias State*, which shows that (3.19) produces reliable TO estimates.

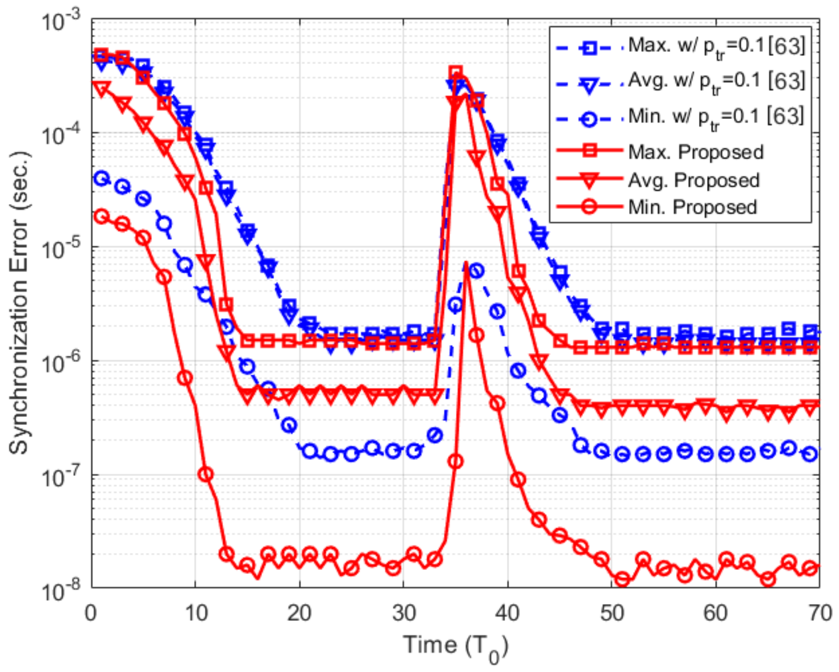


Fig. 3.9. Synchronization error comparison; the proposed protocol vs the random transceiver mode (RTM) in [63] when $J = 14$.

In Fig. 3.10, we investigate the sensitivity of the proposed bias estimation in (3.16) under various initialization of $\widehat{\beta}_j^{\text{init}} = \frac{d}{c}$ by assuming ($\pm\%80$) over and underestimates of the average distance d. We further study the performance of the dynamic step size $\gamma_j[\nu] = a\gamma_j[\nu-2]+b$ and compare it to a fixed step size γ_j. As shown in Fig. 3.10, the proposed protocol still provides an average synchronization error of 3μs even in the presence of severe errors in the estimates of d. Furthermore, the dynamic step size leads to faster error reduction and a lower error floor.

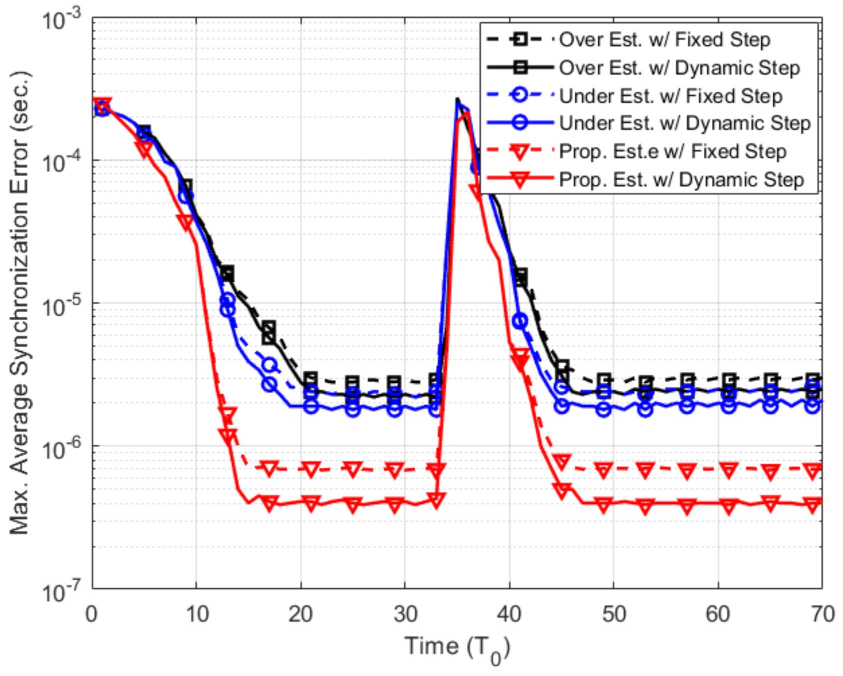


Fig. 3.10. Synchronization performance for dynamic and fixed step size under different bias initializations.

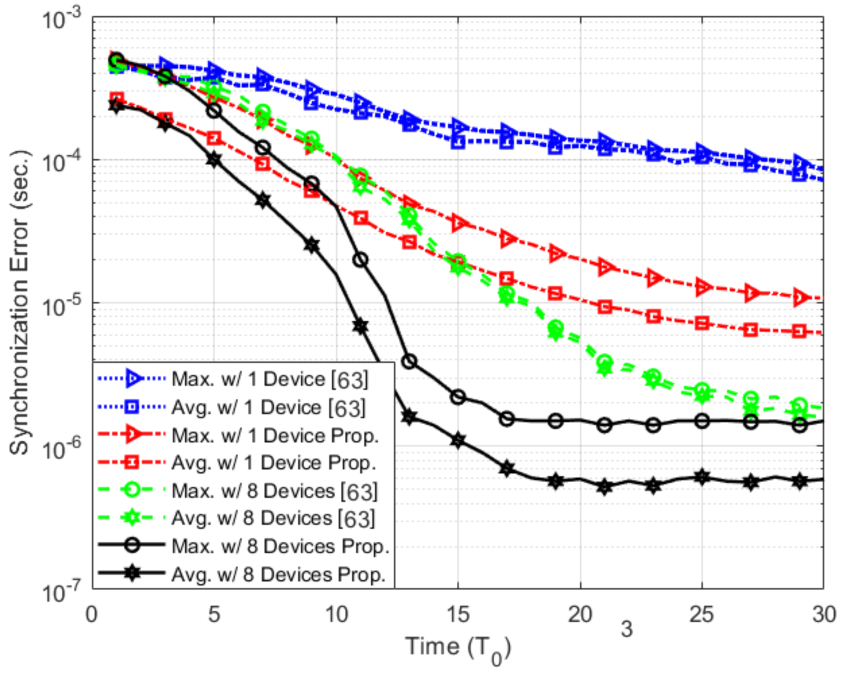


Fig. 3.11. Max. and Avg. synchronization performance based on (3.33) and (3.35) for a partial mesh topology for one and eight common devices.

Next, we consider the performance of the proposed protocol for the partial mesh topology as depicted in Fig. 3.1, where two physically separated device groups are synchronizing through the common devices that act as relaying nodes. In Fig. 3.11, we compare our protocol to [63] with $p_{\mathrm{tr}} = 0.1$. We note that when the number of common devices decreases, the overall synchronization performance degrades as expected. More specifically, if the common devices do not operate as transmitters, which is more likely to happen in the case of random transceiver mode (RTM) in [63], especially if p_{tr} is low, then physically separated devices cannot synchronize themselves. On the contrary, at each clock tick, the devices always alternate their transceiver mode in the proposed protocol, hence, even with a single common device, our protocol attains lower steady-state error.

In Fig. 3.12, we verify the analysis of the synchronization error reduction by Monte-Carlo simulations for different number of devices. It is shown that the analysis tracks well the synchronization error during the steady-state regime regardless of the number of devices. Not surprisingly, the synchronization performance deteriorates as the number of devices increases. However, the performance is better with two devices compared to the setting for multiple devices since the TO estimate in (3.9) only includes the contribution from a single transmitter device.

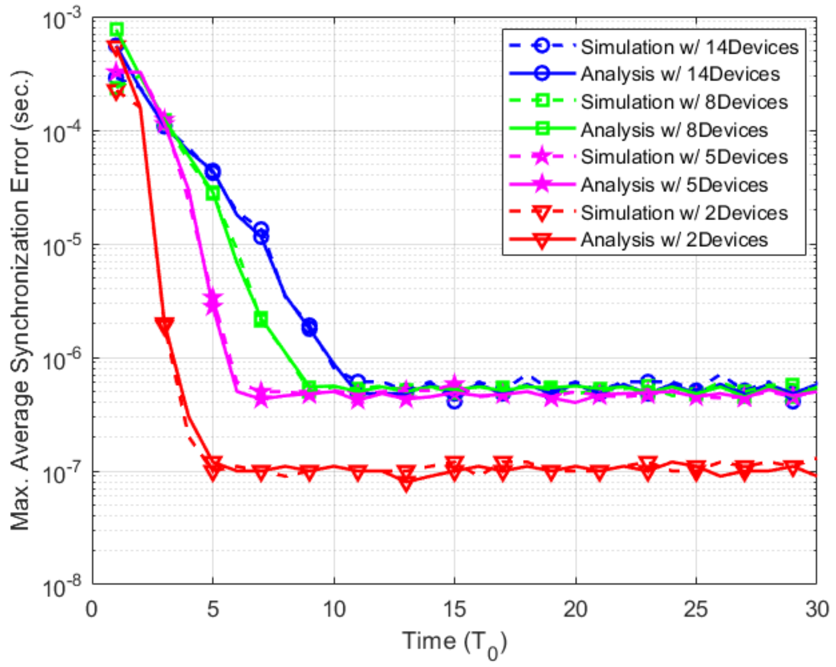


Fig. 3.12. Verification of the analysis with different number of devices.

3.5.3 Timing Error during Data Communication

In Fig. 3.13, we consider the termination of the synchronization process to let the devices initiate timing-advance communication in a distributed manner. Thus, unlike Fig. 3.9, we set $\lambda_{\text{cons}} = 2$ so the devices leave **State 2** after 2 consecutive clock ticks, and proceed to the next state, i.e., *Transition State*, where they will eventually terminate the synchronization process. We note that the performance metrics in (3.33)-(3.35) are not useful after the devices terminate synchronization since $\mathcal{T}_\nu = \emptyset$. Therefore, we investigate the timing error between the communicating devices once they initiate distributed timing-advance communication according to (3.27). To that end, we introduce two performance metrics. The first one is the maximum timing error over all pairs of potentially communicating devices:

$$\Delta t_{comm}^{max}[\nu]{=}\max_{(\eta,i,j)\in\mathcal{C}_\nu} \left|t_i[\eta] - \kappa_i\widehat{\beta}_i[\eta] + \tau_{ij1} - t_j[\nu] + \vartheta_j\widehat{\beta}_j[\nu]\right| \tag{3.36}$$

where $\mathcal{C}_\nu = \left\{(\eta, i, j) \in \mathcal{V}_\nu \times \mathcal{J} \times \mathcal{J} : i \neq j \wedge \left|t_i[\eta] + \tau_{ij1} - t_j[\nu]\right| \leq \frac{T_0}{2}\right\}$. In (3.36), the functions κ_i and ϑ_j indicate whether a device regresses or advances its clock, respectively, according to (3.27) and are defined as follows:

$$\kappa_i = \begin{cases} 0, \text{ if } M_{\nu_0}^i = \text{TX} \\ 1, \text{ if } M_{\nu_0}^i = \text{RX} \end{cases}$$

$$\vartheta_j = \begin{cases} 0, \text{ if } M_{\nu_0}^j = \text{RX} \\ 1, \text{ if } M_{\nu_0}^j = \text{TX} \end{cases}$$

The set $\mathcal{C}_\nu$ in (3.36) contains all triplets (η, i, j) where j is the index of a receiver device at clock tick ν, and i is the index of a device that might transmit to device j from the clock tick $\eta \in \mathcal{V}_\nu$.

The second metric is the maximum over all receivers of the average timing error between a given receiver and all potential transmitters:

$$\Delta t_{comm}^{avg}[\nu]{=}\max_{j\in\mathcal{R}_\nu}\Big|\frac{1}{|\mathcal{H}_j^\nu|}\sum_{(\eta,i)\in\mathcal{H}_j^\nu}(t_i[\eta]-\kappa_i\widehat{\beta}_i[\eta]+\tau_{ij1})-t_j[\nu]+\vartheta_j\widehat{\beta}_j[\nu]\Big| \tag{3.37}$$

where the set $\mathcal{H}_j^\nu = \left\{(\eta, i) \in \mathcal{V}_\nu \times \left\{\mathcal{J} - \{j\}\right\} : \left|t_i[\eta] + \tau_{ij1} - t_j[\nu]\right| \leq \frac{T_0}{2}\right\}$ includes the pairs (η, i) where i is the index of a device that might transmit to device j from the clock tick $\eta \in \mathcal{V}_\nu$.

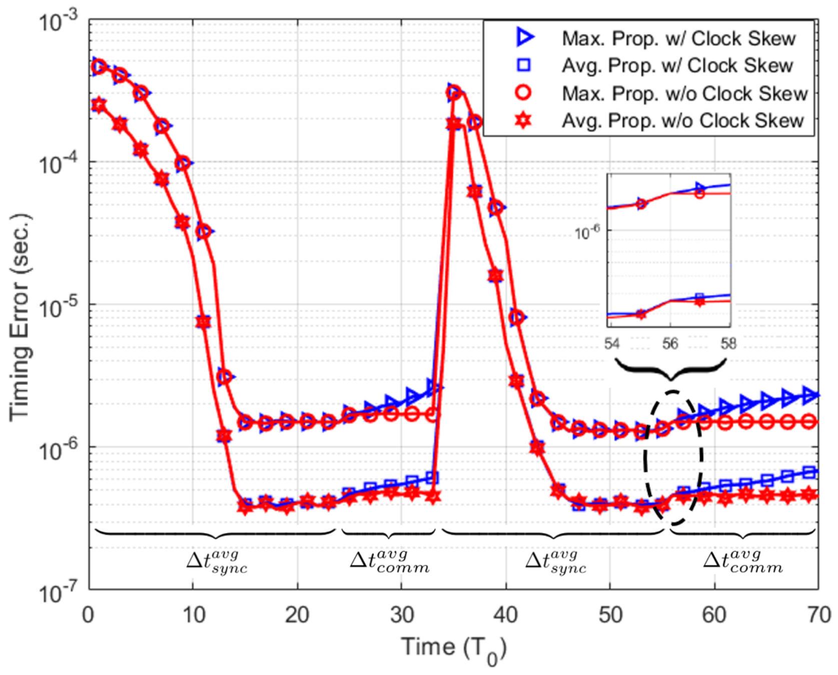


Fig. 3.13. Transitioning from synchronization to data communication in full mesh topology.

Note that in Fig. 3.13, we compare the timing error in two cases: with and without clock skew. We see that the timing error slightly increases in both metrics when the devices stop synchronization and use the clock arrangements in (3.27) as they switch to data communication state at the 25th and 56th clock ticks. Specifically, for the maximum timing error, the degradation is higher since the devices synchronized their clocks based on the weighted average TO and their bias estimates are the approximation of the average propagation delay with respect to the multiple transmitters.

In addition, clock skew increases the timing error since the devices do not employ DPLL update in data communication state. In Table 3.2, we show the change in the timing error due to the clock skew for different number of devices. We assume 20ppm crystal accuracy in our simulations. Here, the second column indicates the timing error right before the synchronization process stops and the third column is the timing error when data communication state starts. As time elapses without DPLL clock updates, the timing error increases. Specifically, after $9T_0$, the timing error is more than 1.5 times the error achieved at the end of the synchronization process. Hence, by using Table 3.2 and assuming that the synchronization error between any device should not exceed 2.5μs, we should set $\lambda_{\text{skew}} = 10$ to re-initiate the synchronization process after $10T_0$. However, if there would not be clock skew, then the devices would use timing-advance data communication without needing to re-initiate the synchronization process.

Table 3.2. Timing-advance data communication performance with clock skew

Comm. Performance	**Sync. Stopped**	**After** $1T_0$	**After** $5T_0$	**After** $9T_0$
Max Sync. Error (14 Dev.)	1.5μs	1.7μs	1.98μs	2.6μs
Avg Sync. Error (14 Dev.)	$.41\mu$s	$.48\mu$s	$.54\mu$s	$.61\mu$s
Max Sync. Error (8 Dev.)	1.35μs	1.58μs	1.79μs	2.1μs
Avg Sync. Error (8 Dev.)	$.39\mu$s	$.43\mu$s	$.5\mu$s	$.59\mu$s
Max Sync. Error (2 Dev.)	$.1\mu$s	$.13\mu$s	$.18\mu$s	$.21\mu$s
Avg Sync. Error (2 Dev.)	$.1\mu$s	$.13\mu$s	$.18\mu$s	$.21\mu$s

3.5.4 Energy-Efficiency of the Synchronization Protocol

Finally, we analyze the total energy consumed during synchronization by considering a network of 5 devices while running the proposed protocol and the random transceiver mode (RTM) algorithm in [63] over the synchronization time $T_{sync} = 50ms$. Note that as the devices may join or leave the network arbitrarily, they may consume less energy compared to the others that are present from the first clock tick throughout the synchronization process even though they achieve the same synchronization error. Therefore, for this particular experiment, we assume no devices join or leave the network after initialization. We consider different power levels for each transceiver mode, namely P_{TX} and P_{RX}, to operate as a transmitter and as a receiver device, respectively. The total energy consumed by the jth device up to the νth clock tick during synchronization is calculated recursively as follows:

$$E_j^{\nu+1} = \begin{cases} E_j^{\nu} + P_{\text{TX}} T_0 \,, & \forall j \in \mathcal{T}_\nu \\ E_j^{\nu} + P_{\text{RX}} T_0 \,, & \forall j \in \mathcal{R}_\nu \end{cases} \tag{3.38}$$

The proposed protocol allows devices to proceed to data communication stage after approximately $15ms$. Then, all the devices operate as a receiver as in Fig. 3.7, where they still keep synchronization but do not actively transmit synchronization signals. Hence, the average energy consumption of the proposed protocol is $E_{sync} = \frac{1}{\mathcal{J}} \sum_{j=1}^{\mathcal{J}} \sum_{\nu=0}^{\infty} E_j^{\nu} = 1.73$mJ, which is 3 and 4.7 times less than the algorithm in [63] with $p_{tr} = 0.5$ and $p_{tr} = 0.9$, respectively. In other words, RTM algorithm not only fails to achieve the same synchronization error level but also consumes more energy except for $p_{tr} = 0.1$. However, we note that in the proposed protocol, 84.5% of the total energy is used for signal transmission, i.e., E_{sync}^{TX}, while only 15.5% of the total energy is used for signal reception, i.e., E_{sync}^{RX}. This means that if the simulation time T_{sync} increases, the energy efficiency of the proposed protocol also increases over time as the devices operate only as a receiver device which consumes less power.

Table 3.3. Energy-efficiency comparisons.

Avg. Energy Consumption	**Proposed Protocol**	**RTM** [63] $p_{tr} = 0.1$	**RTM** [63] $p_{tr} = 0.5$	**RTM** [63] $p_{tr} = 0.9$
E_{sync} (mJ)	1.73	1.25	5.16	8.13
$E_{sync}^{\text{TX}}(\%)$	84.5	76.9	96.9	99.4
$E_{sync}^{\text{RX}}(\%)$	15.5	23.1	3.1	0.6

3.6 Conclusion

In this chapter, we proposed a novel, fully-distributed pulse-based synchronization protocol for half-duplex D2D communications in 5G networks, specifically for the out-of-coverage scenario. The new protocol allows devices to first synchronize themselves in a distributed manner and then simultaneously proceed to timing-advance data communication by maintaining the achieved synchronization. Our protocol also rapidly adapts the changes in the network size by allowing new devices to easily synchronize themselves to the network without disrupting the ongoing synchronization process. Finally, we note that the proposed protocol is not limited to out-of-coverage D2D communications and it can be implemented for any distributed system that demands reliable, high data rate communication. After having implemented the full-fledged synchronization protocol and resolved the synchronization problem, in the next chapter, we address the energy-efficient resource allocation problem in a D2D-aided fog computing scenario.

Chapter 4

D2D-aided Fog Computing under Probabilistic Time Constraints

In this chapter[1], we consider D2D communication as an enabling technology for a task offloading framework while taking the performance instability of device CPUs into account during task processing.

4.1 Introduction

D2D communication is an enabling technology for fog computing by allowing the sharing of computation resources between mobile devices. In this way, DUs can exploit the proximate available computation resources based on different incentive policies to process their computation-intensive tasks. However, mobile devices may suffer CPU throttling in the event of high operating temperatures during task processing as they are not equipped with sophisticated heat-dissipating mechanisms. Naturally, these temperature variations in device CPUs affect the availability of computation resources for task offloading, which unpredictably alters task processing times and overall energy consumption.

In this chapter, motivated by the challenges presented in Section 2.3 of Chapter 2, we focus on the energy-efficient resource allocation strategy for a task offloading framework via D2D communications subject to the uncertainties on CPU performances. Specifically, we address the problem of optimal resource allocation with respect to task partitioning,

[1]Parts of the work presented in this chapter have been published in [132].

computation resources and transmit power in a D2D-aided fog computing scenario, aiming to minimize the expected total energy consumption under probabilistic constraints on the processing time. The main contributions in this chapter are summarized as follows:

- We focus on minimization of the expected total energy consumption under probabilistic constraints on the task processing time. We adopt the partial task offloading scheme to utilize the fog devices in a more flexible manner to increase energy efficiency. Since the formulated problem is non-convex, finding the global optimum is generally intractable; therefore, we propose two sub-optimal solution methods.
- The first sub-optimal method is based on difference of convex (DC) programming, where we develop a CCP algorithm to leverage the DC optimization framework combined with chance-constraint programming to handle the probabilistic constraints. Nonetheless, we find out that the performance of DC programming depends on a good initial point.
- Therefore, we propose a second method that relies on only convex programming, which eliminates the dependence on user-defined initialization. Similar to the first sub-optimal method, we also adopt the chance-constraint method to obtain the deterministic equivalents of the probabilistic constraints.
- As a benchmark, we consider the total energy consumption when a task is processed locally. Simulation results demonstrate that both methods yield higher energy efficiency in comparison to completing the same task only locally. However, the proposed convex-programming method outperforms the DC programming method in terms of energy efficiency and run-time.

4.2 System Model and Problem Formulation

We consider a wireless sub-network comprised of a single active device that has a computationally intensive task to perform and J offloading devices which can be used to offload this task via side-links, as seen in Fig. 4.1. We label the active device by 0 and the offloading devices by $j \in \mathcal{J} = \{1, 2, \ldots\}$. Similar to [133], we assume simultaneous orthogonal side-links to establish D2D communications between the active device and each one of the offloading devices prior to task offloading, which can be achieved using various device association strategies such as distance, availability and maximum utility, etc. [134, 135].

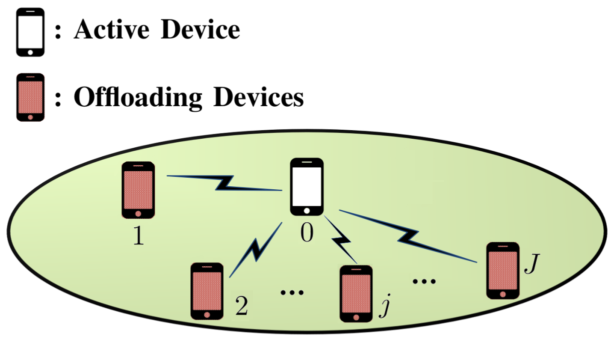


Fig. 4.1. D2D-aided fog computing scenario, where an active device (indexed by 0) can offload its tasks to nearby offloading devices (indexed by $j \in \mathcal{J}$) .

The computation task of the active device is characterized by the tuple $(b, c, t^{\max})$. Here, b indicates the task size in bits, c denotes the number of CPU cycles required to process one bit of data, and $t^{\max}$ is the task processing deadline for completing the task. The device may compute its task locally and/or partially offload it to one or more offloading devices in $\mathcal{J}$. Accordingly, the task size can be decomposed as:

$$b = b_0 + \sum_{j \in \mathcal{J}} b_j \tag{4.1}$$

where b_0 and b_j represent the portions kept at the active device and sent to the jth offloading device, respectively. These portions are collected in the vector $\boldsymbol{b} = [b_0 \; b_1 \; \dots \; b_J]^\top$ where $^\top$ denotes the transpose operation.

To compute the local portion of its task, the active device allocates a part f_0 of its computation resources, measured in CPU cycles per second, which cannot exceed its maximum computation capability $f_0^{\max}$. However, due to unpredictable CPU throttling, e.g., resulting from temperature fluctuations, the actual computation resource used by the device is $\tilde{f}_0 = (1 - \xi_0) f_0$, where $\xi_0 \in [0, 1]$ is a random variable with known distribution. Denoting the time it takes to complete the local portion of the task at the active device as t_0^{co}, we can write:

$$\tilde{f}_0 t_0^{\mathrm{co}} = b_0 c \tag{4.2}$$

The energy consumed for local computation is given by [96]:

$$E^{\mathrm{loc}} = \mu b_0 c \tilde{f}_0^2 = \mu \tilde{f}_0^3 t_0^{\mathrm{co}} \tag{4.3}$$

where μ is an effective capacitance constant that depends on the chip architecture.

The active device also uploads to the jth offloading device the corresponding task portion of size b_j. The achievable data rate for transmission to the jth device is:

$$R_j = W\log_2\left(1 + \frac{P_j G_j}{N_0}\right) \tag{4.4}$$

where P_j is the allocated transmission power, G_j is the channel gain, W is the channel bandwidth, and N_0 is the noise power. Denoting by t_j^{up} the upload time, we have:

$$b_j = R_j t_j^{\text{up}} \tag{4.5}$$

As in the case of the active device, the jth offloading device allocates a part f_j of its computation resources, which cannot exceed its maximum computation capability $f_j^{\max}$, to complete the offloaded task. As before, the actual computation resource used is $\tilde{f}_j = (1-\xi_j)f_j$ where ξ_j is a random variable with known distribution. Then, similar to (4.2), we have:

$$\tilde{f}_j t_j^{\text{co}} = b_j c \tag{4.6}$$

where t_j^{co} is the time it takes the complete the offloaded portion.

Overall, the energy consumed to upload and compute the offloaded tasks is:

$$E^{\text{off}} = \sum_{j\in\mathcal{J}} \left(P_j t_j^{\text{up}} + \mu \tilde{f}_j^3 t_j^{\text{co}}\right) \tag{4.7}$$

Then the total energy consumed to complete the task can be given as a sum of two terms:

$$E = E^{\text{loc}} + E^{\text{off}} = \underbrace{\sum_{j\in\mathcal{J}} \frac{P_j b_j}{R_j}}_{\phi(\mathbf{p},\mathbf{b})} + \underbrace{\sum_{i\in\mathcal{I}} \mu b_i c \tilde{f}_i^2}_{\psi(\mathbf{b},\tilde{\mathbf{f}})} \tag{4.8}$$

where $\mathcal{I} = \{0\} \cup \mathcal{J}$, while $\phi(\mathbf{p}, \mathbf{b})$ and $\psi(\mathbf{b}, \tilde{\mathbf{f}})$ are the total task uploading energy and the total task computation energy, respectively. Furthermore, $\mathbf{p} = [P_1 \ ... \ P_J]^\top$ contains the transmit powers of the active device to its offloading devices and $\tilde{\mathbf{f}} = [\tilde{f}_0 \ \tilde{f}_1 \ ... \ \tilde{f}_J]^\top$ contains the actual computation resources used by the devices.

Finally, we can now formulate the problem of optimal resource allocation, in terms of task portions, computation resources and transmit powers, in the above D2D-aided fog computing scenario. Since the allocated computation resources have a random nature, we aim to minimize the expected value of the total energy consumption subject to probabilistic constraints on the task processing times:

$$\mathscr{P}_1 : \min_{\mathbf{p},\mathbf{b},\mathbf{f},\mathbf{t}} \quad \mathbb{E}_{\boldsymbol{\xi}}\big[E\big] \tag{4.9a}$$

$$\text{s.t.} \quad 0 \leq \sum_{j\in\mathcal{J}} P_j \leq P^{\max} \tag{4.9b}$$

$$\sum_{i\in\mathcal{I}} b_i = b \tag{4.9c}$$

$$\mathbb{P}(t_0^{\text{co}} \leq t^{\max}) \geq \gamma \tag{4.9d}$$

$$\mathbb{P}\big(t_j^{\text{co}} \leq (t^{\max} - t_j^{\text{up}})\big) \geq \gamma, \forall j \in \mathcal{J} \tag{4.9e}$$

$$b_j = R_j t_j^{\text{up}}, \ \forall j \in \mathcal{J} \tag{4.9f}$$

$$f_0 \leq f_0^{\max}, f_j \leq f_j^{\max}, \forall j \in \mathcal{J} \tag{4.9g}$$

$$b_0, b_j, f_0, f_j, t_j^{\text{up}} \geq 0 \quad \forall j \in \mathcal{J} \tag{4.9h}$$

where $\mathbf{f} = [f_0 \ f_1 \ ... \ f_J]^\top$ contains the allocated computation resources and $\mathbf{t} = [t_1^{\text{up}} \ ... \ t_J^{\text{up}}]^\top$ includes the task upload times to each offloading devices. Also, $\mathbb{E}_{\boldsymbol{\xi}}[\cdot]$ is the expectation operator, and $\mathbb{P}(\cdot)$ is the probability of an event.

In problem $\mathscr{P}_1$, the expectation in (4.9a) is taken over the distribution of the random vector $\boldsymbol{\xi} = [\xi_0 \ \xi_1 \ ... \ \xi_J]^\top$, constraint (4.9b) limits the total transmit power of the active device to $P^{\max}$ while constraint (4.9c) guarantees that the task portion sizes add up to the original task size. Constraints (4.9d) and (4.9e) stipulate that the probability of completing the task within the task processing deadline $t^{\max}$ is higher than a given reliability level $\gamma \in [0, 1]$. Constraint (4.9f) ensures that the channel rate and corresponding task uploading time are consistent with the allocated task portions. Finally, constraint (4.9g) indicates that the allocated computation resources cannot exceed the computation capabilities of the devices and constraint (4.9h) expresses the non-negative nature of the decision variables.

4.3 Proposed Sub-Optimal Methods

Due to the non-convex objective function (4.9a) and the non-convex constraints (4.9d), (4.9e), problem $\mathscr{P}_1$ is intractable, we initially considered different evolutionary algorithms such as particle swarm optimization and genetic algorithms to solve it [136, 137]. However, these methods yield low performance with slow convergence for our problem. Therefore, in this section, we propose two sub-optimal methods to efficiently solve problem $\mathscr{P}_1$.

In the first method, we develop a DC algorithm based approach, where we write the non-convex objective function and the non-convex constraints as DCFs, while using chance-constraint programming to handle the probabilistic time constraints. The new optimization problem can then be solved using DC programming. In the second method, to address certain issues related to initialization of the DC programming-based method, we propose a more effective two-step approach which relies solely on convex programming.

4.3.1 DC Programming Method

We start by writing the task uploading energy $\phi(\mathbf{p}, \mathbf{b})$ introduced in (4.8) as a DCF:

$$\phi(\mathbf{p}, \mathbf{b}) = \phi_1(\mathbf{p}, \mathbf{b}) - \phi_2(\mathbf{p}, \mathbf{b}) \tag{4.10}$$

where $\phi_1(\mathbf{p}, \mathbf{b}) = \sum_{j \in \mathcal{J}} (P_j + \frac{b_j}{2R_j})^2$ and $\phi_2(\mathbf{p}, \mathbf{b}) = \sum_{j \in \mathcal{J}} (P_j^2 + \frac{b_j^2}{4R_j^2})$. We also decompose the expected value of the total computation energy $\psi(\mathbf{b}, \tilde{\mathbf{f}})$ as follows:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\xi}}[\psi(\mathbf{b}, \tilde{\mathbf{f}})] &= \mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i \in \mathcal{I}} \mu b_i c (1 - \xi_i)^2 f_i^2\right] \\
&= \mu c \sum_{i \in \mathcal{I}} \eta_i \big((b_i + f_i/2)^2 - (b_i^2 + f_i^2/4)\big) \\
&= \psi_1(\mathbf{b}, \mathbf{f}) - \psi_2(\mathbf{b}, \mathbf{f})
\end{aligned} \tag{4.11}$$

where $\psi_1(\mathbf{b}, \mathbf{f}) = \mu c \sum_{i \in \mathcal{I}} \eta_i (b_i + f_i/2)^2$, $\psi_2(\mathbf{b}, \mathbf{f}) = \mu c \sum_{i \in \mathcal{I}} \eta_i (b_i^2 + f_i^2/4)$ and $\eta_i = \mathbb{E}[(1 - \xi_i)^2]$, $i \in \mathcal{I}$. Finally, the objective function (4.9a) expressed as a DCF:

$$\mathbb{E}_{\boldsymbol{\xi}}[E] \triangleq H(\mathbf{x}) = Y(\mathbf{x}) - Z(\mathbf{x}) \tag{4.12}$$

where $Y(\mathbf{x}) = \phi_1(\mathbf{p}, \mathbf{b}) + \psi_1(\mathbf{b}, \mathbf{f})$ and $Z(\mathbf{x}) = \phi_2(\mathbf{p}, \mathbf{b}) + \psi_2(\mathbf{b}, \mathbf{f})$ are convex functions, and $\mathbf{x} = [\mathbf{p}^\top \mathbf{b}^\top \mathbf{f}^\top \mathbf{t}^\top]^\top$ contains all the search variables for convenience.

As shown in [119], to apply a DC algorithm, each non-convex equality and inequality constraints can be incorporated into (4.12) by using a penalty parameter once their DCFs are available. However, for our problem, we found that this approach yielded slow convergence. Whereas in [122], a DC algorithm is applied to a problem consisting of only non-convex inequality constraints that are decomposed as DCFs. Hence, if we eliminate the equality constraint (4.9f) by incorporating it into (4.12) based on the penalty approach in [119], we can develop a DC-based algorithm as in [122] to solve a problem involving a penalized objective function (which is shown to be DCF [119]) with only inequality constraints (4.9d) and (4.9e).

To this end, we decompose the non-convex equality constraint (4.9f) as:

$$C_j^{\text{eq}}(\mathbf{x}) = \frac{b_j}{R_j} - t_j^{\text{up}} = Y_j^{\text{eq}}(\mathbf{x}) - Z_j^{\text{eq}}(\mathbf{x}),\ \forall j \in \mathcal{J} \tag{4.13}$$

where $Y_j^{\text{eq}}(\mathbf{x}) = (b_j + \frac{1}{2R_j})^2$ and $Z_j^{\text{eq}}(\mathbf{x}) = (b_j^2 + \frac{1}{4R_j^2} + t_j^{\text{up}})$ are convex functions. Then we introduce the penalty term in the objective function, which can be also written as a DCF [119]:

$$H_\lambda(\mathbf{x}) = Y_\lambda(\mathbf{x}) - Z_\lambda(\mathbf{x}) \tag{4.14}$$

where

$$Y_\lambda(\mathbf{x}) = Y(\mathbf{x}) + 2\lambda \sum_{j\in\mathcal{J}} \max\left\{Y_j^{\text{eq}}(\mathbf{x}); Z_j^{\text{eq}}(\mathbf{x})\right\} \tag{4.15}$$

$$Z_\lambda(\mathbf{x}) = Z(\mathbf{x}) + \lambda \sum_{j\in\mathcal{J}} (Y_j^{\text{eq}}(\mathbf{x}) + Z_j^{\text{eq}}(\mathbf{x})) \tag{4.16}$$

and $\lambda \geq 0$ is the penalty parameter.

In order to deal with the probabilistic inequality constraints (4.9d) and (4.9e), we adopt the chance-constraint programming approach [124], and transform them into their deterministic equivalents. Specifically, constraint (4.9d) can be given in terms of the cumulative distribution function (CDF) of ξ_0, $F_{\xi_0}(\cdot)$, as follows:

$$\mathbb{P}\left(\frac{b_0 c}{(1-\xi_0) f_0} \leq t^{\max}\right) = \mathbb{P}\left(\xi_0 \leq \underbrace{\frac{f_0 t^{\max} - b_0 c}{f_0 t^{\max}}}_{z}\right) = F_{\xi_0}(z) \geq \gamma$$

Then, assuming $F_{\xi_0}(\cdot)$ is invertible, we can obtain the deterministic form of constraint (4.9d) as follows:

$$C_0(\mathbf{x}) = \frac{f_0 t^{\max} - b_0 c}{f_0 t^{\max}} - F_{\xi_0}^{-1}(\gamma) \geq 0 \tag{4.17}$$

where $F_{\xi_0}^{-1}(\gamma)$ is the inverse CDF evaluated at γ. The new deterministic constraint (4.17) can now be written as a DCF in the following way:

$$C_0(\mathbf{x}) = \ln(\frac{b_0}{f_0}) - \ln(q_0) = Y_0(\mathbf{x}) - Z_0(\mathbf{x}) \leq 0 \tag{4.18}$$

where $q_0 = t^{\max}c^{-1}(1-F_{\xi_0}^{-1}(\gamma))$ is a non-negative constant, and $Y_0(\mathbf{x}) = -\ln(f_0)$ and $Z_0(\mathbf{x}) = -\ln(b_0) + \ln(q_0)$ are convex functions.

Proceeding in a similar way, the deterministic form of constraint (4.9e) is:

$$C_j(\mathbf{x}) = \frac{f_j(t^{\max} - t_j^{\text{up}}) - b_j c}{f_j(t^{\max} - t_j^{\text{up}})} - F_{\xi_j}^{-1}(\gamma) \geq 0,\ \forall j \in \mathcal{J} \tag{4.19}$$

where $F_{\xi_j}^{-1}(\gamma)$ is the inverse CDF of ξ_j evaluated at γ. In turn, (4.19) can be decomposed as follows:

$$C_j(\mathbf{x}) = \frac{t_j^{\text{up}} f_j q_j}{t^{\max}} + b_j - f_j q_j = Y_j(\mathbf{x}) - Z_j(\mathbf{x}) \leq 0,\ \forall j \in \mathcal{J} \tag{4.20}$$

where $q_j = t^{\max}c^{-1}(1 - F_{\xi_j}^{-1}(\gamma))\ \forall j \in \mathcal{J}$ is a non-negative constant, and $Y_j(\mathbf{x}) = \frac{q_j}{t^{\max}}(t_j^{\text{up}} + f_j/2)^2 + b_j$ and $Z_j(\mathbf{x}) = \frac{q_j}{t^{\max}}\big((t_j^{\text{up}})^2 + f_j^2/4\big) + f_j q_j$ are convex functions.

Our first method is finally obtained by combining the DC programming approach in [122] with the penalized DC approach in [119]; the resulting procedure is presented as Algorithm 4.1. Following initialization, at the kth iteration of the algorithm, we first determine the convex approximations $H_\lambda^{(k)}(\mathbf{x})$ and $C_i^{(k)}(\mathbf{x})$ of $H_\lambda(\mathbf{x})$ and $C_i(\mathbf{x})$ in step 3 and 4, respectively, where ∇ denotes the gradient operator. In step 5, we minimize $H_\lambda^{(k)}(\mathbf{x})$ subject to the indicated constraints using standard convex optimization techniques until the sequence $\{H_\lambda^{(k)}(\mathbf{x})\}$ converges with tolerance ϵ or the maximum iteration number $k^{\max}$ is reached. The algorithm outputs the desired vector $\mathbf{x}^{(k)}$ of the allocated resources.

Algorithm 4.1 DC Algorithm Method

```
1: input Set $k = 0$, initialize $\mathbf{x}^{(0)}$
2: repeat
3:   $H_\lambda^{(k)}(\mathbf{x}) = Y_\lambda(\mathbf{x}) - Z_\lambda(\mathbf{x}^{(k)}) - \nabla Z_\lambda(\mathbf{x}^{(k)})^\top(\mathbf{x} - \mathbf{x}^{(k)})$
4:   $C_i^{(k)}(\mathbf{x}) = Y_i(\mathbf{x}) - Z_i(\mathbf{x}^{(k)}) - \nabla Z_i(\mathbf{x}^{(k)})^\top(\mathbf{x} - \mathbf{x}^{(k)}),\ i \in \mathcal{I}$
5:   solve $\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\text{argmin}}\, H_\lambda^{(k)}(\mathbf{x})$
6:                       s.t. $C_i^{(k)}(\mathbf{x}) \leq 0,\ i \in \mathcal{I}$
7:                            $(4.9b), (4.9c), (4.9g)$ and $(4.9h)$
8:    $k \leftarrow k + 1$
9: until $|H_\lambda(\mathbf{x}^{(k+1)}) - H_\lambda(\mathbf{x}^{(k)})| > \epsilon$ or $k \leq k^{\max}$
10: output $\mathbf{x}^{(k)}$
```

4.3.2 Convex-Programming Method

Although DC programming guarantees a local optimum by converging to a stationary point [138], its performance depends heavily on the choice of the initial point $\mathbf{x}^{(0)}$. To address this limitation, we propose a more effective two-step approach relying solely on convex programming, which eliminates the dependence on user-defined initialization.

Consider an ideal scenario, in which there is no uncertainty in the allocated computation resources and the task uploading is instantaneous, i.e., $\xi_i = 0, \forall i \in \mathcal{I}$ and $t_j^{\text{up}} = 0, \forall j \in \mathcal{J}$. For this scenario let $\mathbf{f}^\star$ and $\mathbf{b}^\star$ be the optimal computation resources and the optimal task partitioning subject to constraint (4.9c), which gives the minimum total energy consumption as $E^\star$. Note that based on (4.2) or (4.6), we have $f_i^\star t_i^{\text{co}} = b_i^\star c, \forall i \in \mathcal{I}$. It can be seen that at the optimal solution, the task completion time must match the given deadline, i.e., $t_i^{\text{co}^\star} = t^{\max}$, since there cannot be any other computation resources, say f_i^+ $\forall i \in \mathcal{I}$ with $f_i^+ < f_i^\star$ that can reduce further the total energy consumption $E^\star$ without violating the time constraint or constraint (4.9c).

Based on the above, we can write the total computation energy in terms of only transmit power and task partitioning by replacing t_0^{co} with $t^{\max}$ and t_j^{co} with $(t^{\max} - t_j^{\text{up}})$ as follows:

$$\psi(\mathbf{p}, \mathbf{b}) = \frac{\mu(b_0 c)^3}{\left(t^{\max}\right)^2} + \sum_{j \in \mathcal{J}} \frac{\mu(b_j c)^3}{\left(t^{\max} - \frac{b_j}{R_j}\right)^2} \tag{4.21}$$

Hence, we decouple the allocation of computation resources and task partitioning in (4.21). More importantly, it can be shown that (4.21) is a convex function over the convex feasible set defined by constraints (4.9b) and (4.9c). Therefore, in the first step of our convex-programming method, we minimize the convex part $\psi(\mathbf{p}, \mathbf{b})$ subject to constraints (4.9b), (4.9c) and a modified form of constraint (4.9f) from problem $\mathscr{P}_1$:

$$\mathscr{P}_2 : \min_{\mathbf{p},\mathbf{b}} \quad \psi(\mathbf{p}, \mathbf{b}) \tag{4.22a}$$

$$\text{s.t.} \quad 0 \le \sum_{j \in \mathcal{J}} P_j \le P^{\max} \tag{4.22b}$$

$$\sum_{i \in \mathcal{I}} b_i = b, \; b_i \ge 0 \tag{4.22c}$$

$$b_j - \alpha R_j t^{\max} \le 0, \forall j \in \mathcal{J} \tag{4.22d}$$

Problem $\mathscr{P}_2$ can be easily solved by means of standard convex optimization methods. In constraint (4.22d), the scaling parameter $\alpha \in (0, 1)$ is used to avoid the task uploading time exceeding the task processing deadline, i.e., $t_j^{\text{up}} > t^{\max}$. In this way, constraint (4.22d) allows the computation time t_j^{co} $\forall j \in \mathcal{J}$ to be within the task processing deadline, and consequently, the solution of $\mathscr{P}_2$ lies in the feasible set of problem $\mathscr{P}_1$. We denote the solution of Problem $\mathscr{P}_2$ as $\mathbf{b}^*$ and $\mathbf{p}^*$.

In the second step, we minimize the expectation of the total computation energy (4.11) with respect to computation resources subject to deterministic equivalents of constraints (4.9d) and (4.9e), wherein the optimal values of $\mathbf{b}^*$ and $\mathbf{p}^*$ from Problem $\mathscr{P}_2$ are used in place of $\mathbf{b}$ and $\mathbf{p}$.

$$\mathscr{P}_3 : \min_{\mathbf{f}} \quad \mathbb{E}_{\boldsymbol{\xi}}[\psi(\mathbf{b}^*, \tilde{\mathbf{f}})] = \sum_{i \in \mathcal{I}} \mu c \eta_i b_i^* f_i^2 \tag{4.23a}$$

$$\text{s.t.} \quad \frac{b_0^* c}{t^{\max}(1 - F_{\xi_0}^{-1}(\gamma))} \leq f_0 \tag{4.23b}$$

$$\frac{b_j^* c}{(t^{\max} - t_j^{\text{up}^*})(1 - F_{\xi_j}^{-1}(\gamma))} \leq f_j, \forall j \in \mathcal{J} \tag{4.23c}$$

Note that $t_j^{\text{up}^*} = \frac{b_j^*}{R_j^*}$ $\forall j \in \mathcal{J}$, where R_j^* is the corresponding data rate for P_j^*; we then form the vector $\mathbf{t}^*$ accordingly. It can be seen that the optimal solution $\mathbf{f}^*$ of problem $\mathscr{P}_3$ can be directly calculated since it satisfies constraints (4.23b) and (4.23c) with equality.

After solving problem $\mathscr{P}_3$, the allocated computation resource at an offloading device, say j, might exceed its computation capability, i.e., $f_j^* > f_j^{\max}$. In this case, the solution of $\mathscr{P}_3$ is not in the feasible set of $\mathscr{P}_1$ as constraint (4.9g) is violated. To address this issue, we reduce f_j^* to $f_j^{\max}$ and we adjust the initially allocated task portion b_j^* so that it can be computed without violating constraint (4.9g). Specifically, we replace b_j^* by

$$b_j^+ = \frac{f_j^{\max} R_j^* t^{\max}(1 - F_{\xi_j}^{-1}(\gamma))}{R_j^* c + f_j^{\max}(1 - F_{\xi_j}^{-1}(\gamma))} \tag{4.24}$$

which is the maximum task portion size that can be computed by utilizing the full available computation resource $f_j^{\max}$. The value of b_j^+ is obtained from constraint (4.23c) by replacing f_j with $f_j^{\max}$. Then the excess task portion, $b_j^* - b_j^+$, is assigned to the active device and/or the rest of the offloading devices. This is done by solving problem $\mathscr{P}_2$ and $\mathscr{P}_3$ after we remove the jth device from the set of available offloading destinations, i.e., we replace $\mathcal{J}$ with $\mathcal{J} - \{j\}$. The process is repeated until constraint (4.9g) is no longer violated by the remaining offloading devices. If the set $\mathcal{J}$ becomes empty, then the leftover portion of the task size is computed at the active device, where we assume that $\frac{b_0^* c}{t^{\max}(1-F_{\xi_j}^{-1}(\gamma))} = f_0^* < f_0^{\max}$ based on constraint (4.23b). Finally, we present the overall progress of our second method in Algorithm 4.2.

Algorithm 4.2 Convex-Programming Method

```
1:  Solve $\mathscr{P}_2$ and $\mathscr{P}_3$ to obtain $\mathbf{p}^*,\ \mathbf{b}^*,\ \mathbf{f}^*$and $\mathbf{t}^*$
2:  for $j \in \mathcal{J}$ do
3:      if $f_j^* > f_j^{\max}$ then
4:          $f_j^* \leftarrow f_j^{\max}$
5:          Calculate the new task size $b_j^+$ using (4.24)
6:          $b_j^* \leftarrow b_j^+$
7:          $b \leftarrow b - b_j^+$
8:          $P^{\max} \leftarrow P^{\max} - P_j^*$
9:          Disregard the $j$th device: $\mathcal{J} \leftarrow \mathcal{J} - \{j\}$
10:         Update $\mathbf{p}^*, \mathbf{b}^*, \mathbf{f}^*$and $\mathbf{t}^*$ by solving $\mathscr{P}_2$ and $\mathscr{P}_3$
11:         Go to line 2
12:     end
13: end
14: Output $\mathbf{p}^*,\ \mathbf{b}^*,\ \mathbf{f}^*$and $\mathbf{t}^*$
```

4.4 Simulation Results

In this section, we compare the energy efficiency and run-time of the proposed methods through Monte Carlo simulations over 4×10^4 independent realizations. In each simulation run, we uniformly place the offloading devices on a disk with a radius set to 15 m centered at the active device. Furthermore, we consider independent Rayleigh fading channels and distance-dependent path loss model, $\text{PL} = 148 + 40\log_{10}(d)$ in dB, where d is the distance in km [139]. As a benchmark we calculated a lower bound on the total energy consumption subject to the uncertainties in the allocated computation resources. To derive the lower bound, similar to the ideal scenario assumed in Appendix E, we consider instantaneous task uploading times and infinite computation resources, i.e., $t_j^{\text{up}} = 0$ and $f_j^{\max} = \infty$, $\forall j \in \mathcal{J}$, respectively. For this ideal scenario as stated in Appendix E, the total energy consumption is minimized when the task sizes are equally distributed among the offloading destinations, i.e., $b_i = \dfrac{b}{|\mathcal{I}|}, \ \forall i \in \mathcal{I}$. Hence, a lower bound on the total energy consumption subject to the probabilistic task processing times can be calculated as follows

$$\min_{\mathbf{f}} \quad \mathbb{E}_{\boldsymbol{\xi}}\big[E\big] = \sum_{i\in\mathcal{I}} \mu c \eta_i b_i f_i^2 \tag{4.25a}$$

$$\text{s.t.} \quad \mathbb{P}(t_i^{\text{co}} \leq t^{\max}) \geq \gamma, \forall i \in \mathcal{I} \tag{4.25b}$$

where the deterministic equivalent of constraint (4.25b) can be obtained as $\frac{b_i c}{t^{\max}(1-F_{\xi_i}^{-1}(\gamma))} \leq f_i, \ \forall i \in \mathcal{I}$ based on the chance-constraint programming. In addition to the lower bound, we also include the total energy consumption when $\mathcal{J} = \emptyset$, i.e., the task is completed locally. For the CPU throttling we assume that $\xi_i, \ i \in \mathcal{I}$, are uniform $\mathcal{U}(0, 0.1)$, i.e., the actual computation resources may be below the allocated ones by up to 10%. We select the task size b from a uniform distribution $\mathcal{U}(2\times 10^4, 4\times 10^5)$, and we set $f_0^{\max}$ to a large value such that the assumption in the previous section holds. The rest of the system parameters are given in Table 4.1.

Table 4.1. System parameters

Parameter Description	Symbol	Value
Number of offloading devices	J	$\{1, 2, 3\}$
CPU cycles to process 1-bit data	c	1500 cycles/bit
Effective capacitance constant	μ	10^{-24} Ws^3
Max. iteration for DC prog.	$k^{\max}$	10^3
Max. transmit power	$P^{\max}$	200 mW
Task processing deadline	$t^{\max}$	$[.4, 1]$ s
Limiting term for task uploading time	α	.85
Reliability level	γ	.95
Convergence tolerance for DC prog.	ϵ	10^{-2}
Penalty parameter for DC prog.	λ	12
Max. radius of a D2D link	-	20 m
Noise level	N_0	-114 dBm
Channel bandwidth	W	10 MHz

In Fig. 4.2, we investigate the effect of the task processing deadline to complete the task. To simulate different computation capabilities of the offloading device we select $f_j^{\max}$ from a uniform distribution $\mathcal{U}(3 \times 10^7, 1 \times 10^8)$ for each simulation run. As seen from Fig. 4.2, regardless of the maximum time constraint, both of our methods significantly reduce the total energy consumption compared to local task computation. However, the performance of the convex-programming method outperforms DC programming in terms of energy efficiency. Specifically, $t^{\max} = 0.4$s, the total energy consumption with our convex-programming method requires almost 30% less energy to compute the same task with respect to computing it only at the local device. However, the performance of the proposed methods deviates from the given lower bounds especially when $t^{\max}$ is very small since more computation resources must be allocated to process tasks in order to meet the processing deadline. Nonetheless, it can be shown that when the constraint on the maximum time limit is relaxed, the performance of the proposed methods approach to a lower bound. Note that by increasing the task processing deadline, we can reduce the required computation resources, which naturally lowers the energy consumption. However, this negatively impacts the quality of service of the given task in terms of latency.

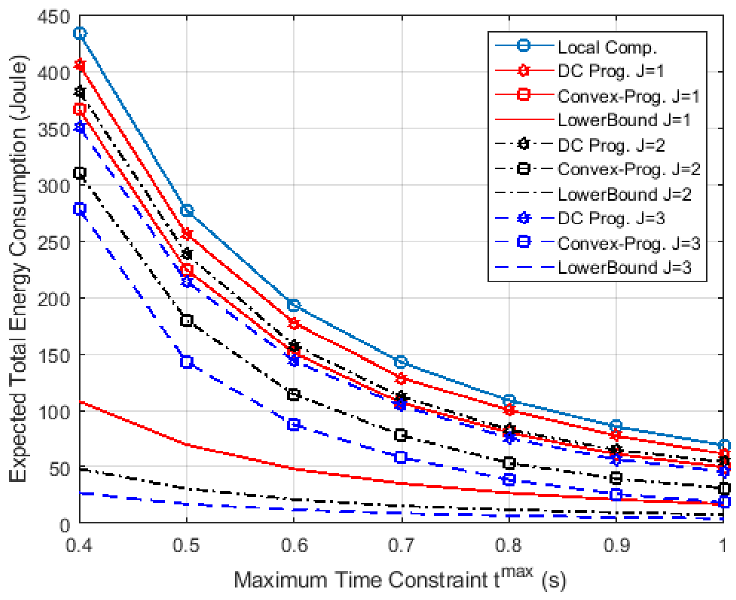


Fig. 4.2. Expected total energy consumption versus $t^{\max}$ for different numbers of offloading devices ($\gamma = .95$).

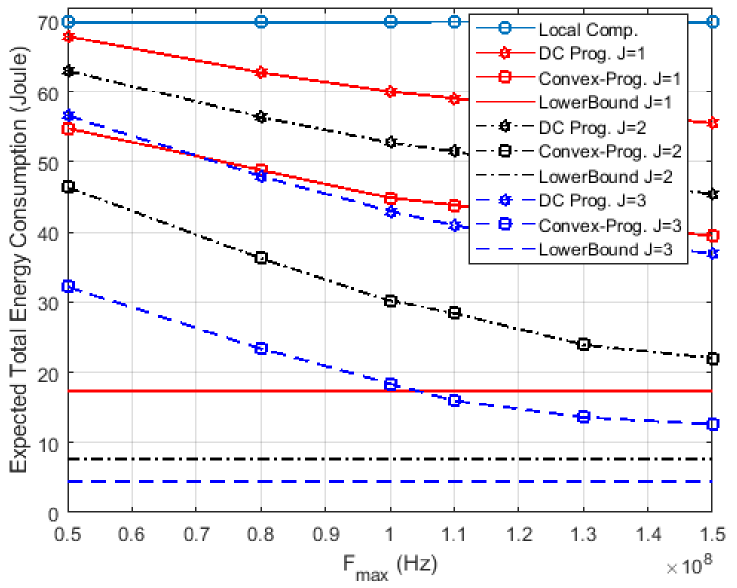


Fig. 4.3. Expected total energy consumption versus $\mathrm{F}_{\max}$ for different numbers of offloading devices ($t^{\max} = 1$, $\gamma = .95$).

In Fig. 4.3, we consider the effect of maximum computation resources at the offloading devices on the total energy consumption. Specifically, we select $f_j^{\max}$ from a uniform distribution $\mathcal{U}(\mathrm{F}_{\min}, \mathrm{F}_{\max})$, where $\mathrm{F}_{\min}$ is set to 3×10^7 Hz while $\mathrm{F}_{\max}$ is ranging from $5\times10^7 - 1.5\times10^8$ Hz. Even though increasing the number of offloading devices drastically reduces the energy consumption, the amount of available computation resources at the offloading devices limits the energy efficiency during task offloading. In comparison to the lower bound, the performance of the proposed methods deviates from the calculated lower bound as $\mathrm{F}_{\max}$ decreases. However, as the constraint on the computation resources is relaxed by increasing $\mathrm{F}_{\max}$, the performance gap between the proposed methods and the lower bound diminishes. Therefore, reducing the total energy consumption not only depends on the number of nearby devices but is also highly affected by the amount of available computation resources that can be allocated by the offloading devices.

Finally, in Table 4.2 we compare the average run-time of proposed methods implemented in MATLAB on an Intel i7-3770 computer with 16GB RAM. The proposed convex-programming based method not only achieves a better performance compared to our DC programming approach in terms of energy efficiency but also its run-time is significantly shorter. Specifically, with the increased number of offloading devices, DC programming takes at least ten times longer to converge within the selected tolerance value ϵ. In addition, we observe that the required number of iterations for DC programming to converge is more than three times compared to our second method that is iteratively running Algorithm 4.2.

Table 4.2. Average run-time comparison ($t^{\max} = .4$)

Simulation setup		DC Prog. Method	Convex-Prog. Method
$\mathrm{F}_{\max} = 4 \times 10^7$	$J = 1$	3.98 s	0.40 s
$\mathrm{F}_{\max} = 1 \times 10^8$	$J = 1$	3.64 s	0.40 s
$\mathrm{F}_{\max} = 4 \times 10^7$	$J = 2$	8.26 s	0.41 s
$\mathrm{F}_{\max} = 1 \times 10^8$	$J = 2$	7.53 s	0.40 s
$\mathrm{F}_{\max} = 4 \times 10^7$	$J = 3$	11.88 s	0.46 s
$\mathrm{F}_{\max} = 1 \times 10^8$	$J = 3$	11.60 s	0.42 s

4.5 Conclusion

In this chapter, we addressed the problem of optimal resource allocation in a D2D-aided fog computing scenario under probabilistic time constraints. Since the formulated problem was nonconvex, we proposed two sub-optimal methods to solve it. The first method relies on DC programming, however, its performance is very sensitive to the choice of the initial point. Hence, we proposed a novel alternative solution based on convex programming, which eliminates the dependence on user-defined initialization. Nevertheless, due to the uncertainties on the allocated computation resources, we incorporated chance-constraint programming into both methods. While both proposed sub-optimal task offloading methods significantly reduce the total energy consumption compared to computing the task locally, the second method outperforms DC programming in terms of energy efficiency and run-time. In the next chapter, we investigate a resource allocation problem by extending the considered fog computing scenario with multiple active and idle devices as well as a central node.

Chapter 5

Energy-Efficient Resource Allocation for D2D-aided Fog Computing

In this chapter, we focus on the minimization of the total energy consumption in a more general fog computing scenario, wherein multiple devices can offload their tasks to nearby fog devices via D2D links and to a centralized ES.

5.1 Introduction

While Cloud computing can provide infinite amount of resources over the internet for devices with computation-intensive tasks, wireless channel conditions and network traffic may limit its utilization for real-time applications with strict processing deadlines. On the contrary, MEC can be an alternative to Cloud computing by taking advantage of servers located at the edge of the network, hence, reduce end-to-end delays and total overheads in the network. However, in comparison to cloud servers, the computation capabilities of ESs are limited, which negatively affects the performance of MEC. In this regard, as a complement to Cloud computing and MEC, fog computing provides an energy-efficient task offloading by leveraging the close proximity of all available computation resources, and in turn, offering ultra-low latency. To this end, D2D communications technology as a part of the future generation networks, can enable fog computing to mitigate the task computation burden of wireless devices by reducing the total energy consumption and task processing times.

Motivated by the advantages of a hybrid-task offloading scheme and the recent literature given in Section 2.3 of Chapter 2, we focus on the minimization of the total energy consumption in a more general fog computing scenario in comparison to the one given in Chapter 4. Specifically, we consider multiple wireless devices with computation-intensive tasks, which can be offload to nearby fog devices via D2D links *and* to a centralized ES. However, different than Chapter 4, we assume deterministic task processing time at each device due to the additional complexity imposed by handling the probabilistic time constraints together with the joint allocation of the computation resources at the ES. We adopt the partial task offloading scheme with transmit power management to improve the utilization, that is, minimize the total energy consumption over the considered network. On the one hand, the ES has more computation capability compared to the fog devices, and hence can process more tasks simultaneously; however, achievable data rates on the ES links may limit the task uploading speed. On the other hand, D2D-aided fog computing can take advantage of close proximity, and hence yields higher data rates and reduces the task offloading time. Therefore, depending on wireless channel conditions as well as constraints on delays, transmit powers and computation resources, a device can either locally compute its task or offload some of its portions for distributed data processing. Within this extended framework, the main contributions of this chapter are summarized as follows:

- We consider a D2D-assisted fog computing scenario, where multiple cellular devices can partially offload their computation-intensive tasks not only to the ES but also to nearby fog devices via D2D links. Our main objective is to develop an optimal resource allocation strategy by minimizing the total energy consumption subject to the constraints on transmit powers, computation resources and task processing times.

- The formulated problem is non-convex and its optimal solution is generally intractable, hence, we propose two sub-optimal methods. For the first method, we begin with investigating the relationship between the task processing time and the total energy consumption. By exploiting this relationship, we then show how the original problem can be relaxed into a sequence of convex subproblems whose solutions can be efficiently obtained via standard convex optimization methods.

- While our first method achieves good performance, its run time is relatively high. To remedy this limitation, we propose a second method, which targets similar goals as the first one, but relies on a low-complexity heuristic resource allocation strategy, thereby avoiding costly calculations of gradients and Hessian matrices in the solution process. We analyze in detail the computational complexity of this method in terms of key system parameters, including the number of mobile devices and task offloading destinations.

- We develop a lower bound on the total energy consumption for the considered task offloading scenario as a performance benchmark for comparison purpose. Computer simulations under a wide range of conditions and parameter settings show that both methods approach the lower bound for a wide range of practical conditions, while the second method leads to a quite significant reduction in run time.

5.2 System Model and Problem Formulation

In this section, we first present the D2D-aided fog computing scenario under study and the associate computation and communication models. We then formulate the problem of optimal resource allocation as a non-convex program.

5.2.1 System Model

We consider a stationary network consisting of a single BS and I active cellular devices with computationally intensive tasks. Each active device can offload its task to an ES connected to the BS via a high-speed link, and up to K nearby fog devices associated to that user via D2D links, as shown in Fig. 5.1. Hence, we identify the available task offloading destinations of the ith active device by a 2-tuple $\kappa \in \mathcal{K}_i = \{(i,0),\ (i,1),...,(i,K)\},\ i \in \mathcal{I} = \{1,...,I\}$. We assume that all active devices can utilize the same ES, indexed by 0, while the K fog devices associated to different active devices are distinct. In the sequel, to simplify the notations, we represent the 2-tuple (i,j) simply as ij. Similar to [133], the assignment of fog devices to the active ones is decided beforehand based on various criteria: distance, availability, and incentive etc.

The computation task of the ith active device is characterized by the tuple $(d_i, c_i, t_i^{\max})$. Here, d_i indicates the task size in bits, c_i denotes the average number of CPU cycles required to process one bit of data, and $t_i^{\max}$ is the task processing deadline[1]. The ith active device

[1]That is, the latest time by which processing, including task uploading time if applicable, must be completed.

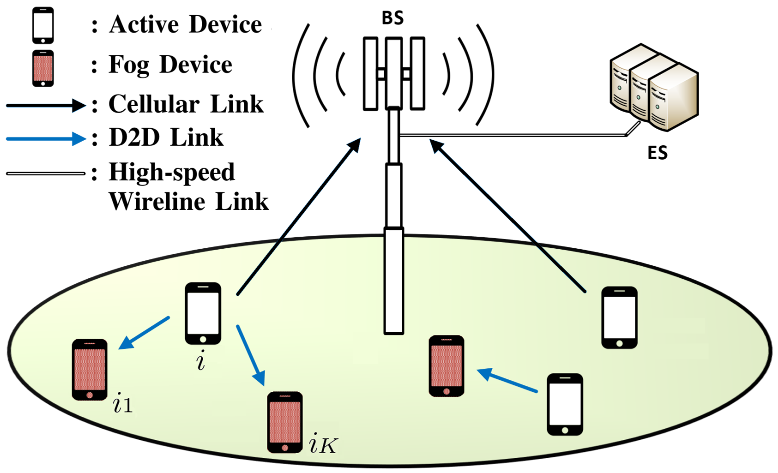


Fig. 5.1. D2D-aided fog computing scenario, where active devices can offload their tasks to nearby fog devices as well as to a more powerful central ES.

may offload parts of its task to available destinations. Accordingly, the computation task size of the ith device can be decomposed as:

$$d_i = b_i + \sum_{\kappa \in \mathcal{K}_i} b_\kappa \tag{5.1}$$

where b_i and b_κ indicate the sizes of the task portions kept at the local device and sent to the κth offloading destination, respectively. The task portion sizes for the ith active device form the vector $\boldsymbol{b}_i = [b_i \; b_{i0} \; b_{i1} \; ... \; b_{iK}]^\top$ with $\|\boldsymbol{b}_i\|_1 = d_i$.

Let f_i, expressed in cycles per second, represent the computation resources allocated by the ith active device to process the local portion of its task. Then, denoting the time taken by the ith active device to compute the local portion of the task as t_i^{co}, we can write:

$$b_i = \frac{f_i t_i^{\text{co}}}{c_i}, \; \forall i \in \mathcal{I}. \tag{5.2}$$

Accordingly, the energy consumption for processing this task is:

$$E_i^{\text{loc}} = \mu b_i c_i f_i^2 = \mu f_i^3 t_i^{\text{co}}, \; \forall i \in \mathcal{I} \tag{5.3}$$

where μ is an effective capacitance constant depending on the chip architecture of the devices [96].

During task offloading, the ith active device uploads b_κ bits to its κth offloading destination over a wireless communication link. The achievable data rate over this link is given by:

$$R_\kappa = W \log_2\left(1 + \frac{P_\kappa G_\kappa}{N_0}\right),\ \forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I} \tag{5.4}$$

where P_κ is the allocated transmit power, G_κ is the channel gain, W is the channel bandwidth, and N_0 is the thermal noise power.

Let t_κ^{up} be the time needed to upload a task portion with size b_κ from the ith active device to the κth offloading destination. Based on the given data rate, the above quantities should satisfy the following relation:

$$b_\kappa = R_\kappa t_\kappa^{\text{up}} \tag{5.5}$$

Similar to (5.2), let f_κ is the computation resources allocated by the κth destination to process the task of size b_κ. Then denoting the time taken by the κth device to compute this task size as t_κ^{co}, we have:

$$b_\kappa = \frac{f_\kappa t_\kappa^{\text{co}}}{c_i},\ \forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I} \tag{5.6}$$

Then, the energy consumption for uploading and processing the task sizes at the offloading destinations of the ith device is given by:

$$E_i^{\text{off}} = \sum_{\kappa \in \mathcal{K}_i} \left(P_\kappa t_\kappa^{\text{up}} + \mu f_\kappa^3 t_\kappa^{\text{co}}\right),\ \forall i \in \mathcal{I} \tag{5.7}$$

Finally, the total energy consumption required to complete the overall task for the ith active device is:

$$E_i = E_i^{\text{loc}} + E_i^{\text{off}},\ \forall i \in \mathcal{I} \tag{5.8}$$

5.2.2 Problem Statement

We formulate the optimal resource allocation problem in a D2D-aided fog computing scenario to minimize the total energy consumption subject to the constraints on computation resources, transmit powers and task processing deadlines. To this end, the overall problem is given as follows:

$$\mathscr{P}_1 : \min_{\mathbf{p},\mathbf{f},\mathbf{b}} \quad \sum_{i\in\mathcal{I}} E_i \tag{5.9a}$$

$$\text{s.t.} \quad 0 \leq \sum_{\kappa\in\mathcal{K}_i} \|b_\kappa\|_0 P_\kappa \leq P^{\max}, \ \forall i \in \mathcal{I} \tag{5.9b}$$

$$\sum_{i\in\mathcal{I}} \|b_{i0}\|_0 f_{i0} \leq f_0^{\max}, \tag{5.9c}$$

$$b_i + \sum_{\kappa\in\mathcal{K}_i} \|P_\kappa\|_0 b_\kappa = d_i, \ \forall i \in \mathcal{I} \tag{5.9d}$$

$$0 \leq t_i^{\text{co}} \leq t_i^{\max}, \ \forall i \in \mathcal{I} \tag{5.9e}$$

$$0 \leq t_\kappa^{\text{up}} + t_\kappa^{\text{co}} \leq t_i^{\max}, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{5.9f}$$

$$0 \leq b_i, b_\kappa, f_i, f_\kappa, P_\kappa \quad \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{5.9g}$$

where the matrix $\mathbf{p} = [\mathbf{p}_1^\top \ \mathbf{p}_2^\top \ ... \ \mathbf{p}_I^\top]^\top$ with $\mathbf{p}_i = [P_{i0} \ P_{i1} \ ... \ P_{iK}]^\top$ contains the allocated transmit powers of each active device to its offloading destinations. In addition, the matrix $\mathbf{f} = [\mathbf{f}_1^\top \ \mathbf{f}_2^\top \ ... \ \mathbf{f}_I^\top]^\top$ with $\mathbf{f}_i = [f_i \ f_{i0} \ f_{i1} \ ... \ f_{iK}]^\top$ contains the allocated computation resources, and the matrix $\mathbf{b} = [\boldsymbol{b}_1^\top \ \boldsymbol{b}_2^\top \ ... \ \boldsymbol{b}_I^\top]^\top$ contains the task splitting decisions of each active device.

In problem $\mathscr{P}_1$, constraint (5.9b) limits the total transmit power at the ith device to $P^{\max}$, constraint (5.9c) restricts the ES to allocate its computation resource beyond its maximum computation capability $f_0^{\max}$, and constraint (5.9d) ensures that the task splitting adds up to original task size. Constraint (5.9d) ensures that the task splitting adds up to original task size, while constraints (5.9e) and (5.9f) require that the time to complete the task does not exceed the task processing deadline $t_i^{\max}$ $\forall i \in \mathcal{I}$. Finally, constraint (5.9g) denotes that the decision variables must be non-negative.

The presence of ℓ_0 norms in the constraints couples the decision variables. In this way, transmit powers and computation resources cannot take arbitrary values if active devices do not offload their tasks. At this point, problem $\mathscr{P}_1$ can be re-formulated as mixed-integer program to avoid ℓ_0 norms however, non-convex objective functions and constraints still impose difficulty as they make the problem intractable. Therefore, in the next section, we propose a sub-optimal solution, which depends solely on convex programming.

5.3 Convex Programming Method

In this section, we first analyze problem $\mathscr{P}_1$ and then show how it can be relaxed into convex subproblems by considering its main limitations. Finally, we present the overall procedures to obtain the proposed method in an algorithm for convenience.

5.3.1 Problem Analysis

We begin with analyzing the task processing times at the optimal solution of problem $\mathscr{P}_1$. Let $\mathbf{p}^\star, \mathbf{f}^\star$ and $\mathbf{b}^\star$ be the global minimizers of problem $\mathscr{P}_1$ yielding the total energy consumption as $E^\star = E^{\text{loc}^\star} + E^{\text{off}^\star}$. At the optimal solution, we have $d_i = b_i^\star + \sum_{\kappa \in \mathcal{K}_i} b_\kappa^\star$ and the corresponding computation resources for these task portions are $f_i^\star$ and $f_\kappa^\star$, respectively. Assume that the optimal task processing time for the local portion of the task is smaller than the task processing deadline $t_i^{\text{co}^\star} = \frac{b_i^\star c_i}{f_i^\star} < t_i^{\max}$. Then there can be $f_i^\ddagger < f_i^\star$ such that $t_i^{\text{co}^\star} < \frac{b_i^\star c_i}{f_i^\ddagger} = t_i^{\max}$, which yields $E_i^{\text{loc}^\ddagger} = \mu b_i^\star c_i {f_i^\ddagger}^2 < E_i^{\text{loc}^\star}$, and consequently reduces the total energy consumption $E_i^\ddagger = E^{\text{loc}^\ddagger} + E^{\text{off}^\star} < E_i^\star$ further. Therefore, $f_i^\star$ cannot be the global minimizer. Similarly, assume that the optimal total task offloading time is smaller than the task processing deadline, $t_\kappa^{\text{up}^\star} + t_\kappa^{\text{co}^\star} = t_\kappa^{\text{off}^\star} < t_i^{\max}$, or equivalently by using (5.5) and (5.6), we have $\frac{b_\kappa^\star}{R_\kappa^\star} + \frac{b_\kappa^\star c_i}{f_\kappa^\star} = t_\kappa^{\text{off}^\star} < t_i^{\max}$, where $R_\kappa^\star$ is the data rate calculated by using the optimal transmit power $P_\kappa^\star$. However, there can be $f_\kappa^\ddagger < f_\kappa^\star$ such that $\frac{b_\kappa^\star}{R_\kappa^\star} + \frac{b_\kappa^\star c_i}{f_\kappa^\ddagger} = t_i^{\max}$ or $P_\kappa^\ddagger < P_\kappa^\star$ such that $\frac{b_\kappa^\star}{R_\kappa^\ddagger} + \frac{b_\kappa^\star c_i}{f_\kappa^\star} = t_i^{\max}$, in which both $f_\kappa^\ddagger$ and $P_\kappa^\ddagger$ yield smaller energy consumption for task offloading, i.e., $E_i^{\text{off}^\ddagger} < E_i^{\text{off}^\star}$. Consequently, $f_\kappa^\star$ and $P_\kappa^\star$ cannot be the optimal minimizers since the total energy consumption in (5.8) can be reduced further. Therefore, at the optimal solutions $\mathbf{p}^\star, \mathbf{f}^\star$ and $\mathbf{b}^\star$, the time it takes the process a given task must match the deadline if the goal is to minimize energy consumption.

Based on the above, we can modify the objective function (5.9a) by replacing t_i^{co} with $t_i^{\max}, \forall i \in \mathcal{I}$ and t_κ^{co} with $(t_i^{\max} - t_\kappa^{\text{up}}), \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i$, and write the total energy consumption of the ith device as a combination of two terms $E_i = \xi_i(\mathbf{p}_i, \mathbf{b}_i) + \psi_i(\mathbf{p}_i, \mathbf{b}_i), i \in \mathcal{I}$. Specifically, the first term is the total energy consumption for uploading the task size portions:

$$\xi_i(\mathbf{p}_i, \mathbf{b}_i) = \sum_{\kappa \in \mathcal{K}_i} \frac{P_\kappa b_\kappa}{R_\kappa} \tag{5.10}$$

while the second term is the total computation energy and obtained from (5.3) and (5.7) as follows:

$$\psi_i(\mathbf{p}_i, \mathbf{b}_i) = \frac{\mu(b_i c_i)^3}{\left(t_i^{\max}\right)^2} + \sum_{\kappa \in \mathcal{K}_i} \frac{\mu(b_\kappa c_i)^3}{\left(t_i^{\max} - \frac{b_\kappa}{R_\kappa}\right)^2} \tag{5.11}$$

Note that reducing both terms simultaneously in E_i could not be possible since decreasing transmit power P_κ might reduce the first term, but it increases the second one due to elevated uploading time $t_\kappa^{\text{up}} = \frac{b_\kappa}{R_\kappa}$. Fortunately, *total computation energy* given in (5.11) is a convex function on a convex domain as demonstrated in Appendix D. Hence, by leveraging the convexity of total computation energy (5.11), we can now decouple problem $\mathscr{P}_1$ into convex sub-problems that can be solved using standard techniques.

5.3.2 Allocation of Transmit Powers and Task Sizes

First, we allocate the transmit powers and task sizes. Specifically, we determine the optimal task splitting under transmit power constraints that, in turn, limit the data rates, without considering the available computation resources at the ES:

$$\mathscr{P}_2 : \min_{\mathbf{p},\mathbf{b}} \sum_{i \in \mathcal{I}} \psi_i(\mathbf{p}_i, \mathbf{b}_i) \tag{5.12a}$$

$$\text{s.t.} \quad 0 \leq \sum_{\kappa \in \mathcal{K}_i} P_\kappa \leq P^{\max}, \ \forall i \in \mathcal{I} \tag{5.12b}$$

$$b_i + \sum_{\kappa \in \mathcal{K}_i} b_\kappa = d_i, \ \forall i \in \mathcal{I} \tag{5.12c}$$

$$b_\kappa - \alpha R_\kappa t_i^{\max} \leq 0, \ \forall i \in \mathcal{I}, \ \forall \kappa \in \mathcal{K}_i \tag{5.12d}$$

$$0 \leq b_i, b_\kappa \quad \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{5.12e}$$

In constraint (5.12d) we include a factor $\alpha \in (0,1)$ to limit the task uploading time and provide sufficient time for task computation at the offloading destinations $t_\kappa^{\text{co}} = (1-\alpha)t_i^{\max}$. This naturally prevents the violation of constraint (5.9f) in problem $\mathscr{P}_1$, and consequently, allows the solution of problem $\mathscr{P}_2$ to be in the feasible solution set of the main problem. However, we note that the choice of α cannot be arbitrary since it plays an important role in the convexity of problem $\mathscr{P}_2$ as demonstrated in Appendix D. Then, we allocate the computation resources corresponding to the transmit powers and the task splitting decisions $\mathbf{p}^*$ and $\mathbf{b}^*$ obtained by solving problem $\mathscr{P}_2$.

5.3.3 Limitation of Computation Resources

To determine the computation resources corresponding to $\mathbf{p}^*$ and $\mathbf{b}^*$, we first calculate the task uploading times using (5.5) as $t_\kappa^{\mathrm{up}^*} = \frac{b_\kappa^*}{R_\kappa^*}, \forall i \in \mathcal{I},\ \forall \kappa \in \mathcal{K}_i$, where R_κ^* is the data rate calculated by using the transmit power P_κ^* (cf. eq. (5.4)). Then, using the remaining time $t_\kappa^{\mathrm{co}^*} = t_i^{\max} - t_\kappa^{\mathrm{up}^*}$, we allocate the computation resources at the κth offloading destination as follows:

$$f_\kappa^* = \frac{b_\kappa^* c_i}{t_i^{\max} - t_\kappa^{\mathrm{up}^*}},\ \forall i \in \mathcal{I}\ \forall \kappa \in \mathcal{K}_i. \tag{5.13}$$

Similarly, the allocated computation resource to the ith device is:

$$f_i^* = \frac{b_i^* c_i}{t_i^{\max}},\ \forall i \in \mathcal{I}. \tag{5.14}$$

Using (5.13) and (5.14) we form the optimal allocated computation resource vector $\mathbf{f}^*$ in the same manner as $\mathbf{f}$. Recall that problem $\mathscr{P}_2$ does not consider constraints (5.9c), therefore, it may happen that the allocated task sizes are such that $\sum_{i\in\mathcal{I}} \frac{b_{i0}^* c_i}{t_i^{\max} - \frac{b_{i0}^*}{R_{i0}^*}} > f_0^{\max}$, consequently, constraint (5.9c) is violated. In that case, we reallocate the task sizes $\mathbf{b}^*$ subject to constraint (5.9c) by fixing the transmit powers to $\mathbf{p}^*$ as follows:

$$\mathscr{P}_3 : \min_{\mathbf{b}} \sum_{i\in\mathcal{I}} \psi_i(\mathbf{p}_i^*, \mathbf{b}_i) \tag{5.15a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} \frac{b_{i0} R_{i0}^* c_i}{R_{i0}^* t_i^{\max} - b_{i0}} \le f_0^{\max} \tag{5.15b}$$

$$b_i + \sum_{\kappa\in\mathcal{K}_i} b_\kappa = d_i,\ \forall i \in \mathcal{I} \tag{5.15c}$$

$$b_\kappa - \alpha t_i^{\max} R_\kappa^* \le 0, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{5.15d}$$

$$0 \le b_i, b_\kappa, \forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i \tag{5.15e}$$

We denote the new task size allocations obtained by solving problem $\mathscr{P}_3$ as $\mathbf{b}^+$. Then, we update the task uploading times as $t_\kappa^{\mathrm{up}^+} = \frac{b_\kappa^+}{R_\kappa^*}, \forall i \in \mathcal{I},\ \forall \kappa \in \mathcal{K}_i$. Finally, we calculate the new allocated computation resources $\mathbf{f}^+$ as in (5.13) and (5.14) by using $\mathbf{b}^+$ and $\mathbf{p}^*$.

5.3.4 Summary of the Proposed Method

In Algorithm 5.1 we summarize the proposed convex-programming-based sub-optimal method to solve problem $\mathscr{P}_1$. The algorithm runs in a central location, for example at a BS, and the obtained solution is relayed to the active devices.

Algorithm 5.1 Convex Programming Method

```
1: Solve Problem $\mathscr{P}_2$ to obtain initial $\mathbf{b}^*$ and $\mathbf{p}^*$
2: Calculate $t_\kappa^{\text{up}^*} = \frac{b_\kappa^*}{R_\kappa^*}$ $\forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i$
3: Calculate $\mathbf{f}^*$ based on $\mathbf{b}^*$ and $\mathbf{p}^*$ by using (5.13) and (5.14)
4: if $\|\mathbf{f}_0^*\|_1 > f_0^{\max}$ then
5:   Re-distribute tasks to obtain $\mathbf{b}^e$ by solving Problem $\mathscr{P}_3$
6:   Obtain the new task portions as $\mathbf{b}^+ = \mathbf{b}^* + \mathbf{b}^e$
7:   Calculate $t_\kappa^{\text{up}^+} = \frac{b_\kappa^+}{R_\kappa^*}$ $\forall i \in \mathcal{I}, \forall \kappa \in \mathcal{K}_i$
8:   Calculate $\mathbf{f}^+$ based on $\mathbf{b}^+$ and $\mathbf{p}^*$ by using (5.13) and (5.14)
9: end
```

5.4 Heuristic Task Offloading Method

In this section, we propose a heuristic algorithm to solve problem $\mathscr{P}_1$. The main goal is to provide an accurate, low-complexity method without computing gradients and Hessian matrices in the solution process. To develop the heuristic method we will follow a sequential approach as in Section 5.3. We start by allocating the transmit powers and the task sizes under data rate constraints, as we did in problem $\mathscr{P}_2$. Then, we take the maximum computation capability of the ES into account as in problem $\mathscr{P}_3$ to prevent the violation of constraint (5.9c).

5.4.1 Initial Resource Allocation

In an ideal scenario given in Appendix E, the total energy consumption is minimized when the task sizes are equally partitioned among the offloading destinations. Based on this, we first initialize the task sizes by partitioning them among the ith active device and its offloading destinations in $\mathcal{K}_i$ as follows:

$$\mathbf{b}_i^* = d_i(|\mathcal{K}_i| + 1)^{-1}\mathbf{1}, \ \forall i \in \mathcal{I} \tag{5.16}$$

where $\mathbf{1}$ is the column vector of all-ones. If the task uploading time between each active device and its offloading destination is instantaneous, then the total energy consumption based on the initial task splitting strategy in (5.16) clearly attains the lower bound in Appendix E. However, in a realistic scenario, due to wireless channel conditions as well as the transmit power constraints, the achievable data rates are limited. Hence, our next step is to reduce the task uploading time as much as possible to approach the lower bound. To this end, we allocate the transmit powers to compensate the channel conditions among the ith active device and its offloading destinations:

$$\mathbf{p}_i^* = \frac{\mathbf{h}_i P^{\max}}{\|\mathbf{h}_i\|_1}, \ \forall i \in \mathcal{I} \tag{5.17}$$

where $\mathbf{h}_i = \|\mathbf{g}_i\|_1 \mathbf{1} - \mathbf{g}_i$ is the vector that contains the channel gains of the ith device that are subtracted from its ℓ_1-norm and the channel gain vector is defined as $\mathbf{g}_i = \left[G_{i0}\ G_{i1}\ \ldots\ G_{iK}\right]^\top$, $i \in \mathcal{I}$. By initializing the transmit powers as in (5.17), each active device allocates more transmit power to its offloading destinations with relatively low channel gain to increase data rates, and in turn, reduce task uploading time. However, for an ith device and its κth offloading destination, b_κ^* and P_κ^* might exceed the task processing deadline, i.e., $t_\kappa^{\mathrm{up}^*} > t_i^{\max}$, which violates constraint (5.9f).

Since $\mathbf{p}^*$ is initialized to compensate physical channel conditions, we update the initial task partitioning $\mathbf{b}^*$ to have a feasible $t_\kappa^{\mathrm{up}^*}$ similar to constraint (5.12d). Specifically, for any device i and its κth offloading destination for which $t_\kappa^{\mathrm{up}^*} > t_i^{\max}$, we set $t_\kappa^{\mathrm{up}^*} = \alpha t_i^{\max}$, which is the maximum time limit for uploading the task sizes as in constraint (5.12d). Then, we re-calculate the corresponding task size as $b_\kappa^+ = R_\kappa^* \alpha t_i^{\max}$, which is smaller than b_κ^*. Therefore, there is an excess task size at the κth device $b_\kappa^{\mathrm{e}} = b_\kappa^* - b_\kappa^+$, which can not be uploaded, and must be re-allocated for processing among other offloading destinations. As in (5.16), we equally re-allocate this excess task size among the ith active device and its remaining offloading destinations whose indices are in the set $\mathcal{K}_i - \{\kappa\}$. However, it is likely that increasing the task sizes at those destinations under fixed transmit power $\mathbf{p}^*$ may increase the task uploading time enough to violate constraint (5.12d). Therefore, we continue re-allocating the task sizes $\mathbf{b}_i^*$ $\forall i \in \mathcal{I}$ until constraint (5.12d) is satisfied by all active devices. Once the feasible task partitioning is obtained with respect to $\mathbf{p}^*$, we can then calculate the computation resources $\mathbf{f}^*$ for the corresponding new task sizes as in (5.13) and (5.14). The overall heuristic resource initialization strategy is given in Algorithm 5.2.

Algorithm 5.2 Initial Resource Allocation

```
1: Calculate $\mathbf{b}^*$ and $\mathbf{p}^*$ based on (5.16) and (5.17)
2: for $i \in \mathcal{I}$ do
3:     Assign a temporary set $\overline{\mathcal{K}}_i = \mathcal{K}_i$
4:     Calculate $t_\kappa^{\mathrm{up}^*} = \frac{b_\kappa^*}{R_\kappa^*}, \forall \kappa \in \overline{\mathcal{K}}_i$
5:     if $t_\kappa^{\mathrm{up}^*} > t_i^{\max}, \forall \kappa \in \overline{\mathcal{K}}_i$ then
6:       Set $t_\kappa^{\mathrm{up}^*} = \alpha t_i^{\max}$ and update $b_\kappa^*$ as $b_\kappa^+ = R_\kappa^* \alpha t_i^{\max}$
7:       Calculate the excess task size $b_\kappa^{\mathrm{e}} = b_\kappa^* - b_\kappa^+$
8:       Update $\overline{\mathcal{K}}_i \leftarrow \overline{\mathcal{K}}_i - \{\kappa\}$ to partition $b_\kappa^{\mathrm{e}}$
9:       $b_i^+ = b_i^* + \frac{b_\kappa^{\mathrm{e}}}{|\overline{\mathcal{K}}_i|+1}$
10:      $b_\kappa^+ = b_\kappa^* + \frac{b_\kappa^{\mathrm{e}}}{|\overline{\mathcal{K}}_i|+1} \; \forall \kappa \in \overline{\mathcal{K}}_i$
11:      Update $\mathbf{b}_i^* \leftarrow \mathbf{b}_i^+$ then calculate $t_\kappa^{\mathrm{up}^*}, \forall \kappa \in \overline{\mathcal{K}}_i$
12:      Go to line 5
13:    end
14: end
15: Calculate $\mathbf{f}^*$ by using $\mathbf{b}^*$ and $\mathbf{p}^*$ based on (5.13) and (5.14)
```

5.4.2 Re-allocating the Excess Resources

After running Algorithm 5.2, the offloaded tasks at the ES may not be processed if the total required computation resources exceed the maximum limit $\|\mathbf{f}_0^*\|_1 > f_0^{\max}$. In that case, the surplus task sizes at the ES are re-allocated to the other offloading destinations. To determine what should be removed from the ES, we introduce the vector $\bar{\mathbf{r}}$ whose elements are inversely proportional to the amount of the computation resources that should be allocated to the ES for the corresponding task sizes $b_{i0}^*, \forall i \in \mathcal{I}$:

$$\bar{\mathbf{r}} = \frac{\mathbf{r}^*}{\|\mathbf{r}^*\|_1} \tag{5.18}$$

where $\mathbf{r}^* = \|\mathbf{f}_0^*\|_1 \mathbf{1} - \mathbf{f}_0^*$. By using $\bar{\mathbf{r}}$, we can calculate the new allocated computation resources at the ES by removing the excess resources as follows:

$$\mathbf{f}_0^+ = \mathbf{f}_0^* - f^{\mathrm{e}} \bar{\mathbf{r}} \tag{5.19}$$

where $f^{\mathrm{e}} = \|\mathbf{f}_0^*\|_1 - f_0^{\max}$, with $\mathbf{f}_0^* = [f_{10}^* \; f_{20}^* \; \ldots \; f_{I0}^*]^\top$, contains the excess computation resources at the ES. Note that $\mathbf{f}_0^+$ in (5.19) not only satisfies constraint (5.9c), but also each element $f_{i0}^+, i \in \mathcal{I}$ of $\mathbf{f}_0^+$ is still proportional to the amount of task sizes of the active devices, which is important in terms of reducing the energy consumption as shown in Appendix E. Finally, we obtain the task sizes that should be left at the ES based on $\mathbf{f}_0^+$. Specifically, given the data rate R_{i0}^*, we calculate the new task size to be offloaded to the ES from the ith device by using (5.5) and (5.6) as follows:

$$b_{i0}^+ = \frac{R_{i0}^* f_{i0}^+ t_i^{\max}}{f_{i0}^+ + c_i R_{i0}^*}, \ \forall i \in \mathcal{I} \tag{5.20}$$

Nevertheless, if the amount of subtracted computation resources surpasses the previously allocated resources by the ES, then some of the elements of $\mathbf{f}_0^+$ in (5.19) might become negative, which by definition is not possible. Therefore, in the case of negative f_{i0}^+ for any $i \in \mathcal{I}$, constraint (5.9c) remains violated since the excess resource f^{e} could not be properly removed from $\mathbf{f}_0^*$.

In Algorithm 5.3, we present an iterative strategy to address the above issue and re-allocate the resources in by properly removing the excess computation resource f^{e} until it becomes zero. Specifically, in line 2, we first calculate the excess computation resources to obtain $\mathbf{f}_0^+$ (5.19), which is the new allocated computation resources by the ES to not violate constraint (5.9c). If $f_{i0}^+ \geq 0, \forall i \in \mathcal{I}$, we can directly determine the corresponding task sizes that can be uploaded to the ES b_{i0}^+, $\forall i \in \mathcal{I}$ (5.20) based on $\mathbf{f}_0^+$ as given in line 15. Then, the excess task size b_i^{e}, which cannot be processed at the ES, is re-allocated among all the offloading destinations of the ith device except the ES $\forall i \in \mathcal{I}$ as given by lines 16 and 17. If the new task uploading time exceeds the deadline, i.e., $\frac{b_\kappa^+}{R_\kappa^*} > t_i^{\max}$, as controlled in line 18, we run lines 5-13 in Algorithm 5.2 without (w/o) line 15 as it is now redundant. Hence, we obtain the final task splitting decision of the ith device by $\boldsymbol{b}_i^+ = [b_i^+ \ b_{i0}^+ \ b_{i1}^+ \ ... \ b_{iK}^+]^\top$ to form $\mathbf{b}^+$. However, if $f_{i0}^+ < 0$ in $\mathbf{f}_0^+$ for any $i \in \mathcal{I}$, we first take its absolute value and add it to the remaining excess resource f^{e} to be removed in the next iterations as it is not properly removed from the resources that the ES initially allocated while obtaining $\mathbf{f}_0^+$ in line 2. This emphasizes that the active devices whose corresponding computation resources allocated by the ES becomes negative cannot utilize the ES for task offloading anymore. Hence, we set $f_{i0}^+ = 0$ if $f_{i0}^+ < 0, i \in \mathcal{I}$ as given in 6. Since those devices are not utilizing the ES, their initial allocated transmit power, i.e., P_{i0}^*, for the ES also becomes redundant. Therefore, we should re-allocate the transmit powers of these active devices among their other offloading destinations as given in line 8 by setting the 1st element of $\mathbf{h}_i$ to zero and using (5.17). Then, in line 10, we replace $\mathbf{f}_0^*$ with $\mathbf{f}_0^+$ to be used in the next iteration in the case of $f^{\mathrm{e}} \neq 0$. Thus, Algorithm 5.3 recursively continues in this manner until $f^{\mathrm{e}} = 0$ and obtains $\mathbf{f}_0^{+}$, which are the new computation resources allocated by the ES without violating constraint (5.9c). Finally, it outputs the new task splitting decisions $\mathbf{b}^{+}$, hence, we can calculate the corresponding computation resources $\mathbf{f}^{+}$ as in (5.13) and (5.14) by using $\mathbf{b}^{+}$ and $\mathbf{p}^{*}$.

Algorithm 5.3 Re-allocating Resources

```
1:  if $\|\mathbf{f}_0^*\|_1 > f_0^{\max}$ then
2:    Calculate $f^{\mathrm{e}} = \|\mathbf{f}_0^*\|_1 - f_0^{\max}$ and $\bar{\mathbf{r}}$ (5.18) to obtain $\mathbf{f}_0^{+}$ (5.19)
3:    Reset $f^{\mathrm{e}} = 0$
4:      if $f_{i0}^{+} < 0, i \in \mathcal{I}$ then
5:        Update $f^{\mathrm{e}} \leftarrow f^{\mathrm{e}} + |f_{i0}^{+}|$
6:        Set $f_{i0}^{+} = 0$
7:        Set the $i$th element of $\mathbf{r}^*$ to 0
8:        Set the 1st element of $\mathbf{h}_i$ to 0 and re-calculate $\mathbf{p}_i^*$ (5.17)
9:      end
10:     Replace the previous allocation $\mathbf{f}_0^* \leftarrow \mathbf{f}_0^{+}$
11:     if $f^{\mathrm{e}} \neq 0$ then
12:       Re-calculate $\bar{\mathbf{r}}$ (5.18) to obtain new $\mathbf{f}_0^{+}$ based on $\mathbf{f}_0^*$ (5.19)
13:       Go to line 3
14:     end
15:   Based on $\mathbf{f}_0^{+}$, obtain $b_{i0}^{+}$, $\forall i \in \mathcal{I}$ (5.20)
16:   Calculate the excess task size $b_i^{\mathrm{e}} = b_{i0}^* - b_{i0}^{+}, \forall i \in \mathcal{I}$
17:   Re-allocate $b_i^{\mathrm{e}}$ among $\overline{\mathcal{K}}_i = \mathcal{K}_i - \{0\}, \forall i \in \mathcal{I}$ for $\mathbf{b}^{+}$
      $b_i^{+} = b_i^* + \frac{b_i^{\mathrm{e}}}{|\overline{\mathcal{K}}_i|+1}, \forall i \in \mathcal{I}$
      $b_\kappa^{+} = b_\kappa^* + \frac{b_i^{\mathrm{e}}}{|\overline{\mathcal{K}}_i|+1},\ \kappa \in \overline{\mathcal{K}}_i, \forall i \in \mathcal{I}$
18:     if $\frac{b_\kappa^{+}}{R_\kappa^*} > t_i^{\max},\ \kappa \in \overline{\mathcal{K}}_i, \forall i \in \mathcal{I}$ then
19:       Update $\mathbf{b}_i^* \leftarrow \mathbf{b}_i^{+}$
20:       Run lines 4-13 in Algorithm 5.2 w/o line 15
21:       Update $\mathbf{b}_i^{+} \leftarrow \mathbf{b}_i^*$
22:     end
23:   Calculate $\mathbf{f}^{+}$ by using $\mathbf{b}^{+}$ and $\mathbf{p}^*$ based on (5.13) and (5.14)
24: end
```

5.4.3 Complexity Analysis

We sequentially calculate the required operations, which are only real multiplications and real additions, throughout the proposed heuristic method. For simplicity, we assume that the maximum number of task offloading destinations for any ith device is $|\mathcal{K}_i| = K$. First, we consider Algorithm 5.2, which takes $I(K+1)$ and $I(5K+1)$ operations for (5.16) and (5.17), respectively. Note that the data rates R^*_κ, $\forall\kappa \in \mathcal{K}_i$ can be obtained in advance, which requires $4K$ operations $\forall i \in \mathcal{I}$, hence, line 4 requires only K operations. Assuming the *if condition* between the lines 5 and 13 is repeated $\forall\kappa \in \mathcal{K}_i$, we have $2K^2 + 5K$ operations. Furthermore, line 15 takes $3K + 2$ operations and repeating all these operations I times yields the time complexity of Algorithm 5.2 as $\mathcal{O}(IK^2 + IK)$.

Second, we consider Algorithm 5.3 and begin with line 2 by calculating f^{e}, $\bar{\mathbf{r}}$ (5.18), and $\mathbf{f}_0^+$ (5.19), which take $I + 1$, $3I$ and $2I$ operations, respectively. Then, assuming lines 4-14 are repeated for every active device, we have $5I^2 + 5IK + 2I$ operations. The rest of the required operations from line 15 to line 17 are $10I + 3K$ and from line 18 to line 20 are $2K^2 + 5K$. Hence, the time complexity of Algorithm 5.3 is $\mathcal{O}(I^2 + K^2 + IK)$.

Finally, the overall time complexity of the proposed heuristic method is $\mathcal{O}(IK^2+I^2+K^2)$.

5.5 Simulation Results

In this section, we compare both the convex-programming method and the heuristic method to the lower-bound on the total energy consumption given in Appendix E through Monte-Carlo simulations under various conditions such as the number of active and fog devices, task sizes, time constraints, and computation resources. For the network layout, we uniformly distribute the active device (AD)s within a 500×500 m^2 area and the fog device (FD)s are placed on a disk with a radius of 15m centered at an active device. We consider independent Rayleigh fading channels between the devices and we use the following path loss models for the cellular wireless links between ADs and the BS, $\mathrm{PL}_{\mathrm{cell}} = 128.1 + 37.6\log_{10}(d)$, and for the D2D links between ADs and FDs, $\mathrm{PL}_{\mathrm{D2D}} = 148 + 40\log_{10}(d)$, where d is the distance in km [139]. At each simulation run, the task size d_i, $\forall i \in \mathcal{I}$ is chosen from a uniform distribution $\mathcal{U}(2\times10^4,\ 4\times10^5)$ and the maximum computation capability of the ES $f_0^{\max}$

is selected based on:

$$f_0^{\max} = \mathbb{E}\left[\sum_{i\in\mathcal{I}} f_i^{\text{opt}}\right] \times \eta \tag{5.21}$$

where f_i^{opt} is obtained from Appendix E and $\eta \in [0\ 1]$ is a scaling term of $f_0^{\max}$. The reason behind choosing $f_0^{\max}$ as such instead of fixing it a pre-defined value is to make sure that condition (5.9c) in the original problem is effective and challenging its solution. Therefore, by changing $f_0^{\max}$ in a controlled manner as in (5.21), we ensure that problem $\mathscr{P}_3$ and Algorithm 5.3 must be employed in the simulations for both proposed methods. To this end, we choose $\eta = 0.8$ to reduce $f_0^{\max}$ for 20% of the total required computation resources on average in the optimum scenario given in Appendix E. The rest of the system parameters are given in Table 5.1 unless otherwise is specified.

Table 5.1. System parameters

Parameter Description	**Symbol**	**Value**
Network size	-	500×500 m^2
Max. radius of a D2D link	-	25 m
Number of active devices	I	$\{2\ 3\ \ldots 12\}$
Number of fog devices	K	$\{0\ 1 \ldots 5\}$
Task size	d_i	$[2\times10^4,\ 4\times10^5]$ bits
CPU cycles to process 1-bit data	c_i	1500 cycles/bit
Effective capacitance constant	μ	10^{-24} Ws^3
Scaling term for $f_0^{\max}$	η	0.8
Limiting term for task uploading time	α	0.85
Max. computation capability at the ES	$f_0^{\max}$	$[0.2,\ 1.5]$ GHz
Max. transmit power	$P^{\max}$	$[0,\ 200]$ mW
Task processing deadline	$t_i^{\max}$	$[0.4,\ 1]$ s
Noise level	N_0	-114 dBm
Channel bandwidth	W	10 MHz

In Fig. 5.2, we investigate the total energy consumption as a function of the number of ADs, where $I \in \{2\ 4\ \ldots\ 12\}$ and the number of FDs follows $K \in \{0\ 1\ 3\ 5\}$. As the number of ADs changes, we set the computation capability of the ES to $f_0^{\max} \in \{0.25\ 0.51\ 0.77\ 1.03\ 1.27\ 1.53\}$ in GHz, which is calculated based on (5.21). We compare both proposed methods to their corresponding lower bound on the total energy consumption obtained in Appendix E. It can be shown that by incorporating D2D communications in the task offloading process, computing a similar task on average takes less energy with the help of each additional fog device compared to utilizing only the ES.

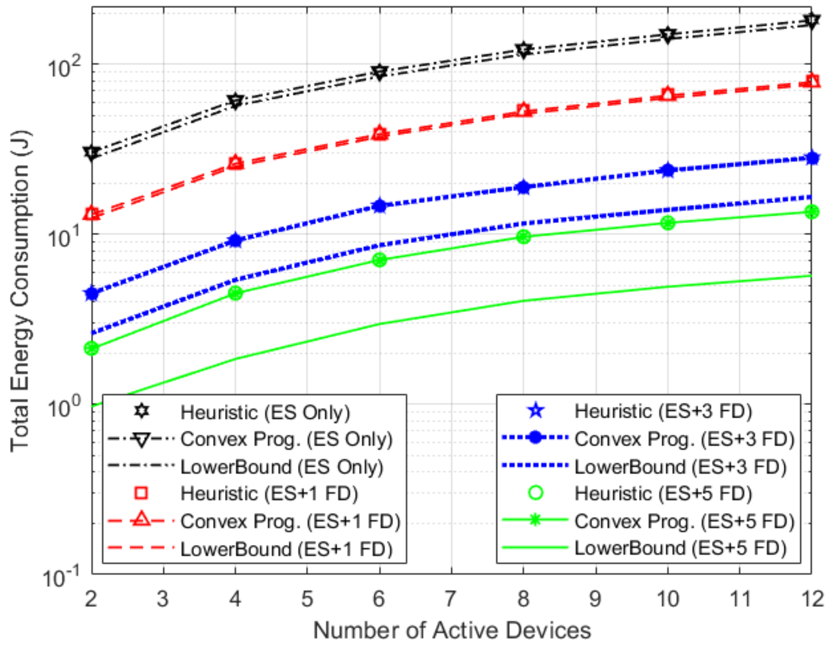
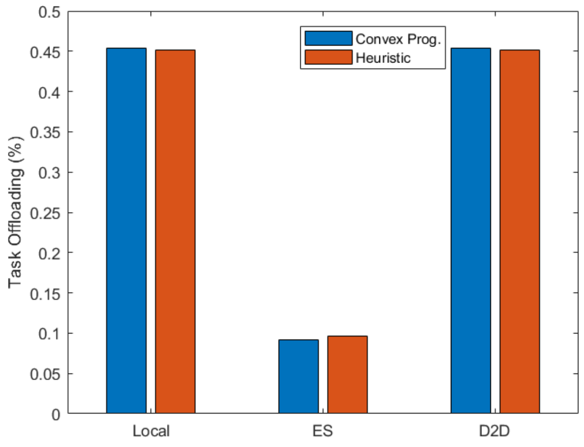
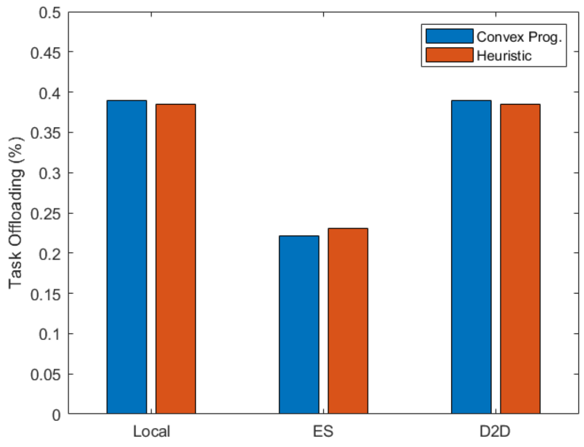


Fig. 5.2. Total energy consumption versus different number of ADs while $t_i^{\max} = 1\ \forall i \in \mathcal{I}$ and $P^{\max} = 100$.

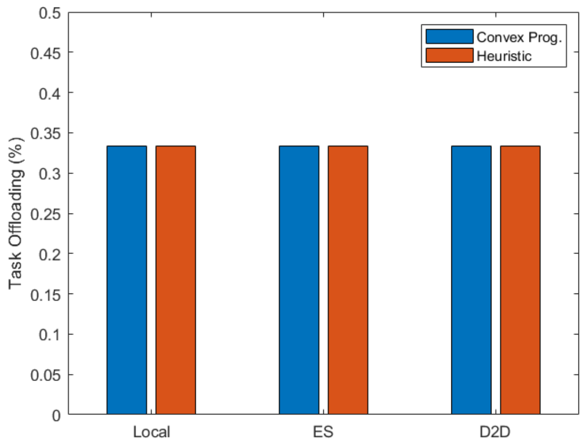
We note that the performance of both proposed methods starts deviating from the lower bound as the number of FDs increases in a considered scenario. This performance gap happens due to two reasons. First, under the limited computation capability of the ES, task partitioning cannot be as equal among the offloading destinations as in the lower bound in Appendix *E*. Second, under the finite data rates and transmit powers, more computation resources must be allocated as opposed to the lower bound in Appendix *E* to compensate the task uploading time in order to meet constraints (5.9e) and (5.9f). Therefore, to validate these explanations and show that the proposed methods achieve near-optimal performance, we individually investigate them.



(a) ES capacity: $f_0^{\max} = 0.2$ GHz



(b) ES capacity: $f_0^{\max} = 0.4$ GHz



(c) ES capacity: $f_0^{\max} = 0.8$ GHz

Fig. 5.3. Utilization of the offloading destinations in percentage under different computation capability of the ES.

To validate the first explanation, in Fig. 5.3 we observe the effect of $f_0^{\max}$ to the total energy consumption by plotting the utilization of the offloading destinations in percentage for a given task under various values of $f_0^{\max}$. We set $f_0^{\max} \in \{.2\ .4\ .8\}$GHz, which are predefined values and not calculated based on (5.21), while we choose $I = 5$, $|\mathcal{K}_i| \in \{0, 1\}$, i.e., the ES and a single fog device, and $t_i^{\max} = 1\ \forall i \in \mathcal{I}$. If both the convex-programming method and the heuristic method distribute the tasks as equal as possible subject to the constraints, then we show that the proposed methods can approach to the optimal solution given as the lower-bound. In Fig. 5.3a, when $f_0^{\max}$ is 0.2 GHz, the utilization of the ES by the devices is limited, but the tasks are distributed equally for local and fog computing. However the performance deviations in terms of the total energy consumption of the convex-programming and the heuristic method from the lower bound are 43% and 65%, respectively. When the limitation of $f_0^{\max}$ is gradually removed by choosing $f_0^{\max} = 0.4$GHz as in Fig. 5.3b, the participation of the ES for task offloading increases, which reduces the performance gap to 1% for the convex-programming and 19% for the heuristic method. Finally, when there is no resource limitation at the ES (relatively to task sizes and the number of devices in the network), both proposed methods achieve near-optimal solution, where the performance deviations are only 0.13% and 1%, respectively.

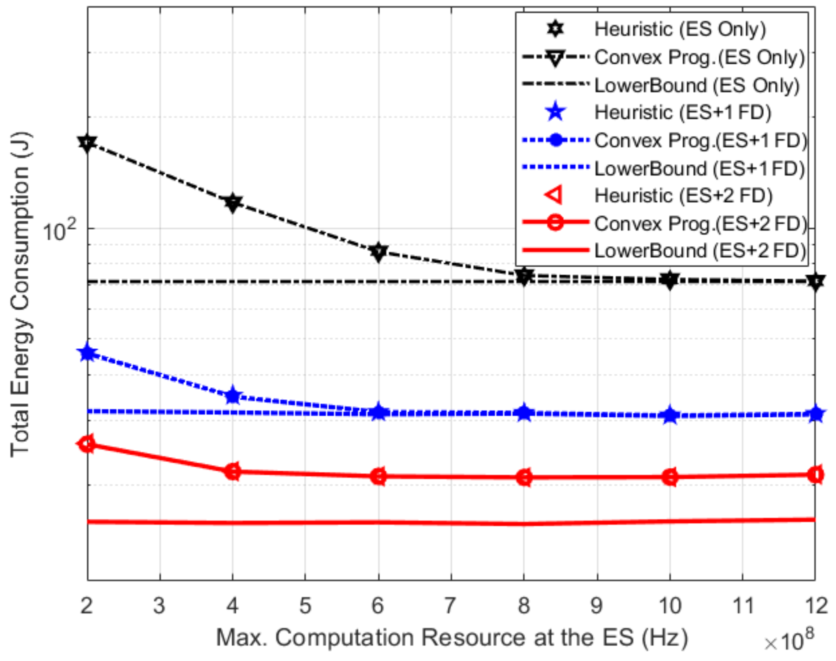


Fig. 5.4. Total energy consumption versus maximum computation capability of the ES, i.e., $f_0^{\max}$.

To validate the second explanation, we study the effect of $f_0^{\max}$ in Fig. 5.4 while changing the number of FDs, hence, the limitation of data rates and transmit powers on the proposed methods can be investigated. We assume that the number of ADs is 6 and we increase $f_0^{\max}$ as in the figure. When the computation capability of the ES is $f_0^{\max} = 0.2$GHz, the performance gap for both proposed methods in the case of utilizing only the ES (black line) is significantly higher compared to incorporating a single FD (blue line) as the limitation of $f_0^{\max}$ is being compensated with the additional computation resources at the FD. However, for values of $f_0^{\max}$ higher than $= 0.6$GHz, the effect of $f_0^{\max}$ almost disappears and both proposed methods approach to near-optimal solution for these two scenarios. Nonetheless, with the addition of one more FD (red line), a constant offset appears between the performance of the proposed methods and the lower bound, which is a direct indication of requiring more computation resources by the devices to compensate the task uploading time under the limited data rates and transmit powers. Therefore, we numerically demonstrated that both proposed methods achieve a near-optimal solution and the performance gaps occur due to the physical limitations in the considered scenarios and not their implementation.

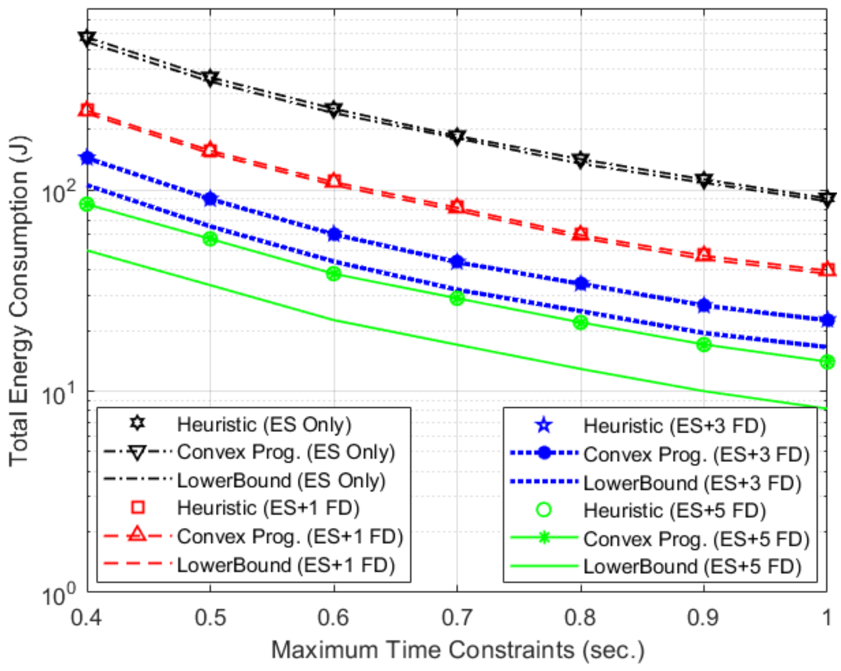


Fig. 5.5. Total energy consumption versus different maximum time constraints, i.e., $t_i^{\max} \in [0.4,\ 1]\ \forall i \in \mathcal{I}$, while $I = 6$ and $P^{\max} = 100$.

In Fig. 5.5, we plot the change in the total energy consumption with respect to the task processing deadline. Similar to the result in Fig. 5.4, as $t_i^{\max}$ $\forall i \in \mathcal{I}$ decreases, more computation resources must be allocated by the devices to process the offloaded tasks within the given deadline, and consequently, the total energy consumption increases. However, it can be also verified that both the convex-programming method and the heuristic method in the case of utilizing only the ES or a single FD achieve a near-optimal performance regardless of the task processing deadline, which again demonstrates that the performance gaps are the direct indications of the physical limitations imposed by the constraints.

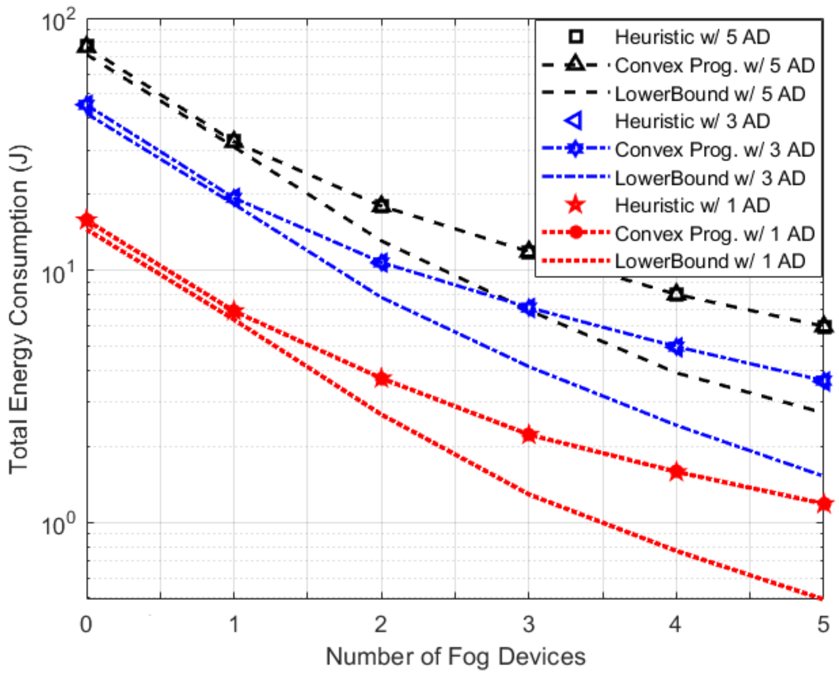


Fig. 5.6. Total energy consumption with respect to different number of ADs and IDs while $t_i^{\max} = 1$ and $P^{\max} = 100$.

In Fig. 5.6, we study the total energy consumption with respect to the change in the number of fog devices, where $K \in \{0\ 1\ \ldots\ 5\}$ and $I \in \{1\ 3\ 5\}$. Similar to Fig. 5.2, the computation capability of the ES is determined based on (5.21) as the number of devices in the network change in each simulation run. It is shown that by increasing the number of fog devices in the task offloading process, the total energy consumption can be significantly decreased. Specifically, it takes ten times less energy to compute a similar task on average with the help of 5 FDs via D2D links instead of utilizing only the ES. Nevertheless, the total energy consumption is higher in the case of 1 AD compared to the case of 5 ADs and 5

FDs. This shows that taking a computation-intensive task off a single device and effectively partitioning it across all the available computation resources can significantly reduce the overall energy consumption even though more devices are involved in the task offloading process.

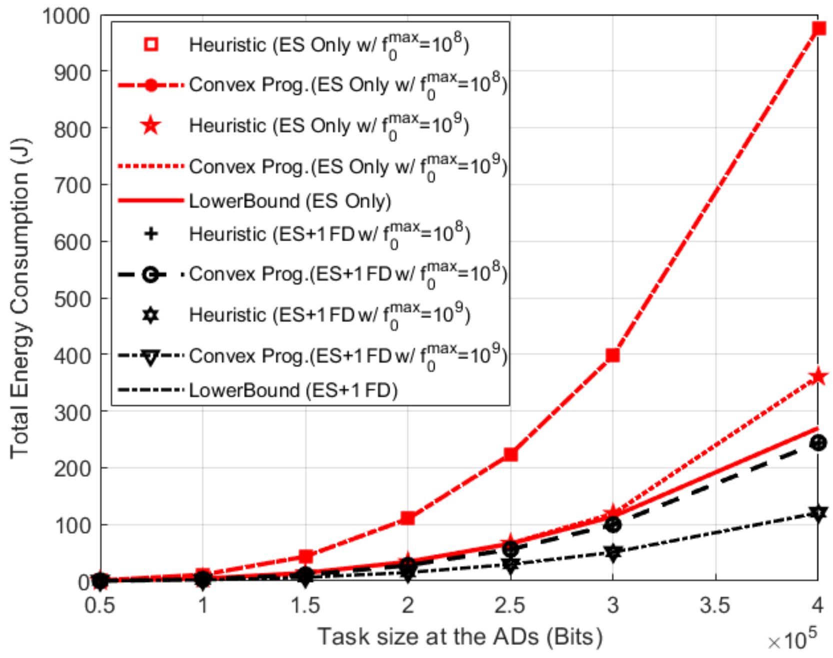


Fig. 5.7. Total energy consumption versus task size under different number of offloading devices and computation capability of the ES, i.e., $|\mathcal{K}_i| \in \{0, 1\}$ and $f_0^{\max} \in \{10^8, 10^9\}$, respectively.

In Fig. 5.7, we study the change in the total energy consumption with respect to the different task sizes. The number of ADs is set to 5 while there is only a single FD. In addition, we set the available computation resource of the ES to pre-defined values as $f_0^{\max} \in \{10^8, 10^9\}$. It is shown that when the task size is 400kbits, there is a drastic increase in the total energy consumption if D2D communication is not used for task offloading, especially when $f_0^{\max} = 10^8$Hz. On the contrary, by employing only a single FD in the task offloading process, the total energy consumption decreases by almost 5 times. More importantly, the change in the total energy consumption with $f_0^{\max} = 10^8$Hz and $f_0^{\max} = 10^9$Hz is relatively small when compared to utilizing only the ES. Hence, we can conclude that by incorporating D2D communication and exploiting nearby computation resources, we can alleviate the dependence on the ES, which is crucial when the traffic density is high and the available resources are scarce in the network.

Finally, we compare the average run-time of both proposed methods implemented in MATLAB and executed on an Intel i7-3770 computer with 16GB RAM. For the first comparison, we assume only one AD and three FDs while changing the task processing deadline from 0.4 to 1 second as presented in Table 5.2. Even though the proposed convex programming method slightly outperforms the heuristic one in terms of energy efficiency as demonstrated earlier, its average run-time is significantly higher. For the second comparison as shown in Table 5.3, we include three more ADs to the network while keeping the number of FDs the same. For $t^{\max} = 0.4$s, the average run time of our heuristic method is almost 2x10^4 times less than the convex programming method. This indicates that the proposed heuristic method becomes more beneficial especially for scenarios with increased network capacity and data traffic.

Table 5.2. Average run-time comparison when $I = 1$ and $J = 3$

Simulation setup	**Convex Prog.**	**Heuristic**
$t^{\max} = 0.4$ s	0.145 s	97.1 μs
$t^{\max} = 0.6$ s	0.135 s	95.4 μs
$t^{\max} = 0.8$ s	0.131 s	95.1 μs
$t^{\max} = 1$s	0.129 s	94.5 μs

Table 5.3. Average run-time comparison when $I = 5$ and $J = 3$

Simulation setup	**Convex Prog.**	**Heuristic**
$t^{\max} = 0.4$ s	3.015 s	0.156 ms
$t^{\max} = 0.6$ s	2.602 s	0.156 ms
$t^{\max} = 0.8$ s	2.331 s	0.154 ms
$t^{\max} = 1$s	2.206 s	0.154 ms

5.6 Conclusion

In this chapter, we addressed the energy-efficient resource allocation problem in a generalized multi-device D2D-aided fog computing scenario, wherein a central ES and proximate fog devices are utilized for task offloading. Since the formulated problem is intractable, we first analyzed it to develop a sub-optimal convex-programming based method, in which the computation resources, task sizes and the transmit powers are allocated to reduce the total energy consumption in the network. Then, based on this first method, we developed a heuristic task offloading method, which does not require computing gradients and Hessian matrices during the solution process. We showed in detail the computational complexity of this method in terms of key system parameters, including the number of mobile devices and task offloading destinations. Finally, we developed a lower bound on the total energy consumption as a performance benchmark for both the convex-programming and the heuristic methods. The computer simulations demonstrated that the proposed methods significantly reduce the total energy consumption compared to processing tasks only locally while attaining near-optimal solutions in comparison to the derived lower bound.

Chapter 6

Conclusion

In this chapter, we first present a detailed summary of the thesis and then discuss some possible research directions that can emerge as future works.

6.1 Summary

D2D communication is a promising technology to be part of 5G and B5G networks by allowing proximate devices to opportunistically establish P2P connections. Naturally, this type of communication technology significantly reduces end-to-end delays, traffic loads and total overheads at base stations while providing high data rates and ultra-low latency that are extremely beneficial for many use cases such as mission critical operations for public safety. In the events of cellular network outage, e.g., natural disasters, D2D communication, especially the out-of-coverage network setting, in which the proximate devices can communicate to each other without coordination of a centralized node, can offer fast localization and intervention. In addition to public safety applications, D2D communication can also be integrated in numerous non-public safety applications such as smart home technologies, augmented reality and online gaming, local advertising, and task offloading. However, implementing realistic, reliable, and energy-efficient D2D communication poses several difficulties of its own. To this end, in this thesis, we addressed two of the most fundamental problems on the realization of D2D communications, namely, synchronization and resource allocation.

In Chapter 3, we addressed a clock synchronization problem in distributed D2D networks by considering the following challenges. First, the duplexing scheme, i.e., full-duplex or half-duplex, can drastically change the synchronization time as full-duplex allows simultaneous signal transmission and reception. However, full-duplex technology is not practical yet, especially for commercial hand-held mobile devices. Second, the physical phenomenon called clock skew arbitrarily leads device clocks to lag or advance in time with respect to each other, hence, become asynchronous even though the clocks are initially synchronized. Third, multipath channels introduce propagation delays, which lead to additional synchronization error at every signal exchange. Thus, a proactive action, i.e., timing-advance synchronization, is required to mitigate the effect of propagation delays. In addition, by considering the nature of D2D networks, where devices arbitrarily join or leave the network and can distort the ongoing synchronization process, a synchronization algorithm should be flexible and robust to accommodate dynamic device numbers. Finally, synchronized devices should be able to terminate the synchronization process, ideally at the same time, to initiate data communication while maintaining the achieved synchronization state. In this regard, to address all these aforementioned issues, we developed a half-duplex timing-advance synchronization algorithm wherein each device becomes a transmitter and receiver in its exchange of synchronization pulses at each clock period. Then, in light of this algorithm, we proposed a novel fully-distributed pulse-based synchronization protocol for half-duplex D2D communications. The proposed protocol achieves distributed devices to become aware of the global synchronization status to be able to initiate data communication. In addition, synchronized devices, which might not need to participate in data communication, can become idle to conserve energy while maintaining the synchronization with respect to the rest of the network. The simulation results demonstrated that the proposed protocol not only achieves fast synchronization in comparison to a benchmark from the literature but also compensates for possible perturbations and performs well over a wide range of conditions such as multi-path frequency selective channels, clock skew, dynamic number of devices, and different network topology.

After resolving the synchronization problem by implementing the full-fledged synchronization protocol, in Chapter 4, we considered D2D communications as an enabling technology for a fog computing framework to develop an energy-efficient task offloading scheme. In comparison to Cloud computing and MEC, fog computing can significantly reduce end-to-end delays and total overheads in the network, and in turn, provide energy efficiency when processing computation-intensive tasks that require ultra-low latency. However, high operating temperatures while processing such tasks may lead to random CPU throttling on mobile device due to DTM, which controls the on-chip temperature by lowering the voltage and frequency of CPUs. Consequently, task completion time become random, which negatively impacts the task offloading performance. To this end, we considered the optimal resource allocation problem with respect to task partitioning, computation resources and transmit power, to minimize the expected total energy consumption under probabilistic constraints on the processing time. However, the formulated problem is nonconvex and its optimal solution is generally intractable, hence, we proposed two sub-optimal methods to efficiently solve this problem. In the first method, we adopted DC programming combined with chance-constraint programming to obtain the deterministic equivalents of the probabilistic constraints. Since the performance of DC programming depends on a good initial point, we proposed a second method, which relies only on convex programming and eliminates the user defined initialization. Similar to the first method, chance-constraint programming was merged into the convex programming to handle the probabilistic constraints. The computer simulations demonstrated that both methods significantly reduce total energy consumption in comparison to processing tasks locally.

In Chapter 5, we considered a more general task offloading scenario compared to Chapter 4, in which multiple devices can offload their tasks to nearby fog devices via D2D links and to a centralized ES. Since an ES has more computation resources relative to fog devices, it can process more tasks simultaneously, which can increase the overall energy efficiency. However, network traffic and wireless channel conditions may limit its utilization and affect the task offloading performance. On the contrary, D2D-aided fog computing can significantly reduce the task uploading time by leveraging the close proximity of fog devices, hence, mitigate energy consumption. In this regard, we addressed the optimal resource allocation problem in the considered multi-device D2D-assisted fog computing scenario to minimize the total energy consumption subject to the constraints on transmit powers, computation resources and task processing times. Since the formulated problem is nonconvex, we proposed two sub-optimal methods. In the first method, we initially analyzed the original problem and then showed that it can be relaxed into sequence of convex sub-problems, whose solutions can be efficiently obtained by using the convex optimization theory. In light of this first method, we proposed a second method, which depends only on a low-complexity heuristic resource allocation strategy to avoid costly calculations of gradients and Hessian matrices in the solution process. We also showed in detail the computational complexity of this method in terms of key system parameters such as the number of mobile devices and task offloading destinations. Finally, we derived a lower-bound on the total energy consumption as a benchmark to compare the performance of the proposed methods. The simulation results showed that both methods achieve a near-optimal solution in terms of energy efficiency. In what follows we present some future work that may extend our research.

6.2 Future Works

Although in this thesis we addressed some of the fundamental and practical challenges upon implementing D2D communications, future research may provide additional merits by considering the following aspects.

First, as opposed to Chapter 3, where we consider a clock synchronization problem in stationary networks and assume time-invariant channels, achieving distributed synchronization subject to time-varying channels can be investigated. In the case of high device mobility, the coherence time and the assumption of time-invariant channels may no longer be valid due to Doppler effects. Consequently, the performance of pulse-based clock synchronization will be diminished due to the additional CFO as well as the effect of time varying channels, i.e., $h_{ij}(\tau, t) = \sum_{p \in \mathcal{P}} \rho_{ijp}(t)\delta(t - \tau_{ijp}(t)), i \in \mathcal{T}_\nu \; j \in \mathcal{J}_\nu$ as they introduce time varying propagation delays to received signals. Even though the proposed Alternating Transceiver Mode algorithm in Section 3.3.1 would eliminate the randomness that is introduced to received synchronization signals due to arbitrarily choosing synchronization signal broadcasters, the bias term $\beta_j[\nu]$ in Section 3.13 would be time-varying, i.e, $\beta_j[\nu, t]$. Therefore, future research might focus on developing an adaptive-bias tracking algorithm to eliminate the effect of time dependent bias. Hereafter, a similar timing-advance based synchronization protocol can be investigated to allow devices to first synchronize themselves and then proceed to data communication. In addition, maintaining the achieved synchronization in such a scenario also poses several practical issues as synchronization is not only deteriorating due to clock skew but also due to the mobility of devices. Hence, more frequent re-initialization of the synchronization process might be needed among the synchronized devices. Consequently, in comparison to the proposed synchronization protocol, which lets synchronized devices stay idle and passively maintain synchronization as much as clock skew allows, future research should put more emphasis on energy efficiency.

Second, compared to the assumption of deterministic task arrival during task offloading in Chapter 4 and Chapter 5, future research may extend the resource allocation problem by including randomness in task arrival during task offloading. In this way, the optimal resource allocation strategy should take the expected task sizes into account while allocating the limited computation resources. Moreover, future research may investigate the effect of imperfect Channel State Information (CSI) in task offloading by assuming the full channel properties are not available at the devices, hence, task uploading time becomes random. Also, similar to Fig. 3.1, fog computing in a partial mesh topology with dynamic network size, where devices arbitrarily join or leave the network, can be investigated. In such a scenario, one or more fog devices can be utilized for task offloading by multiple active devices simultaneously. Hence, such a task offloading framework should seamlessly adapt to the change in the availability of computation resources, which imposes the total computation capability of devices to be random. Finally, adaptive resource allocation and device association problems in a task offloading scenario can be jointly addressed by considering mobile networks with high mobility, wherein an efficient hand-off strategy can be investigated for device discovery.

Appendix A

Reduction of TO in Multi-Device Setting

We consider distributed multi-device pulse-based synchronization over multipath channels with half-duplex technology. Consequently, there is no single timing reference for the clocks of the devices to converge to. Therefore, we analyze the reduction of timing offset, which is interpreted as the synchronization error by the devices. We assume each device runs the proposed protocol and, based on Algorithm 3.1, the devices keep alternating their transceiver mode at each clock tick ν after initialization. We will use the index j to denote a receiver device at clock tick ν and i to denote a transmitter device. These devices become transmitters and receivers, respectively, at the next clock tick, i.e., $j \in \mathcal{R}_\nu = \mathcal{T}_{\nu+1}$ and $i \in \mathcal{T}_\nu = \mathcal{R}_{\nu+1}$. We further assume that the network remains constant, i.e., no new devices join or leave the network, and channels are time-invariant. Therefore, the cardinality of the sets is $|\mathcal{T}_\nu| = T$ and $|\mathcal{R}_\nu| = R$. For simplicity of the analysis and with no loss of generality, we assume that the signal contributions are coming from the same time slot of the receiver. Hence, we have $\eta = \nu$ and the set of pairs formed by the index of transmitter devices and the path indices contributing to the received signal of the jth receiver device becomes $\mathcal{D}_j^{\nu,\nu}$. In addition, we use the superscript $\mathcal{T}$ and $\mathcal{R}$ on the device clock models to indicate their transceiver mode at the corresponding clock ticks. Thus, in the high SNR region, the weighted average TO estimate over multipath channels at the device $j \in \mathcal{R}_\nu$ equals to:

$$\widehat{\Delta t}_j[\nu] = \sum_{(i,p)\in\mathcal{D}_j^{\nu,\nu}} \mu_{ijp}\big(t_i^{\mathcal{T}}[\nu] + \tau_{ijp}\big) - t_j^{\mathcal{R}}[\nu]. \tag{A.1}$$

For the ith device $\widehat{\Delta t}_i[\nu+1]$ can be defined similarly. Then, the average TO estimate of the jth device when it becomes a receiver again is given as follows:

$$\begin{aligned}\widehat{\Delta t}_j[\nu+2] &= \sum_{(i,p)\in\mathcal{D}_j^{\nu+2,\nu+2}} \mu_{ijp}\big(t_i^{\mathcal{T}}[\nu+2]+\tau_{ijp}\big) - t_j^{\mathcal{R}}[\nu+2]\\ &= \sum_{(i,p)\in\mathcal{D}_j^{\nu+2,\nu+2}} \mu_{ijp}\big(t_i^{\mathcal{R}}[\nu+1]+\alpha_i T_0-2\widehat{\beta}_i[\nu+1]+\widehat{\Delta t}_i[\nu+1]\\ &\quad +\tau_{ijp}\big)-t_j^{\mathcal{T}}[\nu+1]-\alpha_j T_0\\ &= \sum_{(i,p)\in\mathcal{D}_j^{\nu+2,\nu+2}} \mu_{ijp}\big(t_i^{\mathcal{T}}[\nu]+2\alpha_i T_0-2\widehat{\beta}_i[\nu+1]+\widehat{\Delta t}_i[\nu+1]\\ &\quad +\tau_{ijp}\big)-t_j^{\mathcal{R}}[\nu]-2\alpha_j T_0+2\widehat{\beta}_j[\nu]-\widehat{\Delta t}_j[\nu]\end{aligned} \tag{A.2}$$

where $t_i^{\mathcal{T}}[\nu+2]$ and $t_j^{\mathcal{R}}[\nu+2]$ are obtained from (3.15) and (3.11) at the clock tick $(\nu+1)$, respectively, knowing that the receiver device becomes transmitter and vice versa (cf. Section 3.3.1). We have $\mathcal{D}_j^{\nu,\nu}=\mathcal{D}_j^{\nu+2,\nu+2}$, and using (A.1), the weighted average TO estimate in (A.2) is further simplified to:

$$\begin{aligned}\widehat{\Delta t}_j[\nu+2] &= \sum_{(i,p)\in\mathcal{D}_j^{\nu+2,\nu+2}} \mu_{ijp}\big(\widehat{\Delta t}_i[\nu+1]-2\widehat{\beta}_i[\nu+1]+2\alpha_i T_0\big)\\ &\quad +2\widehat{\beta}_j[\nu]-2\alpha_j T_0\end{aligned} \tag{A.3}$$

Similarly, the weighted average TO estimate at the ith device is:

$$\begin{aligned}\widehat{\Delta t}_i[\nu+3] &= \sum_{(j,p)\in\mathcal{D}_i^{\nu+3,\nu+3}} \mu_{ijp}\big(\widehat{\Delta t}_j[\nu+2]-2\widehat{\beta}_j[\nu+2]+2\alpha_j T_0\big)\\ &\quad +2\widehat{\beta}_i[\nu+1]-2\alpha_i T_0\end{aligned} \tag{A.4}$$

where it is simplified as in (A.3) by using $\widehat{\Delta t}_i[\nu+1]$ and for the sets $\mathcal{D}_i^{\nu+1,\nu+1}=\mathcal{D}_i^{\nu+3,\nu+3}$.

To track the error at the devices when they are receivers, we can generalize (A.3) for the device $j\in\mathcal{R}_\nu$ as follows:

$$\mathbf{y}[\nu]=\mathbf{W}\big(\mathbf{x}[\nu-1]-2\mathbf{r}[\nu-1]+2\mathbf{a}T_0\big)+2\mathbf{p}[\nu-2]-2\mathbf{b}T_0 \tag{A.5}$$

whereas the generalization of (A.4) for the device $i\in\mathcal{R}_{\nu+1}$ is given as:

$$\mathbf{x}[\nu+1]=\mathbf{V}\big(\mathbf{y}[\nu]-2\mathbf{p}[\nu]+2\mathbf{b}T_0\big)+2\mathbf{r}[\nu-1]-2\mathbf{a}T_0 \tag{A.6}$$

where $\mathbf{y}[\nu]$ and $\mathbf{x}[\nu+1]$ are the vectors that contain the weighted average TO estimates of the receiver devices in the sets $\mathcal{R}_\nu$ and $\mathcal{R}_{\nu+1}$, respectively. Here, the notation $[\cdot]^\top$ is the transpose of a vector. In addition, $\mathbf{W}$ and $\mathbf{V}$ are the matrices with compatible dimensions that contain the normalized channel weights of the jth and the ith devices, i.e., μ_{ijp} and μ_{jip}, respectively. Note that the row sums of $\mathbf{W}$ and $\mathbf{V}$ are normalized to one and since their product is a square matrix, it becomes a right stochastic matrix [140]. This feature will be used later in the proof. Furthermore, $\mathbf{p}[\nu]$ and $\mathbf{b}$ are the vectors that contain the biases and clock skews of the devices in $\mathcal{R}_\nu$, whereas $\mathbf{r}[\nu-1]$ and $\mathbf{a}$ are the vectors that contain the biases and clock skews of the devices in $\mathcal{R}_{\nu-1}$.

For simplicity of the analysis, we assume fixed step sizes as in (3.16), hence, online bias estimates are generalized for both $j \in \mathcal{R}_\nu$ and $i \in \mathcal{R}_{\nu+1}$ as follows:

$$\begin{aligned}\mathbf{p}[\nu] &= \mathbf{p}[\nu-2] + \mathbf{m} \odot \operatorname{sgn}(\mathbf{y}[\nu-2]) \\ \mathbf{r}[\nu+1] &= \mathbf{r}[\nu-1] + \mathbf{n} \odot \operatorname{sgn}([\mathbf{x}[\nu-1]) \end{aligned} \tag{A.7}$$

where $\odot$ is the Hadamard product, $\mathbf{m}$ and $\mathbf{n}$ are the vectors with compatible lengths, which contain the step sizes of the jth and the ith devices in the sets $\mathcal{R}_\nu$ and $\mathcal{R}_{\nu+1}$, respectively,

To continue the analysis, we assume that the synchronization error of each device is greater than the synchronization threshold chosen in the network, i.e., $|\mathbf{y}[\nu]| > \lambda_{\text{sync}}$ and $|\mathbf{x}[\nu+1]| > \lambda_{\text{sync}}$, where the absolute values and comparisons are element-wise. In other words, **C1** is satisfied, hence, the devices should try to decrease their errors until $\overline{\mathbf{C1}}$ is satisfied, where they stop updating their bias estimates and maintain the reduced error level.

In order to show the reduction in absolute value of the synchronization error, we compare the weighted average TO estimates at two consecutive clock ticks, i.e., $|\mathbf{y}[\nu+2]|$ and $|\mathbf{y}[\nu]|$ as ν increases. Hence, by using (A.6) in (A.5) at the $(\nu+2)$th clock tick, we obtain the following:

$$\begin{aligned}\mathbf{y}[\nu+2] &= \mathbf{W}\big(\mathbf{x}[\nu+1] - 2\mathbf{r}[\nu+1] + 2\mathbf{a}T_0\big) + 2\mathbf{p}[\nu] - 2\mathbf{b}T_0 \\ &= \mathbf{W}\mathbf{V}\mathbf{y}[\nu] - 2(\mathbf{W}\mathbf{V}-\mathbf{I})\mathbf{p}[\nu] - 2\mathbf{W}\mathbf{n}\odot\operatorname{sgn}(\mathbf{x}[\nu-1]) \\ &\quad + 2(\mathbf{W}\mathbf{V}-\mathbf{I})\mathbf{b}T_0 \end{aligned} \tag{A.8}$$

where $\mathbf{I}$ is the identity matrix with compatible size. By subtracting $\mathbf{y}[\nu]$ from the both sides, we further obtain:

$$\begin{aligned}\mathbf{y}[\nu+2] &= \mathbf{y}[\nu] + \big(\mathbf{W}\mathbf{V}-\mathbf{I}\big)\overbrace{\big(\mathbf{y}[\nu]-2\mathbf{p}[\nu]+2\mathbf{b}T_0\big)}^{\mathbf{q}[\nu]} \\ &\quad + 2\mathbf{W}\big(\mathbf{r}[\nu-1]-\mathbf{r}[\nu+1]\big) \\ &= \mathbf{y}[\nu]+\big(\mathbf{W}\mathbf{V}-\mathbf{I}\big)\mathbf{q}[\nu]-2\mathbf{W}\mathbf{n}\odot\operatorname{sgn}(\mathbf{x}[\nu-1]).\end{aligned} \tag{A.9}$$

The multiplication of $(\mathbf{W}\mathbf{V}-\mathbf{I})$, which is a zero row-sum matrix, with a vector that has identical elements yields a zero vector. In this case, if all the elements in $\mathbf{q}[\nu]$ approaches the same values as ν increases, then $(\mathbf{W}\mathbf{V}-\mathbf{I})\mathbf{q}[\nu] \rightarrow \mathbf{0}$, which is a zero vector. In order to prove that, we can re-arrange $\mathbf{q}[\nu]$ by using (A.5) and (A.6) through recursive iterations as follows:

$$\begin{aligned}\mathbf{q}[\nu] = {} & \overbrace{\mathbf{W}\mathbf{V}\mathbf{W}\mathbf{V}\overbrace{\big(\mathbf{y}[\nu-4]-2\mathbf{p}[\nu-4]+2\mathbf{b}T_0\big)}^{\mathbf{q}_2[\nu]}}^{\mathbf{q}_1[\nu]} \\ & - 2\mathbf{W}\mathbf{V}\mathbf{W}\mathbf{m}\odot\operatorname{sgn}(\mathbf{x}[\nu-5]) \\ & - 2\mathbf{W}\mathbf{V}\mathbf{n}\odot\operatorname{sgn}(\mathbf{y}[\nu-4]) \\ & - 2\mathbf{W}\mathbf{m}\odot\operatorname{sgn}(\mathbf{x}[\nu-3]) \\ & - 2\mathbf{n}\odot\operatorname{sgn}(\mathbf{y}[\nu-2])\end{aligned} \tag{A.10}$$

Note that by multiplying the same stochastic matrices recursively, i.e., $\mathbf{W}\mathbf{V}\mathbf{W}\mathbf{V}...$, we obtain a right stochastic matrix that has identical elements in each column. Then, multiplication of $\mathbf{W}\mathbf{V}\mathbf{W}\mathbf{V}\mathbf{q}_2[\nu] = \mathbf{q}_1[\nu] \approx \big[q_1[\nu], q_1[\nu], ..., q_1[\nu]\big]^\top$ approximates a vector that has identical elements. Thus, $\big(\mathbf{W}\mathbf{V}-\mathbf{I}\big)\mathbf{q}_1[\nu] \approx \mathbf{0}$. In addition, the remaining terms in $\mathbf{q}[\nu]$ include the fixed step sizes, which are the same for each device, and the sign of the error remains unchanged[1]. Therefore, they can be re-arranged such that -2$(\mathbf{W}\mathbf{V}-\mathbf{I})\mathbf{z} = \mathbf{0}$, where $\mathbf{z} = \mathbf{W}\mathbf{m}\odot\operatorname{sgn}(\mathbf{x}[\nu-5]) = \mathbf{W}\mathbf{m}\odot\operatorname{sgn}(\mathbf{x}[\nu-3])$. Hence, we can conclude that $(\mathbf{W}\mathbf{V}-\mathbf{I})\mathbf{q}[\nu] \rightarrow \mathbf{0}$ as ν increases and we left with the following:

$$\mathbf{y}[\nu+2] = \mathbf{y}[\nu] - 2\mathbf{W}\mathbf{n}\odot\operatorname{sgn}(\mathbf{x}[\nu-1]) \tag{A.11}$$

Note that sign of $\mathbf{y}[\nu]$ is dominated by $\mathbf{W}\mathbf{x}[\nu-1]$ as given in (A.5). Then, $\operatorname{sgn}(\mathbf{y}[\nu]) = \operatorname{sgn}(\mathbf{W}\mathbf{x}[\nu-1])$, or equivalently $\operatorname{sgn}(\mathbf{x}[\nu-1]) = \operatorname{sgn}(\mathbf{W}^\dagger\mathbf{y}[\nu])$, where $\mathbf{W}^\dagger$ is the pseudo-inverse

[1]Synchronization error must be decreased to zero before changing its sign.

of $\mathbf{W}$. By multiplying both sides in (A.11) with $\mathbf{W}^{\dagger}$, we obtain the following:

$$\mathbf{y}'[\nu+2] = \mathbf{y}'[\nu] - 2\mathbf{n} \odot \operatorname{sgn}(\mathbf{y}'[\nu]) \tag{A.12}$$

where $\mathbf{y}'[\nu] = \mathbf{W}^{\dagger}\mathbf{y}[\nu]$. Since the step-sizes are always positive, we can conclude that the quantity in (A.12) is reducing by $2\mathbf{n}$ at each clock tick. In other words, the synchronization error decreases in absolute value, that is $|\mathbf{y}'[\nu+2]| < |\mathbf{y}'[\nu]|$, where the vectors are compared element-wise.

Finally, when the desired synchronization error is achieved, i.e., $\overline{\mathbf{C1}}$ is satisfied, the devices switch to **Fixed Bias State**, hence, stop updating their bias estimates. In this case, $\mathbf{r}[\nu+1] = \mathbf{r}[\nu-1]$ or equivalently the term $\mathbf{n} \odot \operatorname{sgn}(\mathbf{x}[\nu-1])$ is no longer present in (A.7). Hence, from (A.11), we can conclude that the synchronization errors are reached a steady-state level that is smaller than or equal to the pre-defined synchronization error, i.e., $\mathbf{y}[\nu+2] = \mathbf{y}[\nu] \leq \lambda_{\text{sync}}$ as $\nu \to \infty$.

Appendix B

Rate of TO Reduction

We consider a simplified scenario consisting of two devices labeled as D1 and D2 communicating over a flat reciprocal channel with propagation delay τ. We assume that the devices follow the alternating transceiver mode and based on p_{tr}, D1 operates as a transmitter, whereas D2 is a receiver at the νth clock tick. We further assume that the relative clock skew is negligible, i.e., $\Delta\alpha_{12} = \alpha_1 - \alpha_2 = 0$. Hence, the first TO estimate of D2 is equal to $\widehat{\Delta t}_{12}[\nu] = t_1[\nu] - t_2[\nu] + \tau$. However, $\widehat{\Delta t}_{12}[\nu]$ is the initialization error due to misaligned clock phases. To observe the effect of the protocol, we consider the next TO estimate of D2, which occurs at the $(\nu + 2)$th clock tick (see Section 3.3.1) as follows:

$$\begin{aligned}
\widehat{\Delta t}_{12}[\nu+2] &= t_1[\nu+2] - t_2[\nu+2] + \tau \\
&= t_1[\nu+1] + \alpha_1 T_0 + \widehat{\Delta t}_{21}[\nu+1] - 2\widehat{\beta}_1[\nu+1] \\
&\quad - t_2[\nu+1] - \alpha_2 T_0 + \tau \\
&= t_1[\nu+1] - t_2[\nu+1] - \tau + \widehat{\Delta t}_{21}[\nu+1] + 2\tau - 2\widehat{\beta}_1[\nu+1] \\
&= 2(\tau - \widehat{\beta}_1[\nu+1])
\end{aligned} \tag{B.1}$$

Note that the corresponding clock simplifications, i.e., $t_1[\nu+2]$ and $t_2[\nu+2]$, are derived respectively from (3.11) and (3.15) for a transmitter and a receiver device when $\epsilon = 1$. We further note that $\widehat{\beta}_1[\nu+1] = \widehat{\beta}^{\text{init}}$ and without loss in generality, we assume $\widehat{\Delta t}_{12}[\nu+2] > 0$.

Similarly, at the $(\nu+4)$th clock tick, D2 estimates its TO again and it is given by

$$\begin{aligned}\widehat{\Delta t}_{12}[\nu+4] &= 2(\tau - \widehat{\beta}_1[\nu+3]) \\ &= 2(\tau - \widehat{\beta}_1[\nu+1] - \gamma\,\mathrm{sgn}(\widehat{\Delta t}_{21}[\nu+3]) \\ &= \widehat{\Delta t}_{12}[\nu+2] - 2\gamma\,\mathrm{sgn}(\widehat{\Delta t}_{21}[\nu+3]) \qquad \text{(B.2)}\end{aligned}$$

where the bias estimate of D1, i.e., $\widehat{\beta}_1[\nu+3]$, is simplified accordingly from (3.16) and we assume a fixed step size, i.e., γ, for updating the bias estimate. Note that TO estimate at each device must decrease to zero before changing sign, consequently, we have $\mathrm{sgn}(\widehat{\Delta t}_{21}[\nu+3]) = \mathrm{sgn}(\widehat{\Delta t}_{12}[\nu+4]) = \mathrm{sgn}(\widehat{\Delta t}_{12}[\nu+2]) = \mathrm{sgn}(\widehat{\Delta t}_{12}[\nu])$. Since we assume $\widehat{\Delta t}_{12}[\nu+2] > 0$, we can simplify (B.2) as $\widehat{\Delta t}_{12}[\nu+4] = \widehat{\Delta t}_{12}[\nu+2] - 2\gamma$. Finally, the change in the TO estimate of D2 for two consecutive clock ticks where it operates as a receiver is given by:

$$m = \frac{\widehat{\Delta t}_{12}[\nu+4] - \widehat{\Delta t}_{12}[\nu+2]}{(\nu+4)-(\nu+2)-1} = -2\gamma \qquad \text{(B.3)}$$

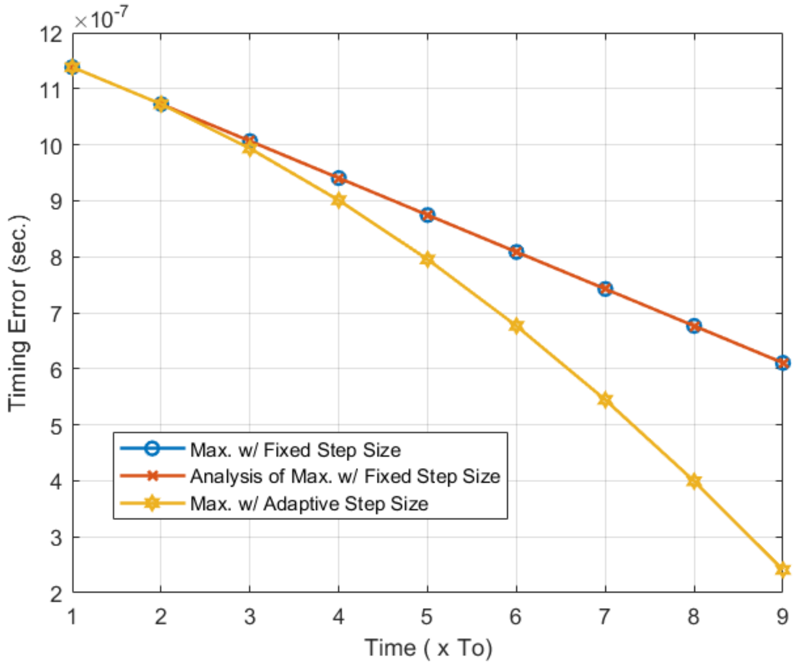


Fig. B.1. Synchronization performance based on the scenario in Appendix A with remaining parameters chosen from Table I.

In Fig. B.1, we plot the maximum synchronization error (3.33) against a straight line with negative slope as given by (B.3). The results show a very close match between the two curves, thereby supporting our analysis. For comparison, we also plot the synchronization error with adaptive step size.

Appendix C

Expectation of TO Estimate

We assume two devices labeled D1 and D2 communication over a flat reciprocal channel with propagation delay τ and the relative clock skew is negligible, i.e., $\Delta\alpha_{12} = \alpha_1 - \alpha_2 = 0$. In order for a device, say D2, to estimate the synchronization for updating its clock, it should operate as a receiver, while D1 should be a transmitter or vice versa. Therefore, the probability of this event is $p_e = 2p_{\text{tr}}(1 - p_{\text{tr}})$. Then, the synchronization error at the νth clock tick for D2 is $\Delta t_{12}[\nu] = t_1[\nu] + \tau - t_2[\nu]$, whereas for D1, it is $\Delta t_{21}[\nu] = t_2[\nu] + \tau - t_1[\nu]$. Hence, the expected initial synchronization error can be given as $\Delta\theta = |t_1[\nu] - t_2[\nu]| = |\theta_1 - \theta_2| = |\Delta\theta_{12}| = |\Delta\theta_{21}|$. Now, based on the alternating transceiver mode and the clock updates according to (11) and (15), where we assume $\widehat{\beta}_{12}[\nu] = \widehat{\beta}_{21}[\nu] \approx \tau \ \forall\nu$, the expected TO estimate at D2 (similar for D1) is equal to:

$$\mathbb{E}\big[\widehat{\Delta t}_{12}[\nu]\big] = \Bigg(\underbrace{(1-p_e)^{\nu}}_{\mathrm{P}_1} + \underbrace{p_e(1-\epsilon)^{(\nu-1)}}_{\mathrm{P}_2} + \underbrace{\sum_{k=1}^{\nu-1}(1-p_e)^k p_e(1-\epsilon)^{(\nu-1-k)}}_{\mathrm{P}_3}\Bigg)\Delta\theta, \ \forall\nu \geq 1 \quad \text{(C.1)}$$

Here, P_1 is the probability that the devices never operate at the opposite modes, which is the worst scenario since they cannot detect the error and update their clocks. Furthermore, P_2 is the probability that the devices operate on the opposite modes at the first clock tick and then start alternating between them, hence, it is the best scenario. Finally, P_3 is the probability that the devices start alternating their mode and update their clocks once they operate at the opposite modes, which is a mixed scenario.

Appendix D

Convexity of Problem $\mathscr{P}_2$

We begin with proving the each function that forms the objective function of Problem $\mathscr{P}_2$ given in (5.11) is convex. Since the summation of convex functions is a convex function, without loss of generality, we can consider an active device and a single offloading device $\kappa \in \mathcal{K}_i = \{i0\}$ to prove the convexity of (5.11) as follows:

$$\psi_i(\mathbf{p}_i, \mathbf{b}_i) = \underbrace{\frac{\mu(b_i c_i)^3}{\left(t_i^{\max}\right)^2}}_{\psi_{\text{loc}}} + \underbrace{\frac{\mu(b_\kappa c_i)^3}{\left(t_i^{\max} - \frac{b_\kappa}{R_\kappa}\right)^2}}_{\psi_{\text{off}}},\ \forall i \in \mathcal{I}$$

We denote the Hessian of ψ_{loc} and ψ_{off} by $\Delta^2\psi_{\text{loc}}$ and $\Delta^2\psi_{\text{off}}$, respectively. Obtaining $\Delta^2\psi_{\text{loc}}$ is straightforward since it only depends on b_i. Hence, after calculating its eigenvalues as 0 and $\frac{6 b_i \mu c_i^3}{(t_i^{\max})^2}$, we can determine that $\Delta^2\psi_{\text{loc}}$ is positive semi-definite and ψ_{loc} is convex since $b_i \geq 0,\ \forall i \in \mathcal{I}$. For the Hessian of ψ_{off}, we have:

$$\Delta^2\psi_{\text{off}} = \begin{bmatrix} \dfrac{\partial {\psi_{\text{off}}}^2}{\partial b_\kappa^2} & \dfrac{\partial {\psi_{\text{off}}}^2}{\partial b_\kappa \partial P_\kappa} \\ \dfrac{\partial {\psi_{\text{off}}}^2}{\partial P_\kappa \partial b_\kappa} & \dfrac{\partial {\psi_{\text{off}}}^2}{\partial P_\kappa^2} \end{bmatrix}$$

where we can show that its trace and determinant are positive to prove the convexity of ψ_{off}. Specifically, the first element of the main diagonal is equal to:

$$\frac{\partial {\psi_{\text{off}}}^2}{\partial b_\kappa^2} = \frac{6\mu b_\kappa^3 c_i^3}{R_\kappa^2\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^2} + \frac{12\mu b_\kappa^2 c_i^3}{R_\kappa\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^2} + \frac{6\mu b_\kappa c_i^3}{\left(t_i^{\max} - t_\kappa^{\text{up}}\right)^2}$$

while the second entry is:

$$\frac{\partial {\psi_{\text{off}}}^2}{\partial P_\kappa^2} = \frac{m_i b_\kappa^4}{u\big(t_i^{\max} - t_\kappa^{\text{up}}\big)^3} + \frac{2 m_i b_\kappa^4}{u_\kappa \ln(v_\kappa)\big(t_i^{\max} - t_\kappa^{\text{up}}\big)^3} + \frac{3 b_\kappa^5 m_i \ln(2)}{W u_\kappa \ln^2(v)\big(t_i^{\max} - t_\kappa^{\text{up}}\big)^4}$$

where $m_i = 2\mu c_i^3 G_\kappa^2 \ln(2)$, $v_\kappa = (1 + \frac{P_\kappa G_\kappa}{N_0})$ and $u_\kappa = W{N_0}^2 v_\kappa^2 \ln^2(v_\kappa)$ are positive variables. Since the task uploading time is always smaller than the task processing deadline due to constraint (5.15d), i.e., $t_i^{\max} \leq \alpha t_\kappa^{\text{up}}$, the trace of $\Delta^2 \psi_{\text{off}}$ is positive.

Furthermore, the determinant of $\Delta^2 \psi_{\text{off}}$ is calculated as:

$$\det(\Delta^2 \psi_{\text{off}}) = \frac{4\mu^2 c_i^6 b_\kappa^5 G_\kappa^2 \ln(2) s_i}{N_0^2 W^3 \ln^5(v_\kappa) v_\kappa^2 \big(t_i^{\max} - t_\kappa^{\text{up}}\big)^7}$$

where

$$s_i = b_\kappa^2 \ln^2(2) + W^2 t_i^{\max} \ln^2(v_\kappa)\big(6 t_i^{\max} - 7 t_\kappa^{\text{up}} + 3 t_i^{\max} \ln(v_\kappa)\big)$$

We can show that $\det(\Delta^2 \psi_{\text{off}})$ is positive if s_i and $(t_i^{\max} - t_\kappa^{\text{up}})$ are positive. Both can be ensured simultaneously if $t_i^{\max} > 7/6 t_\kappa^{\text{up}}$. Hence, we must select $\alpha \in (0, 0.85)$ in constraint (5.15d) to keep the determinant positive for any values of b_κ and P_κ, $\forall \kappa \in \mathcal{K}_i, \forall i \in \mathcal{I}$. Note that the energy efficiency increases when we set $\alpha = 0.85$ as it is the highest value to relax the constraint on time to upload intended tasks. Therefore, ψ_{off} is a convex function over the convex set based on the constraints in problem $\mathscr{P}_2$. Finally, we can conclude that $\psi_i(\mathbf{p}_i, \mathbf{b}_i),\ \forall i \in \mathcal{I}$ is convex as it is a summation of convex functions.

Appendix E

Optimal Task Offloading for the Ideal Case

We calculate the lower bound on the total energy consumption (5.8) by evaluating the optimal task offloading strategy in the ideal scenario. As demonstrated in Section 5.3.1, the task size d_i, regardless of how it is split, must be computed at exactly $t_i^{\max}$ to minimize energy consumption. Hence, we can calculate the total resource required for computing d_i as follows:

$$f_i^{\text{tot}} = \frac{d_i c_i}{t_i^{\max}} \tag{E.1}$$

At this point, the main objective becomes finding the optimal task splitting strategy, i.e, $d_i = b_i + \sum_{\kappa \in \mathcal{K}_i} b_\kappa$, such that the allocated computation resources, i.e., f_i and f_κ $\forall \kappa \in \mathcal{K}_i$, minimize the total energy consumption in (5.8).

Suppose that there is no constraint on the computation capability of device $k \in \mathcal{K}_i$ and the task uploading speed is very high. Hence, we can assume $P_\kappa = 0$ and $t_\kappa^{\text{up}} = 0$, which yields $t_i^{\text{co}} = t_\kappa^{\text{co}} = t_i^{\max}$, $\forall i \in \mathcal{I}$ $\forall \kappa \in \mathcal{K}_i$. Then, based on (5.2) and (5.6), we have $b_i = \dfrac{f_i t_i^{\max}}{c_i}$ and $b_\kappa = \dfrac{f_\kappa t_i^{\max}}{c_i}$, $\forall \kappa \in \mathcal{K}_i$, equivalently, the total required resource is $f_i^{\text{tot}} = f_i + \sum_{\kappa \in \mathcal{K}_i} f_\kappa$. Therefore, by using (5.3) and (5.7), the total energy consumption of the ith device is equal to:

$$E_i = \mu f_i^3 t_i^{\max} + \sum_{\kappa \in \mathcal{K}_i} \mu f_\kappa^3 t_i^{\max} \tag{E.2}$$

To obtain the lower bound for (E.2), we define the following problem:

$$\min_{\mathbf{f}} \quad \sum_{i\in\mathcal{I}} E_i \tag{E.3}$$

$$\text{s.t.} \quad f_i + \sum_{\kappa\in\mathcal{K}_i} f_\kappa = f_i^{\text{tot}}, \ \forall i \in \mathcal{I} \tag{E.4}$$

Since the problem is convex, we can find the optimal value by using Lagrange multiplier method, where the Lagrangian function for the ith device with a Lagrange multiplier λ_i is defined as follows:

$$\mathcal{L}_i(\mathbf{f}_i, \lambda_i) = E_i - \lambda_i(f_i + \sum_{\kappa\in\mathcal{K}_i} f_\kappa - f_i^{\text{tot}}) \tag{E.5}$$

Then, by taking the gradient of (E.5) and solving it as $\nabla\mathcal{L}_i(\mathbf{f}_i, \lambda_i) = 0$, we have:

$$3\mu \begin{bmatrix} f_i^2 \\ f_{i0}^2 \\ f_{i1}^2 \\ \vdots \\ f_{iK}^2 \end{bmatrix} t_i^{\max} - \lambda_i \mathbf{1} = 0 \tag{E.6}$$

where $\mathbf{1}$ is the column vector of all-ones. Hence, we obtain $f_i = f_\kappa = \sqrt{\frac{\lambda_i}{3\mu t_i^{\max}}}$ $\forall\kappa \in \mathcal{K}_i$, and by using (E.4), we have $\lambda_i = 3\mu t_i^{\max}\big(\frac{f_i^{\text{tot}}}{|\mathcal{K}_i|+1}\big)^2$. Consequently, the optimal computation resources are calculated as $f_i = f_\kappa = \dfrac{f_i^{\text{tot}}}{|\mathcal{K}_i|+1}$, $\forall\kappa \in \mathcal{K}_i$. Finally, by considering (E.1), the optimal task splitting based on (5.2) and (5.6) becomes $b_i = b_\kappa = \dfrac{d_i}{|\mathcal{K}_i|+1}$.

This shows that the minimum total energy consumption is achieved when the tasks are split equally. Accordingly, the allocated computation resource at each device should be identical, where we can denote the optimum resource as $f_i^{\text{opt}} = \frac{f_i^{\text{tot}}}{|\mathcal{K}_i|+1}$. Therefore, the lower bound of the total energy consumption for the given task size d_i can be calculated as follows:

$$E_i^* = \mu\big(|\mathcal{K}_i| + 1\big)(f_i^{\text{opt}})^3 t_i^{\max} \leq E_i \tag{E.7}$$

References

[1] Cisco Annual Internet Report, 2018-–2023 [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Cisco visual networking index: global mobile data traffic forecast update, 2016—2021 [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[3] A. Asadi, Q. Wang and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys and Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.

[4] R. I. Ansari et al., "5G D2D networks: Techniques, challenges, and future prospects," *IEEE Systems Jour.*, vol. 12, no. 4, pp. 3970–3984, Dec. 2018.

[5] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device communication in 5G cellular networks: Challenges, solutions, and future directions," *IEEE Commun. Mag.*, vol. 52, pp. 86–92, May 2014.

[6] B. a. Coll-Perales, J. Gozalvez and J. L. Maestre, "5G and beyond: Smart devices as part of the network fabric," *IEEE Network*, vol. 33, no. 4, pp. 170–177, Jul. 2019.

[7] J. Cao, M. Ma, H. Li, R. Ma, Y. Sun, P. Yu, L. Xiong, "A survey on security aspects for 3GPP 5G networks," *IEEE Commun. Surveys and Tutorials*, vol. 22, no. 1, pp. 170–195, 2020.

[8] D. H. Lee, K. W. Choi, W. S. Jeon, and D. G. Jeong, "Resource allocation scheme for device-to-device communication for maximizing spatial reuse," in *Proc. IEEE Wireless Commun. and Net. Conf.*, pp. 112–117,Apr. 2013.

[9] B. Kaufman, J. Lilleberg, and B. Aazhang, "Spectrum sharing scheme between cellular users and ad-hoc device-to-device users," *IEEE Trans. Wireless Commun.*, vol. 12, no. 3, pp. 1038–1049, Mar. 2013.

[10] M. Zulhasnine, C. Huang, and A. Srinivasan, "Efficient resource allocation for device-to-device communication underlaying LTE Network," in *Proc. IEEE Int. Conf. on Wireless and Mobile Comp., Networking and Commun.*, pp. 368—375, Oct. 2010.

[11] C. Yu, K. Doppler, C. B. Ribeiro and O. Tirkkonen, "Resource sharing optimization for device-to-Device communication underlaying cellular networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 8, pp. 2752—63, Aug. 2011.

[12] "Technical specification group radio access network; study on LTE device-to-device proximity services," 3rd Generation Partnership Project (3GPP), TR 36.843, Mar. 2014, Sections A.2.1.1 - A.2.1.2. [Online].

[13] F. Jameel, Z. Hamid, F. Jabeen, S. Zeadally and M. A. Javed, "A survey of device-to-device communications: Research issues and challenges," *IEEE Commun. Surveys and Tutorials*, vol. 20, no. 3, pp. 2133–2168, 2018.

[14] L. Wei, R. Q. Hu, Y. Qian and G. Wu, "Enable device-to-device communications underlaying cellular networks: Challenges and research aspects," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 90–96, June 2014.

[15] W. Cheng, X. Zhang and H. Zhang, "Optimal power allocation with statistical QoS provisioning for D2D and cellular communications over underlaying wireless networks," *IEEE Jour. on Selec. Areas in Commun.*, vol. 34, no. 1, pp. 151–162, Jan. 2016.

[16] P. Phunchongharn, E. Hossain, and D. I. Kim, "Resource allocation for device-to-device communications underlaying LTE-advanced networks," *IEEE Wireless Commun.*, vol. 20, no. 4, pp. 91–100, Aug. 2013.

[17] Z. Yang, N. Huang, H. Xu, Y. Pan, Y. Li and M. Chen, "Downlink resource allocation and power control for device-to-device communication underlaying cellular networks," *IEEE Commun. Lett.*, vol. 20, no. 7, pp. 1449–1452, Jul. 2016.

[18] J. Wang, D. Zhu, C. Zhao, J. Li, and M. Lei, "Resource sharing of underlaying device-to-device and uplink cellular communications," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1148–1151, Jun. 2013.

[19] S. Dominic and L. Jacob, "Distributed resource allocation for D2D communications underlaying cellular networks in time-varying environment," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 388–391, 2018.

[20] P. Jänis et al. C. Yu, K. Doppler, C. Ribeiro, C. Wijting, K. Hugl, O. Tirkkonen, and V. Koivunen, "Device-to-device communication underlaying cellular communications systems," *Int. Jour. Commun., Network and Sys. Sci.*, vol. 2, no. 3, pp. 169–178. Jun. 2009.

[21] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro and K. Hugl, "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Commun. Mag.*, vol. 47, no. 12, pp. 42–49, Dec. 2009.

[22] H. Min, W. Seo, "Reliability improvement using receive mode selection in the device-to-device uplink period underlaying cellular networks," *IEEE. Trans. Commun.*, vol. 10, no.2, pp. 413–418, Feb. 2011.

[23] S. Xu and H. Wang, "Transmission mode selection and communication establishment in the hybrid device-to-device and cellular networks," in *Prof. IEEE Int. Conf. on Ubiquitous and Future Net.*, pp. 156–161, Jul. 2012.

[24] C. Yu, O. Tirkkonen, K. Doppler and C. Ribeiro, "On the Performance of device-to-device underlay communication with simple power control," in *Proc. IEEE Vehic. Tech. Conf.*, pp. 1–5, Apr. 2009.

[25] C.-H. Yu *et al.*, "Power optimization of device-to-device communication underlaying cellular communication," in *Proc. IEEE Int. Conf. on Commun.*, pp. 1–5, June 2009.

[26] E. Zihan, K. W. Choi, and D. I. Kim, "Distributed random access scheme for collision avoidance in cellular device-to-device communication," *IEEE Trans. Wireless Commun.*, vol. 14, no. 7, pp. 3571–3585, Jul. 2015.

[27] J. P. Jänis, V. Koivunen, C. B. Ribeiro, K. Doppler and K. Hugl, *et al.*, "Interference-avoiding MIMO schemes for device-to-device radio underlaying cellular networks," *IEEE PIMRC*, pp. 2385–2389, Sept. 2009.

[28] R. Ibrahim, M. Assaad, B. Sayrac and A. Ephremides, "Overlay D2D vs. cellular communications: A stability region analysis," in *Proc. IEEE Int. Symp. on Wireless Commun. Sys.*, pp. 378–383, Aug. 2017.

[29] A. Asadi, V. Mancuso and R. Gupta, "DORE: An experimental framework to enable outband D2D relay in cellular networks," *IEEE/ACM Trans. on Networking*, vol. 25, no. 5, pp. 2930–2943, 2017.

[30] R. Chevillon, G. Andrieux, R. Négrier and J. Diouris, "Spectral and Energy Efficiency Analysis of mmWave Communications With Channel Inversion in Outband D2D Network," *IEEE Access*, vol. 6, pp. 72104–72116, 2018.

[31] "Technical specification group SA; feasibility study for proximity services (ProSe)", 3rd generation partnership project (3GPP), (Release 12)," TR 22.803 V1.0.0, Aug. 2012.

[32] A. Alnoman and A. Anpalagan, "On D2D communications for public safety applications," in *Proc. IEEE Canada Int. Humanitarian Tech. Conf.*, pp. 124–127, Jul. 2017.

[33] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Stochastic geometry study on device-to-device communication as a disaster relief solution," *IEEE Trans. on Vehic. Tech.*, vol. 65, no. 5, pp. 3005–3017, 2016.

[34] T. Castel et al., "LTE as a potential standard for public safety indoor body-to-body networks," in *Proc. IEEE Symp. on Commun. and Vehic. Tech.*, pp. 1–6, Nov. 2015.

[35] B. M. ElHalawany, R. Ruby and K. Wu, "D2D communication for enabling internet-of-things: Outage probability analysis," *IEEE Trans. on Vehic. Tech.*, vol. 68, no. 3, pp. 2332–2345, 2019.

[36] T. Perumal, S. K. Datta and C. Bonnet, "IoT device management framework for smart home scenarios," in *Proc. IEEE Glob. Conf. on Consumer Elec.*, pp. 54–55, Oct. 2015.

[37] Y. Siriwardhana, P. Porambage, M. Liyanage and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys and Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[38] D. Wu, L. Zhou and Y. Cai, "Social-aware rate based content sharing mode selection for D2D content sharing scenarios," *IEEE Trans. on Multimedia*, vol. 19, no. 11, pp. 2571–2582, 2017.

[39] Y. Liu, M. Peng, G. Shou, Y. Chen and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and internet of things," *IEEE Int. of Things Jour.*, vol. 7, no. 8, pp. 6722–6747, 2020.

[40] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation Offloading," *IEEE Commun. Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[41] X. Chen, Z. Liu, Y. Chen and Z. Li, "Mobile Edge Computing Based Task Offloading and Resource Allocation in 5G Ultra-Dense Networks," *IEEE Access*, vol. 7, pp. 184172–184182, 2019.

[42] A. u. R. Khan, M. Othman, S. A. Madani and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys and Tutorials*, vol. 16, no. 1, pp. 393–413, 2014.

[43] M. Zeng, Y. Li, K. Zhang, M. Waqas and D. Jin, "Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach," in *Proc. IEEE Int. Conf. on Commun.*, pp. 1–6, May 2018.

[44] S. Luo, X. Chen, Z. Zhou, X. Chen and W. Wu, "Incentive-aware micro computing cluster formation for cooperative fog computing," *IEEE Trans. on Wireless Commun.*, vol. 19, no. 4, pp. 2643–2657, 2020.

[45] R. Beraldi, A. Mtibaa and A. N. Mian, "CICO: A credit-based incentive mechanism for COoperative fog computing paradigms," in *Proc. IEEE Glob. Commun. Conf.*, pp. 1–7, Dec. 2018.

[46] P. Pierleoni, R. Concetti, A. Belli and L. Palma, "Amazon, Google and Microsoft solutions for IoT: Architectures and a performance comparison," *IEEE Access*, vol. 8, pp. 5455–5470, 2020.

[47] Y. Lan, X. Wang, D. Wang, Z. Liu and Y. Zhang, "Task caching, offloading, and resource allocation in D2D-aided fog computing networks," *IEEE Access*, vol. 7, pp. 104876–104891, 2019.

[48] "Technical specification group radio access network," 3rd generation partnership project (3GPP), TS 38.300, Stage 2 (Release 16), Jul. 2020 [Online].

[49] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger and W. Xu, "Connected roads of the future: Use cases, requirements, and design considerations for vehicle-to-everything communications," *IEEE Vehic. Tech. Mag.*, vol. 13, no. 3, pp. 110–123, 2018.

[50] Z. Liu, T. Peng, S. Xiang and W. Wang, "Mode selection for device-to-device (D2D) communication under LTE-Advanced networks," in *Proc. IEEE Int. Conf. on Commun.*, pp. 5563–5567, Jun. 2012.

[51] M. Jung, K. Hwang, and S. Choi, "Joint mode selection and power allocation scheme for power-efficient Device-to-Device (D2D) communication," in *Proc. IEEE Vehic. Tech. Conf.*, pp. 1–5, May 2012.

[52] S. Hakola, T. Chen, J. Lehtomaki, and T. Koskela, "Device-to-device (D2D) communication in cellular network - performance analysis of optimum and practical communication mode selection," in *Proc. IEEE Wireless Commun. and Net. Conf.*, pp. 1—6, Apr. 2010.

[53] G. Fodor *et al.*, "Design aspects of network assisted device-to-device communications," *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 170–177, Mar. 2012.

[54] Y. Luo and Y. Yang, "D2D friendly jamming and cooperative relaying for combating a full-duplex active eavesdropper," in *IEEE Proc. Int. Conf. on Commun. Tech.*, pp. 572–577, Oct. 2019.

[55] X. Lin, J. G. Andrews, A. Ghosh, and R. Ratasuk, "An overview of 3GPP device-to-device proximity services," *IEEE Commun. Mag.*, vol. 52, no. 4, pp. 40–48, Apr. 2014.

[56] M. J. Cannon, "On the design of D2D synchronization in 3GPP Release-12," in *Proc IEEE Int. Conf. on Commun.*, pp. 633–638, Jun. 2015.

[57] O. Karatalay, I. Psaromiligkos, B. Champagne and B. Pelletier, "Fast converging distributed pulse-coupled clock synchronization for half-duplex D2D communications over multipath channels," in *Proc IEEE Int. Symp. on Signal Process. and Info. Tech.*, pp. 123–128, Dec. 2018.

[58] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz, "Distributed synchronization in wireless networks," *IEEE Signal Process. Mag.*, vol. 25, no. 5, pp. 81–97, Sept. 2008.

[59] D. Tétreault-La Roche, B. Champagne, I. Psaromiligkos and B. Pelletier, "On the use of distributed synchronization in 5G device-to-device networks," *Proc. IEEE Int. Conf. on Commun.*, pp. 1938–1883, Jul. 2017.

[60] W. Sun, F. Brännström and E. G. Ström, "Network synchronization for mobile Device-to-Device systems," *IEEE Trans. on Commun.*, vol. 65, no. 3, pp. 1193–1206, Mar. 2017.

[61] O. Olabiyi, A. Annamalai and L. Qian, "Leader election algorithm for distributed ad-hoc cognitive radio networks," in *Proc. IEEE Cons. Commun. and Net. Conf.* pp. 859–863, Jan. 2012.

[62] H. Han, J. Kim, H. P. Hyuck and M. Kwon, "An effective distributed synchronization method for device-to-device communications," *Proc. IEEE Int. Conf. on Consumer Electronics*, pp. 346–347, Mar. 2017.

[63] M. A. Alvarez, U. Spagnolini, “Half-duplex scheduling in distributed synchronization,” in *Proc. IEEE Int. Conf. on Commun.*, pp. 6240–6245, Jun. 2015.

[64] M. A. Alvarez and U. Spagnolini, “Distributed time and carrier frequency synchronization for dense wireless networks,” *IEEE Trans. on Signal and Info. Process. over Networks*, vol. 4, no. 4, pp. 683–696, Dec. 2018.

[65] K. Doppler, C. B. Ribeiro and J. Kneckt, “Advances in D2D communications: Energy efficient service and device discovery radio,” in *Proc. IEEE Wireless VITAE*, pp. 1–6, Chennai, Feb. 2011.

[66] Y. Fan, L. Zhai and H. Wang, “Cost-efficient dependent task offloading for multiusers,” *IEEE Access*, vol. 7, pp. 115843–115856, 2019.

[67] C. Wang, F. R. Yu, C. Liang, Q. Chen and L. Tang, “Joint computation offloading and interference management in wireless cellular networks with mobile edge computing,” *IEEE Trans. on Vehic. Tech.*, vol. 66, no. 8, pp. 7432–7445, 2017.

[68] H. Xing, L. Liu, J. Xu and A. Nallanathan, “Joint task assignment and resource allocation for D2D-enabled mobile-edge computing,” *IEEE Trans. on Commun.*, vol. 67, no. 6, pp. 4193–4207, 2019.

[69] C. Zhao, Y. Cai, A. Liu, M. Zhao and L. Hanzo, “Mobile edge computing meets mmWave communications: Joint beamforming and resource allocation for system delay minimization,” *IEEE Trans. on Wireless Commun.*, vol. 19, no. 4, pp. 2382–2396, 2020.

[70] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne and L. Hanzo, “Energy-efficient resource allocation for latency-sensitive mobile edge computing,” *IEEE Trans. on Vehic. Tech.*, vol. 69, no. 2, pp. 2246–2262, 2020.

[71] J. M. Kim, Y. G. Kim and S. W. Chung, “Stabilizing CPU frequency and voltage for temperature-aware DVFS in mobile devices,” *IEEE Trans. on Comp.*, vol. 64, no. 1, pp. 286–292, 2015.

[72] O. Kwon, W. Jang, G. Kim, C. G. Lee, “Optimal planning of dynamic thermal management for NANS (N-App N-Screen) services” *Electronics* 7, no. 11: 311. https://doi.org/10.3390/electronics7110311

[73] O. Sahin and A. K. Coskun, “On the impacts of greedy thermal management in mobile devices,” *IEEE Embedded Sys. Let.*, vol. 7, no. 2, pp. 55–58, June 2015.

[74] A. Guchhait, “Maximum likelihood estimation of clock skew in sparse one-way packet transmissions for machine type communication applications,” in *Proc IEEE Int. Conf. on Commun.*, pp. 1–6, May 2016.

[75] O. Al-Kofahi, “Evaluating time synchronization using application-layer time-stamping,” in *Proc IEEE Wireless Commun. and Net. Conf.*, pp. 1–6, Apr. 2016.

[76] H. Wang, H. Zeng, M. Li, B. Wang and P. Wang, “Maximum likelihood estimation of clock skew in wireless sensor networks with periodical clock correction under exponential delays,” *IEEE. Trans. on Signal Process.*, vol. 65, no. 10, pp. 2714–2724, May 2017.

[77] A. K. Karthik and R. S. Blum, “Optimum full information, unlimited complexity, invariant, and minimax clock skew and offset estimators for IEEE 1588,” *IEEE Trans. on Commun.*, vol. 67, no. 5, pp. 3624–3637, May 2019.

[78] Y.-W. Hong, and A. Scaglione, “A scalable synchronization protocol for large scale sensor networks and its applications,” *IEEE Jour. on Selec. Areas in Commun.*, vol. 23, no. 5, pp. 1085–1099, May 2005.

[79] O. Karatalay, I. Psaromiligkos, B. Champagne and B. Pelletier, “Fully distributed energy-efficient synchronization for half-duplex D2D communications,” in *Proc IEEE Pers. Indoor and Mobile Commun.*, pp. 1–7, Sept. 2019.

[80] Z. Zhang, K. Long, A. V. Vasilakos and L. Hanzo, “Full-duplex wireless communications: Challenges, solutions, and future research directions”, in *Proceed. of the IEEE*, vol. 104, no. 7, pp. 1369–1409, 2016.

[81] M. A. Alvarez, B. Azari, U. Spagnolini, “Time and frequency self-synchronization in dense cooperative network,” in *Proc. Asilomar Conf. on Signals, Systems and Computers*, pp. 1811–1815 Nov. 2014.

[82] K. Manolakis and W. Xu, “Time synchronization for multi-link D2D/V2X communication,” in *Proc. IEEE Vehic. Tech. Conf.*, pp. 1–6, Sept. 2016.

[83] E. Garcia, S. Mou, Y. Cao, and D. W. Casbeer, “An event-triggered consensus approach for distributed clock synchronization,” in *Proc. IEEE American Control Conf.*, pp. 279–284, May 2017.

[84] N. Gresset and J. Letessier, “A random broadcast consensus synchronization algorithm for large scale wireless mesh networks,” in *Proc. IEEE Wireless Commun. Net. Conf.*, pp. 1573–1577, Apr. 2012.

[85] L. Song, D. Niyato, Z. Han, and E. Hossain *Wireless Device-to-Device Communications and Networks*. Cambridge Univ. Press, 2015.

[86] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Commun. Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[87] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, “Dynamic computation offloading for mobile-edge computing with energy harvesting devices,” *IEEE Jour. on Selec. Areas in Commun.*, vol. 34, no. 12, pp. 3590–3605, 2016.

[88] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues and M. Guizani, “Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay,” *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 44–51, 2018.

[89] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, “Cost-efficient workload scheduling in cloud assisted mobile edge computing,” in *IEEE Proc. Int. Symp. on Quality of Service*, pp. 1-–10. June 2017.

[90] M. Huang, W. Liu, T. Wang, A. Liu and S. Zhang, “A cloud–MEC collaborative task offloading scheme with service orchestration,” *IEEE Int. of Things Jour.*, vol. 7, no. 7, pp. 5792–5805, 2020.

[91] J. Yan, S. Bi, Y. J. Zhang and M. Tao, “Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency,” *IEEE Trans. on Wireless Commun.*, vol. 19, no. 1, pp. 235–250, 2020.

[92] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach and F. Giust, “Mobile-Edge computing architecture: the role of MEC in the internet of things,” *IEEE Consumer Electronics Mag.*, vol. 5, no. 4, pp. 84–91, 2016.

[93] X. Chen, Z. Liu, Y. Chen and Z. Li, “Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks,” *IEEE Access*, vol. 7, pp. 184172–184182, 2019.

[94] L. Liu, Z. Chang, X. Guo, and T. Ristaniemi, “Multi-objective optimization for computation offloading in mobile-edge computing,” in *IEEE Proc. Symp. Comput. Commun.*, pp. 832–837, Jul. 2017.

[95] X. Chen, L. Jiao, W. Li and X. Fu, “Efficient multi-user computation offloading for mobile-edge cloud computing,” *IEEE/ACM Trans. on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[96] N. Li, J. Martinez-Ortega and V. H. Diaz, “Distributed power control for interference-aware multi-user mobile edge computing: A game theory approach,” *IEEE Access*, vol. 6, pp. 36105–36114, 2018.

[97] B. Wu, J. Zeng, L. Ge, Y. Tang and X. Su, “A game-theoretical approach for energy-efficient resource allocation in MEC network,” in *IEEE Proc. Int. Conf. on Commun.*, pp. 1–6, May 2019.

[98] H. Hong, “From cloud computing to fog computing: Unleash the power of edge and end devices,” 2017 *IEEE Proc. Int. Conf. on Cloud Comp. Tech. and Sci.*, pp. 331–334, Dec. 2017.

[99] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar and J. H. Abawajy, “Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study,” *IEEE Access*, vol. 5, pp. 9882–9910, 2017.

[100] J. Wen, C. Ren and A. K. Sangaiah, “Energy-efficient device-to-device edge computing network: An approach offloading both traffic and computation,” *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 96–102, 2018.

[101] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein and F. H. P. Fitzek, "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166079–166108, 2019.

[102] Y. He, J. Ren, G. Yu and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. on Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, 2019.

[103] L. Pu, X. Chen, J. Xu and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Jour. on Selec. Areas in Commun.*, vol. 34, no. 12, pp. 3887–3901, 2016.

[104] B. Gao, Z. Zhou, F. Liu, F. Xu and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. on Mobile Comp.*, doi: 10.1109/TMC.2021.3064847.

[105] C. Yi, S. Huang and J. Cai, "Joint resource allocation for device-to-device communication assisted fog computing," *IEEE Trans. on Mobile Comp.*, vol. 20, no. 3, pp. 1076–1091, 2021.

[106] L. Li, L. Gu, J. Hong and S. Jiang, "Joint computation offloading and wireless resource allocation in mobile edge computing," in *IEEE Proc. Int. Conf. on Comp. and Commun.*, pp. 705–711, Dec. 2018.

[107] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[108] Z. Ning, P. Dong, X. Kong and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Int. of Things Jour.*, vol. 6, no. 3, pp. 4804–4814, June 2019.

[109] X. Meng, W. Wang and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017.

[110] S. Yu, R. Langar and X. Wang, “A D2D-multicast based computation offloading framework for interactive applications,” in *IEEE Proc. Glob. Commun. Conf.*, pp. 1–6, Dec. 2016.

[111] W. Liu, Y. Teng, M. Liu and M. Song, “Joint offloading and computation resource allocation in D2D assisted hybrid framework,” in *IEEE Proc. Annual Int. Symp. on Pers., Indoor and Mobile Radio Commun.*, pp. 1–6, Sept. 2019.

[112] T. Yang et al., ”Compact and voltage-scalable sensor for accurate thermal sensing in dynamic thermal management,” in *Proc. IEEE Int. Midwest Symp. on Circuits and Systems*, pp. 33–36, Aug. 2017.

[113] Y. G. Kim, M. Kim, J. Kong and S. W. Chung, ”An adaptive thermal management framework for heterogeneous multi-core processors,” *IEEE Trans. on Computers*, vol. 69, no. 6, pp. 894–906, 2020.

[114] A. Prakash, H. Amrouch, M. Shafique, T. Mitra and J. Henkel, “Improving mobile gaming performance through cooperative CPU-GPU thermal management,” *Proc ACM/EDAC/IEEE Design Automation Conf.*, pp. 1–6, June 2016.

[115] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, U.K.: Cambridge Univ. Press, 2004.

[116] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer-Verlag, 2006.

[117] N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma,. “Finding approximate local minima for nonconvex optimization in Linear Time, in *STOC*, 2017, http://arxiv.org/abs/1611.01146

[118] H. A. Le Thi, T. Pham Dinh, “DC programming in communication systems: Challenging problems and methods,” *Vietnam Jour. Comp. Sci.* 1, 15–28, 2014, https://doi.org/10.1007/s40595-013-0010-5.

[119] A. S. Strekalovsky, “On nonconvex optimization problems with D.C. equality and inequality constraints”, *IFAC-Papers OnLine*, vol. 51, issue 32, pp. 895–900, ISSN 2405–8963, 2018.

[120] R. Horst, P. M. Pardalos, and N. V. Thoai, *Introduction to global optimization*, Springer New York, NY, 2000.

[121] B. K. Sriperumbudur, and G. R. G. Lanckriet, "On the Convergence of the Concave-Convex Procedure," in *Proc. Neural Inf. Proc. Sys.*, vol. 22, pp. 1759–1767, 2009.

[122] A. A. Ahmadi, G. Hall, "DC decomposition of nonconvex polynomials with algebraic techniques," *Math. Program.*, 169, 69–94, 2018.

[123] Tao, P.D., Souad E.B., "Duality in dc (difference of convex functions) optimization. subgradient methods," *Trends in Mathematical Optimization*, pp. 277–293, Springer (1988).

[124] A. Charnes, and W. W. Cooper, "Chance-constrained programming," *Management Science*, vol. 6, no. 1 pp. 73–79. 1959.

[125] O. Karatalay, I. Psaromiligkos, B. Champagne and B. Pelletier, "A distributed pulse-based synchronization protocol for half-duplex D2D communications," *IEEE Open Jour. of the Commun. Soc.*, vol. 2, pp. 245–261, 2021.

[126] Y. Wu, Q. Chaudhari and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 124–138, Jan. 2011.

[127] P. Martí, J. Torres-Martínez, C. X. Rosero, M. Velasco, J. Miret and M. Castilla, "Analysis of the effect of clock drifts on frequency regulation and power sharing in inverter-based islanded microgrids," *IEEE Trans. on Power Electronics*, vol. 33, no. 12, pp. 10363–10379, Dec. 2018.

[128] "Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception," 3rd Generation Partnership Project (3GPP), TR 36.101, Mar. 2020. [Online].

[129] D. Chu, "Polyphase codes with good periodic correlation properties," *IEEE Trans. Inf. Theory*, vol. 18, no. 4, pp. 531–532, Jul. 1972.

[130] M. M. Mansour, "Optimized architecture for computing Zadoff-Chu sequences with application to LTE," in *Proc. IEEE Glob. Commun. Conf.*, pp. 1–6, Nov. 2009.

[131] "Mobile and wireless communications enablers for the twenty-twenty information society (METIS)," Deliverable D1.4 METIS Channel Models, ICT-317669-METIS/D1.4, Feb. 2015. [Online] Available: https://metis2020.com/wp-content/uploads/deliverables/METIS_D1.4_v1.0.pdf

[132] O. Karatalay, I. Psaromiligkos, and B. Champagne, "Energy-efficient D2D-aided fog computing under probabilistic time constraints," in *Proc. IEEE Glob. Commun. Conf.*, pp. 01–07, Dec. 2021.

[133] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. on Net.*, vol. 27, no. 1, pp. 85–97, 2019.

[134] J. Kim and J. W. Lee, "The mode selection scheme for group device-to-device communications underlay cellular networks," in *Proc. IEEE Int. Conf. on Info. and Commun. Tech.*, pp. 259–260, Oct. 2014.

[135] L. P. Qian, Y. Wu, L. Huang and W. Zhang, "Optimal user association and resource allocation for device-to-device communications underlaying cellular networks," in *Proc. IEEE Int. Conf. on Comp., Net. and Commun.*, pp. 1–6, Feb. 2016.

[136] J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proc. IEEE Int. Conf. on Neural Net.*, pp. 1942–1948, Nov. 1995.

[137] J.R. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, USA, MIT Press, 1992.

[138] N. Vucic, S. Shi and M. Schubert, "DC programming approach for resource allocation in wireless networks," in *Proc. Int. Symp. on Modeling and Opt. in Mob., Ad Hoc, and Wireless Net.*, pp. 380–386, May 2010.

[139] Y. Dai, M. Sheng, K. Zhao, L. Liu, J. Liu and J. Li, "Interference-aware resource allocation for D2D underlaid cellular network using SCMA: A hypergraph approach," in *Proc. IEEE Wireless Commun. and Network Conf.*, pp. 1–6, Apr. 2016.

[140] P. A. Gagniuc, *Markov Chains, From Theory to Implementation and Experimentation.* USA, NJ: John Wiley and Sons, 2017.