Energy-Aware Cooperative Computation in Mobile Devices

Ajita Singh, Yuxuan Xing, Hulya Seferoglu
ECE Department, University of Illinois at Chicago
asingh64@uic.edu, yxing7@uic.edu, hulya@uic.edu

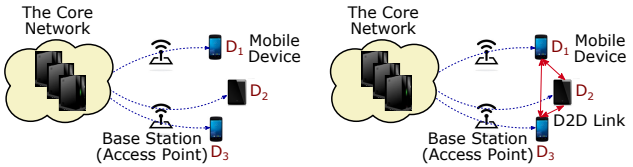
***Abstract*—New data intensive applications, which are continuously emerging in daily routines of mobile devices, significantly increase the demand for data, and pose a challenge for current wireless networks due to scarce resources. Although bandwidth is traditionally considered as the primary scarce resource in wireless networks, the developments in communication theory shifts the focus from bandwidth to other scarce resources including processing power and energy. Especially, in device-to-device networks, where data rates are increasing rapidly, processing power and energy are becoming the primary bottlenecks of the network. Thus, it is crucial to develop new networking mechanisms by taking into account the processing power and energy as bottlenecks. In this paper, we develop an energy-aware cooperative computation framework for mobile devices. In this setup, a group of cooperative mobile devices, within proximity of each other, (i) use their cellular or Wi-Fi (802.11) links as their primary networking interfaces, and (ii) exploit their device-to-device connections (*e.g.,* Wi-Fi Direct) to overcome processing power and energy bottlenecks. We evaluate our energy-aware cooperative computation framework on a testbed consisting of smartphones and tablets, and we show that it brings significant performance benefits.**

I. Introduction

The dramatic increase in mobile applications and the number of devices demanding for wireless connectivity poses a challenge in today's wireless networks [1], [2], and calls for new networking mechanisms.

One of the promising solutions to address the increasing data and connectivity demand is Device-to-Device (D2D) networking. As illustrated in Fig. 1(a), the default operation in current wireless networks is to connect each device to the Internet via its cellular or Wi-Fi interface. The D2D connectivity idea, which is illustrated in Fig. 1(b), breaks this assumption: it advocates that two or more devices in close proximity can be directly connected, *i.e.,* without traversing through auxiliary devices such as a base station or access point. D2D networking, that can be formed by exploiting D2D connections such as Wi-Fi Direct [3], is a promising solution to the ever increasing number and diversity of applications and devices. In this context, it is crucial to identify scarce resources and effectively utilize them to fully exploit the potential of D2D networking.

Although bandwidth is traditionally considered as the primary scarce resource in wireless networks, in D2D networks, thanks to close proximity among devices and the developments in communication theory, the main bottleneck shifts from bandwidth to other scarce resources including processing power and energy. Next, we present our pilot study demonstrating that processing power can be more pronounced as a bottleneck than bandwidth in D2D networks.



(a) The default operation (b) D2D connectivity

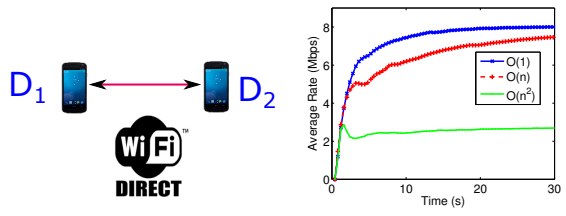
Fig. 1. (a) The default operation for the Internet connection. (b) D2D connectivity: two or more mobile devices can be connected directly, *i.e.,* without traversing through the core network, if they are in close proximity by exploiting local area connections such as Wi-Fi Direct.

Pilot Study: We developed a prototype for this pilot study as shown in Fig. 2(a), where a mobile device D_2 receives data from another device D_1 over a Wi-Fi Direct link. We use Android operating system [4] based Nexus 7 tablets [5] as mobile devices. In this experiment, after receiving the packets, the mobile device D_2 performs operations with complexities of $\mathcal{O}(1)$, $\mathcal{O}(n)$, and $\mathcal{O}(n^2)$ above the transport layer (TCP), where n is the packet size, and the operations we perform are counting the bytes in the packets. In particular, $\mathcal{O}(1)$, $\mathcal{O}(n)$, and $\mathcal{O}(n^2)$ correspond to (i) no counting, (ii) counting every byte in a packet once, and (iii) counting every byte in a packet n times, respectively. We demonstrate in Fig. 2(c) the received rate at the mobile device D_2 (note that this is the rate we measure at the mobile device D_2 after performing computations) versus time. This figure demonstrates that the received rate decreases significantly when the complexity increases. □

Our pilot study shows that even if actual bandwidth is high and not a bottleneck, processing power could become a bottleneck in D2D networks. Similar observations can be made for the energy bottleneck as detailed in our technical report [6]. Furthermore, with the advances in communication theory, *e.g.,* millimeter wave communication [7], it is expected that data rates among devices in close proximity will increase significantly, which will make processing power and energy more pronounced as bottlenecks. However, existing applications, algorithms, and protocols are mainly designed by assuming that bandwidth is the main bottleneck. Thus, it is crucial to develop new networking mechanisms when bandwidth is not the primary bottleneck, but processing power and energy are.

Thus, in this paper, our goal is to create group of devices that help each other cooperatively by exploiting high rate D2D

ISBN 978-3-901882-83-8 © 2016 IFIP



(a) Setup (b) Rate vs Time

Fig. 2. **Pilot Study:** (a) Setup: Data is transmitted from mobile device D_1 to another mobile device D_2. In this setup, the mobile devices are Android operating system (OS) [4] based Nexus 7 tablets [5]. The specific version of the Anroid OS is Android Lollipop 5.1.1. The devices have 16GB storage, 2GB RAM, Qualcomm Snapdragon S4 Pro, 1.5GHz CPU, and Adreno 320, 400MHz GPU. Packet size is $500B$. (b) Transmission rate versus time for different computational complexities at the receiver side. Note that we present the rate that we measure at the mobile device after performing the computations. The presented rates are the averages over 10 seeds.

connections to overcome the processing power and energy bottlenecks. The next example demonstrates our approach.

Example 1: Let us consider Fig. 1(a) again, where device D_1 would like to receive a file from a remote resource via its cellular or Wi-Fi connection. Assume that the cellular (or Wi-Fi) rates of all devices are 1Mbps, but device D_1 can receive data with 500kbps rate due to processing power bottleneck, *i.e.,* device D_1 has limited processing power (similar to our pilot study we presented earlier). In a traditional system, D_1 will behave as a single end point, so its receiving rate will be limited to 500kbps. On the other hand, if devices D_1, D_2, and D_3 will behave as a group and cooperate, then devices D_2 and D_3 can also receive and process 500kbps portions of data, and transmit the processed data to device D_1 over D2D connections. This increases the receiving rate of device D_1 to 1.5Mbps from 500kbps, which is a significant improvement.

This example could be extended for scenarios when energy (battery of mobile devices) is limited. For example, if device D_2's battery level is too low, its participation to the group activity should be limited. □

Application Areas. The scenario in the above motivating example could arise in different practical applications from health, education, entertainment, and transportation systems. The following are some example applications. *Health:* A person may own a number of health monitoring devices (activity monitoring, hearth monitoring, etc.) which may need updates from the core network. These updates - potentially coded for error correction, compression, and security reasons - should be processed (decoded) by these devices. Processing takes time, which may lead to late reaction to the update (which may require timely response) and energy consumption. On the other hand, by grouping mobile devices, the person's smartphone or tablet could receive the update, process, and pass the processed data to the health monitoring devices via high rate D2D links. *Education & Entertainment:* A group of students may want to watch the video of a lecture from an online education system (or an entertainment video) while sitting together and using several mobile devices. In this setup, one of the devices can download a base layer of a video and decode, while the other devices could download enhancement layers and decode. The decoded video layers could be exchanged among these mobile devices via high rate D2D links. As in the motivating example, if one device's download and decoding rate is limited to 500kbps, it could be improved to 1.5Mbps with the help of other devices.

Note that the processing overhead in these applications could be due to any computationally intensive task related to data transmission. For example, for video transmission applications, H.264/AVC decoders introduce higher computational complexity when higher quality guarantees are needed [8], [9]. Another example could be network coding; for example, data could be network coded at the source to improve throughput, error correction, packet randomization potential of network coding [10]. However, most of the network coding schemes introduce high computational complexity at the receiver side; $\mathcal{O}(n^3)$, [11], [12], which limits the transmission rate. Encryption could be another example that introduces processing overhead [13].

Thus, there exist several applications and scenarios where bandwidth and energy could be bottlenecks, while bandwidth is not the bottleneck. This makes our approach demonstrated in Example 1 crucial. In particular, in this paper, we develop an *energy-aware cooperative computation* framework for mobile devices. In this setup, a group of cooperative mobile devices, within proximity of each other, (i) use their cellular or Wi-Fi (802.11) links as their primary networking interfaces, and (ii) exploit their D2D connections (Wi-Fi Direct) for cooperative computation. Our approach is grounded on a network utility maximization (NUM) formulation of the problem and its solution [14]. The solution decomposes into several parts with an intuitive interpretation, such as flow control, computation control, energy control, and cooperation & scheduling. Based on the structure of the decomposed solution, we develop a stochastic algorithm; *energy-aware cooperative computation*.[1] The following are the key contributions of this work:

- We consider a group of cooperative mobile devices within proximity of each other. In this scenario, we first investigate the impact of processing power to transmission rate. Then, we develop an energy-aware cooperative computation model, where devices depending on their energy constraints could cooperate to get benefit of aggregate processing power in a group of cooperative devices.
- We characterize our problem in a NUM framework by taking into account processing power, energy, and bandwidth constraints. We solve the NUM problem, and use the solution to develop our stochastic algorithm; *energy-aware cooperative computation (EaCC)*. We show that EaCC provides stability and optimality guarantees.
- An integral part of our work is to understand the performance of EaCC in practice. Towards this goal, we

[1]Note that our work focuses on cooperative resource utilization in mobile devices. In this sense, our work is complementary to and synergistic with: (i) creating incentive mechanisms in D2D networks, and (ii) providing privacy and security for D2D users [15], [16]. Looking into the future, it is very likely that our proposed work on the design, analysis, and implementation of cooperative resource utilization is gracefully combined with the work on creating incentives and providing privacy and security.

develop a testbed consisting of Nexus 5 smartphones and Nexus 7 tablets. All devices uses Android 5.1.1 as their operation systems. We implement *EaCC* in this testbed, and evaluate it. The experimental results show that our algorithm brings significant performance benefits.

The structure of the rest of the paper is as follows. Section II presents related work. Section III gives an overview of the system model. Section IV presents the NUM formulation of our cooperative computation scheme. Section V presents our stochastic algorithm; *EaCC*. Section VI evaluates the performance of our scheme in a real testbed. Section VII concludes the paper.

II. Related Work

This work combines ideas from D2D networking, network utility maximization, and stochastic network control.

The idea of D2D networking is very promising to efficiently utilize resources, so it has found several applications in the literature. In particular, D2D connections are often used to form cooperative groups for data streaming applications, and for the purpose of (i) content dissemination among mobile devices [17], [18], (ii) cooperative video streaming over mobile devices [19], [20], [21], [22], and (iii) creating multiple paths and providing better connectivity by using multiple interfaces simultaneously [23], [24]. As compared to this line of work, we investigate the impact of processing power and energy in D2D networks, and develop mechanisms to effectively utilize these scarce resources.

D2D networking is often used for the purpose of offloading cellular networks. For example, previous work [25], [26], [17] disseminate the content to mobile devices by taking advantage of D2D connections to relive the load on cellular networks. Instead of offloading cellular networks, our goal is to create energy-aware cooperation framework to overcome the processing power and energy bottlenecks of mobile devices.

There is an increasing interest in computing using mobile devices by exploiting connectivity among mobile devices [27]. This approach suggests that if devices in close proximity are capable of processing tasks cooperatively, then these devices could be used together to process a task as it is a cheaper alternative to remote clouds. This approach, sparking a lot of interest, led to some very interesting work in the area [28], [29], [30]. As compared to this line of work, we focus on processing power and energy bottlenecks in mobile devices and address the problem by developing energy-aware cooperative computation mechanism.

An integral part of our proposed work in this task is to develop efficient resource allocation mechanisms. In that sense, our approach is similar to the line of work emerged after the pioneering work in [31], [32], [33]. However, our focus is on energy-aware cooperative computation.

III. System Model

We consider a cooperative system setup with N mobile devices, where $\mathcal{N}$ is the set of the mobile devices. Our system model for three nodes are illustrated in Fig. 3(a). The source in Fig. 3(a) represents the core network and base stations (access points). This kind of abstraction helps us focus on the bottlenecks of the system; processing power, energy of mobile devices, and downlink/uplink data rates. In this setup, mobile devices communicate via D2D connections such as Wi-Fi Direct, while the source communicates with mobile devices via cellular or Wi-Fi links. We consider in our analysis that time is slotted and t refers to the beginning of slot t.

Connecting Devices Together: The total flow rate towards device n in Fig. 3(a) (as also explained in Fig. 3(b)) is $\sum_{k\in\mathcal{N}} x_{n,k}(t)$, where $x_{n,n}(t)$ is the transmission rate of the packets from the source towards device n, and these packets will be used by device n. Note that $x_{n,k}(t)$ is the transmission rate of the packets from the source towards device n, and these packets will be processed by device n and forwarded to device k. On the other hand, $y_n(t)$ is the total flow rates targeting device n as demonstrated in Fig. 3(b). The source constructs a queue $S_n(t)$ for the packets that will be transmitted to the mobile device n. The evolution of $S_n(t)$ based on $y_n(t)$ and $x_{k,n}(t)$ is expressed as

$$S_n(t+1) \leq \max[S_n(t) - \sum_{k\in\mathcal{N}} x_{k,n}(t), 0] + y_n(t), \qquad (1)$$

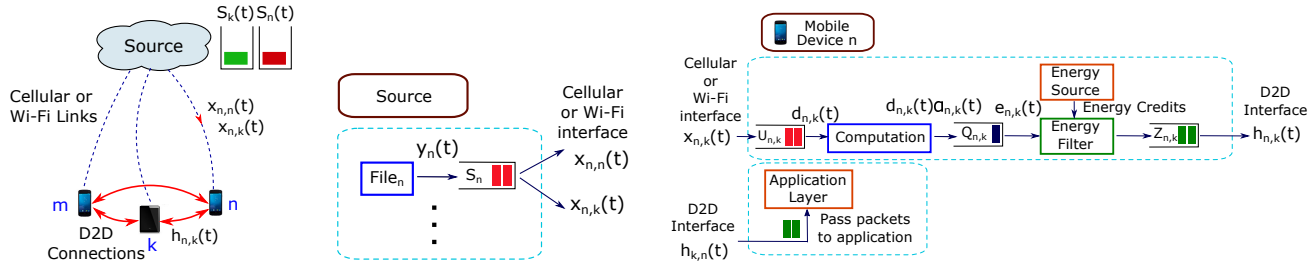
where the inequality comes from the fact that there may be less than $y_n(t)$ packets arriving into $S_n(t)$ at time t in practice (*e.g.,* in real time applications, the number of available packets for transmission could be limited).

The flow rate $y_n(t)$ is coupled with a utility function $g_n(y_n(t))$, which we assume to be strictly concave function of $y_n(t)$. This requirement is necessary to ensure stability and utility optimality of our algorithms. The ultimate goal in our resource allocation problem is to determine the flow rates; $y_n(t)$ which maximize the sum utility $\sum_{n\in\mathcal{N}} g_n(y_n(t))$.

Finally, flow rate over D2D connection between device n and k is $h_{n,k}(t)$, $k \neq n$. Note that $h_{n,k}(t)$ is to help node k using node n as a processing device.

Inside a Mobile Device: In each device, we develop different modules depending on where data is arriving from (as shown in Fig. 3(c)); *i.e.,* from the source via cellular or Wi-Fi interface, or other mobile devices via D2D interfaces.

When data is arriving from a D2D interface, it is directly passed to the application layer, as this data is already processed by another device. On the other hand, when data is arriving from the source via cellular or Wi-Fi interfaces, packets go through multiple queues as shown in Fig. 3(c), where $U_{n,k}$, $Q_{n,k}$, and $Z_{n,k}$ represent three different queues constructed at mobile device n for the purpose of helping node k. Incoming packets via cellular or Wi-Fi links are stored in $U_{n,k}$, which then forwards the packets to *computation* block with rate $d_{n,k}(t)$. The computation block processes the packets, and pass them to queue $Q_{n,k}$. Note that the output rate from computation block is $d_{n,k}(t)\alpha_{n,k}(t)$, where $\alpha_{n,k}(t)$ is a positive real value. This value captures any possible rate changes at the computation block, *i.e.,* $\alpha_{n,k}(t)$ is a rate shaper. For example, if the computation block is H.264/AVC decoder or transcoder, we expect that the rate at the output of the computation block should be higher than the input. Thus, $\alpha_{n,k}(t)$ captures this fact for any n, k, t. On the other hand, if there is no rate change



(a) System Model (b) Building blocks of the source. (c) Building blocks of mobile device n

Fig. 3. (a) System model for the scenario of three devices; n, m, k. The source in this model represents the core network and base stations (access points). (b) Building blocks of the source. File$_n$, $\forall n$ is read and inserted in the buffer $S_n(t)$, and packets are transmitted from $S_n(t)$. $x_{n,k}(t)$ is the transmission rate of the packets from the source towards device n, and these packets will be processed by device n and forwarded to device k. (c) Building blocks of mobile device n. If packets are received from the source via cellular and Wi-Fi interfaces, then they go to the computation and energy control blocks. If packets are received from other mobile devices via D2D interface, they are directly passed to the application.

after the processing, then $\alpha_{n,k}(t) = 1$.

The processed (and possibly rate shaped) packets are queued at $Q_{n,k}(t)$ and passed to *energy filter*. The energy filter is coupled to the energy source, which determines the amount of energy that can be spent to support the tasks at each slot. The amount of energy is determined according to *energy credits*. In particular, the energy source, depending on the battery level as well as the estimate on the expected battery consumption in the near future, calculates the number of packets that can be supported by the mobile device, and the same number of energy credits enter the energy filter. (Note that both energy filter, energy source, and energy credits are not real, but virtual entities, so they can be modeled by using a few counters in practice.) Thus, at each transmission slot, packets are transmitted from $Q_{n,k}(t)$ to $Z_{n,k}(t)$ with rate $e_{n,k}(t)$ if there exist energy credits in the energy filter. Finally, packets from $Z_{n,k}(t)$ are transmitted to application if device n is the destination of the data (*i.e.,* $n = k$), or they are transmitted to the original destination via D2D interface with rate $h_{n,k}(t)$.

The computation and energy filter blocks in Fig. 3(c) model the processing and energy bottlenecks of the mobile device, respectively. If packets in $U_{n,k}$ increases too much, this means that the computation block, hence processing power, is the bottleneck, so node n should not receive much packets from the source. Similarly, if $Q_{n,k}$ increases too much, this means that energy filter is the bottleneck, so again node n should not receive much packets. Note that there could be also some buildup in $Z_{n,k}$ if the link between node n and k is the bottleneck of the system, and it should be taken into account when the energy-aware cooperative computation framework is developed.

Also, it is crucial in our system model to put energy filter after the computation block, because if device n will help device k, the actual amount of packets that are supposed to be transmitted are the processed packets, which will cause energy consumption (*i.e.,* not the packets before processing).

Based on the above intuitions and observations, we will develop our resource allocation problem and algorithm in the next sections. The evolution of the queues $U_{n,k}(t)$, $Q_{n,k}(t)$, and $Z_{n,k}(t)$ are provided in Table I.

Links: In our system model, we consider two scenarios: (i) cellular + Wi-Fi Direct, and (ii) Wi-Fi + Wi-Fi Direct.

TABLE I
EVOLUTION OF QUEUES $U_{n,k}(t)$, $Q_{n,k}(t)$, AND $Z_{n,k}(t)$.

$U_{n,k}(t+1) \leq \max[U_{n,k}(t) - d_{n,k}(t), 0] + x_{n,k}(t)$
$Q_{n,k}(t+1) \leq \max[Q_{n,k}(t) - e_{n,k}(t), 0] + d_{n,k}(t)\alpha_{n,k}(t)$
$Z_{n,k}(t+1) \leq \max[Z_{n,k}(t) - h_{n,k}(t), 0] + e_{n,k}(t)$

In both cases, the D2D links between mobile devices are Wi-Direct. In the first case, *i.e.,* in cellular + Wi-Fi Direct, the links between the source and mobile devices are cellular, while they are Wi-Fi in the second case, *i.e.,* in Wi-Fi + Wi-Fi Direct. These two scenarios are different from each other, because in the first scenario, cellular and Wi-Fi Direct links could operate simultaneously as they use different parts of the spectrum. On the other hand, in the second scenario, both Wi-Fi and Wi-Fi Direct use the same spectrum, so they time share the available resources. Our model and energy-aware cooperative computation framework are designed to operate in both scenarios. Next, we provide details about our link models.[2]

In the system model in Fig. 3(a), each mobile device $n \in \mathcal{N}$ is connected to the Internet via its cellular or Wi-Fi link. At slot t, $\boldsymbol{C}^s(t)$ is the channel state vector of these links, where $\boldsymbol{C}^s(t) = \{C_1^s(t), ..., C_n^s(t), ..., C_N^s(t)\}$. We assume that $C_n^s(t)$ is the state of the link between the source and mobile device n, and it takes "ON" and "OFF" values depending on the state of the channel. Without loss of generality, if mobile device n does not have Internet connection, then $C_n^s(t)$ is always at "OFF" state, which means there is no cellular or Wi-Fi connection.

Since we consider that mobile devices are in close proximity and transmission range, they form a fully connected clique topology. At slot t, $\boldsymbol{C}^w(t)$ is the channel state vector of the D2D links, where $\boldsymbol{C}^w(t) = \{C_{1,2}^w(t), ..., C_{n,k}^w(t), ..., C_{N-1,N}^w(t)\}$. We assume that $C_{n,k}^w(t)$ is the state of the D2D link between node n and k.

We consider protocol model in our formulations [34], where each mobile device can either transmit or receive at the same time at the same frequency. Assuming that $\boldsymbol{C}(t) =$

[2]Note that the link models described in this section provide a guideline in our algorithm development and basis in our theoretical analysis. However, in Section VI, we relax the link model assumptions we made in this section, and evaluate our algorithms on real devices and using real links.

$\{\boldsymbol{C}^s(t), \boldsymbol{C}^w(t)\}$ is the channel state vector of the system including both the links between the source and mobile devices as well as among mobile devices, $\Gamma_{\boldsymbol{C}(t)}$ denotes the set of the link transmission rates feasible at time slot t depending on our protocol model. In particular, for cellular + Wi-Fi Direct setup, $\Gamma_{\boldsymbol{C}(t)}$ is the set that allows more links to operate at the same time, while for the Wi-Fi + Wi-Fi Direct setup, $\Gamma_{\boldsymbol{C}(t)}$ is a more limited set due to the interference among the links.

IV. Problem Formulation

In this section, we characterize the stability region of the energy-aware cooperative computation problem, and formulate network utility maximization (NUM) framework. The solution of the NUM framework provides us insights for developing the stochastic control algorithms in the next section.[3]

A. Stability Region

We provide the stability region of the cooperative computation system for both cellular + Wi-Fi Direct and W-Fi + Wi-Fi Direct setups. First, the flow conservation constraint at the source should be $y_n \leq \sum_{k\in\mathcal{N}} x_{k,n}$ to stabilize the system. This constraint requires that the total outgoing rate from the source, *i.e.,* $\sum_{k\in\mathcal{N}} x_{k,n}$ should be larger than the generated rate y_n.

Furthermore, the following flow conservation constraints inside a mobile device should be satisfied for stability; $x_{n,k} \leq d_{n,k}$, $d_{n,k}\alpha_{n,k} \leq e_{n,k}$, and $e_{n,k} \leq h_{n,k}$. These constraints are necessary for the stability of queues $U_{n,k}$, $Q_{n,k}$, and $Z_{n,k}$, respectively. Finally, the transmission rates over the links should be feasible, *i.e.,* $\{x_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N}, k\in\mathcal{N}} \in \Gamma_{\boldsymbol{C}}$.

Thus, we define the stability region as $\Lambda = \{\{y_n, x_{n,k}, d_{n,k}, e_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{N}} \mid y_n,\ x_{n,k},\ d_{n,k},\ e_{n,k},\ h_{n,k} \geq 0,\ \forall n \in \mathcal{N}, k \in \mathcal{N},\ y_n \leq \sum_{k\in\mathcal{N}} x_{k,n},\ x_{n,k} \leq d_{n,k},\ d_{n,k}\alpha_{n,k} \leq e_{n,k},\ e_{n,k} \leq h_{n,k},\ \{x_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{N}} \in \Gamma_{\boldsymbol{C}}\}$.

B. NUM Formulation

Now, we characterize our NUM problem.

$$\max_{\boldsymbol{y}} \sum_{n\in\mathcal{N}} g_n(y_n)$$
$$\text{s.t. } y_n, x_{n,k}, d_{n,k}, e_{n,k}, h_{n,k} \in \Lambda,\ \forall n \in \mathcal{N}, k \in \mathcal{N} \quad (2)$$

The objective of the NUM problem in (2) is to determine $y_n, x_{n,k}, d_{n,k}, e_{n,k}, h_{n,k}$ for $\forall n \in \mathcal{N}, k \in \mathcal{N}$ which maximize the total utility $\sum_{n\in\mathcal{N}} g_n(y_n)$.

C. NUM Solution

Lagrangian relaxation of the flow conservation constraints that characterize the stability region Λ gives the following Lagrange function:

$$L = \sum_{n\in\mathcal{N}} g_n(y_n) - \sum_{n\in\mathcal{N}} s_n(y_n - \sum_{k\in\mathcal{N}} x_{k,n}) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} u_{n,k}(x_{n,k} - d_{n,k}) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} q_{n,k}(d_{n,k}\alpha_{n,k} - e_{n,k}) - \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} z_{n,k}(e_{n,k} - h_{n,k}) \quad (3)$$

where s_n, $u_{n,k}$, $q_{n,k}$, and $z_{n,k}$ are the Lagrange multipliers. Note that we will convert these Lagrange multipliers to queues S_n, $U_{n,k}$, $Q_{n,k}$, and $Z_{n,k}$ when we design our stochastic algorithm in the next section.

The Lagrange function in (3) is decomposed into sub-problems such as flow, computation, and energy controls as well as cooperation and scheduling. The solutions of (3) for y_n, $d_{n,k}$, $e_{n,k}$, $x_{n,k}$, and $h_{n,k}$ are expressed as:

$$\underline{\text{Flow control:}}\ \max_{\boldsymbol{y}} \sum_{n\in\mathcal{N}} (g_n(y_n) - y_n s_n) \quad (4)$$

$$\underline{\text{Computation control:}}\ \max_{\boldsymbol{d}} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} d_{n,k}(u_{n,k} - q_{n,k}\alpha_{n,k}) \quad (5)$$

$$\underline{\text{Energy control:}}\ \max_{\boldsymbol{e}} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} e_{n,k}(q_{n,k} - z_{n,k}) \quad (6)$$

$$\underline{\text{Cooperation \& Scheduling:}}$$
$$\max_{\boldsymbol{x},h} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{N}} [x_{n,k}(s_k - u_{n,k}) + z_{n,k}h_{n,k}]$$
$$\text{s.t. } \{x_{n,k}, h_{n,k}\}_{\forall n\in\mathcal{N},k\in\mathcal{N}} \in \Gamma_{\boldsymbol{C}} \quad (7)$$

Next, we design a stochastic algorithm; energy-aware cooperative computation inspired by the NUM solutions in (4), (5), (6), (7).

V. Energy-Aware Cooperative Computation

Now, we provide our energy-aware cooperative computation algorithm which includes *flow control*, *computation control*, *energy control*, and *cooperation & scheduling*.

Energy-Aware Cooperative Computation (EaCC):

- *Flow Control:* At every time slot t, $y_n(t)$ is determined by maximizing $\max_{\boldsymbol{y}} [Mg_n(y_n(t)) - S_n(t)y_n(t)]$ subject to $y_n(t) \leq R_n^{\max}$, where $R_n^{\max}$ is a positive constant larger than the transmission rate from the source, and M is a large positive constant. Note that $S_n(t)$ is the queue size at the source of flow and stores packets that are supposed to be transmitted to mobile device n. After $y_n(t)$ is determined, $y_n(t)$ packets are inserted in queue $S_n(t)$ (as illustrated in Fig. 3(a)).
- *Computation Control:* At every time slot t, the computation control algorithm at device n determines $d_{n,k}(t)$ by optimizing

$$\max_{\boldsymbol{d}} \sum_{k\in\mathcal{N}} d_{n,k}(t)[U_{n,k}(t) - Q_{n,k}(t)\alpha_{n,k}(t)]$$
$$\text{s.t. } \sum_{k\in\mathcal{N}} d_{n,k}(t) \leq D_n^{\max} \quad (8)$$

[3]Note that NUM optimizes the average values of the parameters that are defined in Section III. By abuse of notation, we use a variable, *e.g.,* ϕ as the average value of $\phi(t)$ in our NUM formulation if both ϕ and $\phi(t)$ refers to the same parameter.

where $D_n^{\max}$ is a positive constant larger than the processing rate of the computation block in device n dedicated to help device k. The interpretation of (8) is that at every time slot t, $d_{n,k^*} = D_n^{\max}$ packets are passed to the computation block (in Fig. 3(b)) if $U_{n,k^*}(t) - Q_{n,k^*}(t) > 0$, where k^* is the mobile device that maximizes (8). Otherwise, no packets are sent to the computation block. The packets that are being processed by the computation block are passed to $Q_{n,k}(t)$. Note that some computation blocks may require to receive a group of packets to be able to process them. In that case, $D_n^{\max}$ is arranged accordingly (*i.e.,* it can be increased to transfer a group of packets).

- *Energy Control:* At every time slot t, the energy control algorithm at device n determines $e_{n,k}(t)$ by optimizing

$$\max_{\boldsymbol{e}} \sum_{k\in\mathcal{N}} e_{n,k}(t)[Q_{n,k}(t) - Z_{n,k}(t)]$$
$$\text{s.t.} \sum_{k\in\mathcal{N}} e_{n,k}(t) \leq E_{n,k}^{\max} \quad (9)$$

  where $E_n^{\max}$ is a positive constant larger than the energy capacity of device n dedicated to help device k. The interpretation of (9) is that at every time slot t, $e_{n,k^*} = E_n^{\max}$ packets are passed to the energy filter (as illustrated in Fig. 3(b)) if $Q_{n,k^*}(t) - Z_{n,k^*}(t) > 0$, where k^* is the mobile device that maximizes (9). Otherwise, no packets are sent to the energy filter. The packets passing through the energy filter are inserted in $Z_{n,k}(t)$.
- *Scheduling & Cooperation:* At every time slot t, the scheduling and cooperation algorithm determines transmission rates over links, *i.e.,* $x_{n,k}(t)$ and $h_{n,k}(t)$ by maximizing

$$\max_{\boldsymbol{x},\boldsymbol{h}} \sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{N}} [x_{n,k}(t)(S_k(t) - U_{n,k}(t)) + h_{n,k}(t)Z_{n,k}(t)]$$
$$\text{s.t. } \boldsymbol{x}, \boldsymbol{h} \in \Gamma_{\boldsymbol{C}(t)} \quad (10)$$

  For cellular + Wi-Fi Direct system, (10) is decomposed into two terms: maximizing $\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{N}} x_{n,k}(t)(S_k(t) - U_{n,k}(t))$ and $\sum_{n\in\mathcal{N}} \sum_{k\in\mathcal{N}} h_{n,k}(t)Z_{n,k}(t)$, because cellular and Wi-Fi Direct transmissions operate simultaneously and transmission over one link does not affect the other. On the other hand, for Wi-Fi + Wi-Fi Direct setup, the joint optimization in (10) should be solved.
  Note that transmissions over all links are unicast transmissions in our work, where unicast is dominantly used in practice over cellular, Wi-Fi, and Wi-Fi Direct links. Also, it is straightforward to extend our framework for broadcast transmissions.

Theorem 1: If channel states are i.i.d. over time slots, and the arrival rates $E[y_n(t)] = A_n, \forall n \in \mathcal{N}$ are interior of the stability region Λ, then energy-aware cooperative computation stabilizes the network and the total average queue sizes are bounded.

Furthermore, if the channel states are i.i.d. over time slots, and the traffic arrival rates are controlled by the flow control algorithm of energy-aware cooperative computation, then the admitted flow rates converge to the utility optimal operating point with increasing M.

Proof: The proof is provided in [6]. ■

Our energy-aware cooperative computation framework has several advantages: (i) distributed, (ii) takes into account scarce resources such as processing power and energy in addition to bandwidth to make control decisions, and (iii) utilizes available resources; processing power, energy, and bandwidth in a utility optimal manner. Theorem 1 shows the theoretical performance guarantees of our framework, while we focus on its performance in a practical setup in the next section.

VI. Performance Evaluation

In this section, we evaluate our *energy-aware cooperative computation* (EaCC) scheme using a testbed that consists of Android based smartphones and tablets. The evaluation results show that our scheme significantly improves throughput as compared to (i) no-cooperation, where each device receives its content from the source without cooperating with other devices, and (ii) cooperation, where multiple mobile devices cooperate, but the cooperating devices do not do computation and energy control for other devices (mobile devices just receive packets from the source, and relay them to other mobile devices without processing and energy control). Next, we present testbed setup and results in detail.

A. Setup & Implementation Details

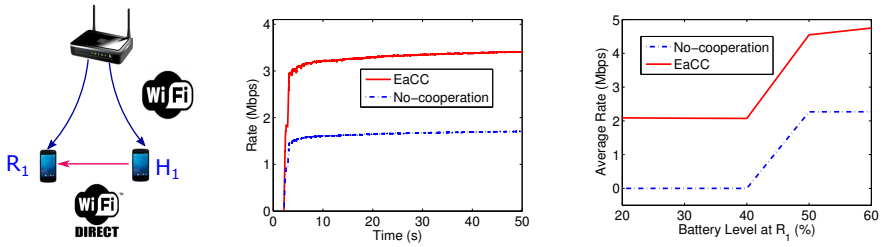
Devices: We implemented a testbed of the setup shown in Fig. 3(a) using real mobile devices, specifically Android 5.1.1 based Nexus 5 smartphones and Nexus 7 tablets.

We classify devices as (i) a source device, which acts as the source in Fig. 3(a), (ii) helper devices, which receive data from the source, process it, and transmit to other devices (receivers) to help them, and (iii) receiver devices, which receive data from both the source device and the helpers. A receiver device processes data arriving from the source, but it does not process the data arriving from helpers as the helpers send already processed data.[4] Note that a device could be both receiver and a helper device depending on the configuration.

Integration to the Protocol Stack: We implemented our energy-aware cooperative computation (EaCC) framework as a slim layer between transport and application layers. In other words, we implemented our framework on top of TCP. This kind of implementation has benefits, because (i) mobile devices do not need rooting, and (ii) our framework and codes could be easily transferred to mobile devices using other operating systems such as iOS.

Source Configuration and EaCC Implementation: We implemented the source node in Fig. 3 using a Nexus 5 smartphone. Basically, multiple files; File$_n$, File$_k$ requested by devices n and k are read by using the public java class *BufferedInputStream* according to the flow control algorithm described in Section V and shown in Fig. 3(b). The bytestream

[4]Note that we relax this assumption in our technical report [6] for multimedia applications.



(a) System Model (b) Rate vs. Time (c) Rate vs. Energy

Fig. 4. (a) System model consisting of a source device, one receiver, and one helper. Wi-Fi is used between the source and the receiver device (R_1) and the helper device (H_1), while Wi-Fi Direct is used to connect R_1 to H_1. (b) Average rate versus time for the setup shown in (a) for the case that all the devices are Android based Nexus 7 tablets. (c) Average rate versus energy level at receiver device R_1. In this setup, all devices are Android based Nexus 5 smartphones. In both (b) and (c), the average rate is calculated as the average over 10 trials (with different seeds). The computation under consideration in this experiment is $O(n^2)$, which counts the number of bytes in a packet for each byte in the packet (*i.e.,* recursive counting).

is packetized by setting each packet to $500B$, and packets are inserted into source buffers; $S_n(t)$, $S_k(t)$. We set the flow control parameters as; $M = 500$, $R_n^{\max} = 100$, and slot duration is $20msec$. We used log function as our utility function. In this setup, reading files, converting bytestream into packets, and inserting packets into the input queues are done by multiple threads, *i.e.,* a thread runs for each file; File$_n$ in Fig. 3(b).

The other set of threads at the source device make packet transmission decisions from the source device to receiver and helper devices. In particular, the source node collects $U_{n,k}(t)$ information from all mobile devices. At each time slot, the source node checks $S_k(t)-U_{n,k}(t)$, and if $S_k(t)-U_{n,k}(t) > 0$, then 100 packets are transmitted from $S_k(t)$ to the TCP socket at the source device for transmission to mobile device n.

EaCC Operation on Mobile Devices: All mobile devices (including helper or receiver+helper devices) implement all the building blocks illustrated Fig. 3(c). Multiple threads are used to make these blocks operating simultaneously.

The first thread at mobile device n receives packets that are transmitted by the source node, and inserts these packets in $U_{n,k}$.

The second thread has two tasks. First, it transfers packets from $U_{n,k}$ to $Q_{n,k}$ according to the computation control algorithm in (8), where $D_n^{\max} = 100$ packets and the slot duration is $20msec$. We set $\alpha_{n,k}(t) = 1$ in our experiments as our applications do not change the rate as explained later in this section. The second task of this thread is to actually do the computation tasks related to the application. In our experiments, the computation block counts the bytes in the packets. In particular, similar to the pilot study in the introduction, $\mathcal{O}(1)$, $\mathcal{O}(n)$, and $\mathcal{O}(n^2)$ correspond to (i) no counting, (ii) counting every byte in a packet once, and (iii) counting every byte in a packet n times, respectively.

The third thread transfers packets from $Q_{n,k}$ to $Z_{n,k}$ using the energy control algorithm in (9), where we set $E_{n,k}^{\max}$ depending on the battery level of the device. For example, if the battery level is below some threshold, $E_{n,k}^{\max}$ is limited. We evaluated different configurations in our experiments as we explain later. The slot duration is again set to $20msec$.

The final thread transfers packets from $Z_{n,k}$ to application layer if $n = k$, or transmits to node k if $n \neq k$. In the second case, *i.e.,* if $n \neq k$, the number of packets in TCP socket is checked at every time slot, where the time slot duration is $20msec$. If it is below a threshold of 500 packets, then 100 packets are removed from $Z_{n,k}$ and inserted to the TCP socket to be transmitted to node k.

When node n receives packets from node k, it directly passes the packets to the application layer as illustrated in Fig. 3(c), because these packets are the ones that are already processed by node k. If node n is both a helper and a receiver device, it runs all the threads explained above in addition to the receiving thread from node k (illustrated in Fig. 3(c)).

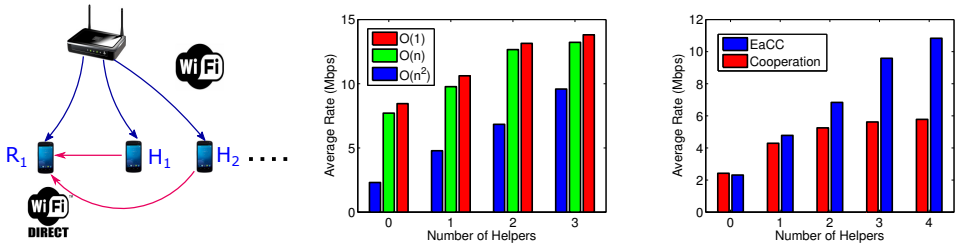
Information Exchange: Our implementation is lightweight in the sense that it limits control information exchange among mobile devices. The only control information that is transmitted in the system is $U_{n,k}$ from each mobile device to the source node. Each mobile device n collects $U_{n,k}$, $\forall k \in \mathcal{N}$, and transmits this information to the source node periodically, where we set the periods to $100msec$.

Connections: All the devices in the system including the source device, helpers, receivers, and helper+receiver devices are connected to each other using Wi-Fi Direct connections in our testbed. The source node is configured as the group owner of the Wi-Fi Direct group. We note that cooperation in this setup does not bring any benefit in terms of bandwidth utilization as all the links use the same transmission channel in a Wi-Fi Direct group. However, as we demonstrate later in this section, it brings benefit due to cooperative processing power and energy utilization, which is our main focus in this paper. Therefore, this setup (where all the devices are connected to each other using Wi-Fi Direct links) well suits to our evaluation purposes.

Test Environment: We conducted our experiments using our testbed in a lab environment where several other Wi-Fi networks were operating in the background. We located all the devices in close proximity of each other, and we have evaluated EaCC for varying levels of computational complexity, number of receivers, and number of helpers. Next, we present our evaluation results.

B. Results

We first consider a setup as shown in Fig. 4(a) which consists of a source device, one receiver (R_1), and one helper (H_1). Fig. 4(b) shows the average rate versus time graph for the setup shown in Fig. 4(a) when all three devices are Android



(a) System Model (b) Rate vs. Number of Helpers (c) Rate vs. Number of Helpers

Fig. 5. (a) System model consisting of a source device, one receiver, and multiple helpers. Wi-Fi is used between the source and the receiver device (R_1) and the helper devices ($H_1, \ldots$), while Wi-Fi Direct is used to connect the receiver devices with the helper devices. (b) EaCC: Average rate measured at receiver R_1 versus the number of helpers. (c) Average rate measured at receiver R_1 versus the number of helpers for EaCC and cooperation, when the complexity is $O(n^2)$.

based Nexus 7 tablets. The average rate is calculated as the average over 10 trials (with different seeds). The computation under consideration in this experiment is $O(n^2)$, which counts the number of bytes in a packet n times, n is the packet size. As can be seen, if there is no cooperation, the rate measured at R_1 is on the order of 1.5Mbps. On the other hand, EaCC increases the rate to almost 3Mbps. This means that helper device H_1 helps the receiver device R_1 process the packets in EaCC. In this setup, EaCC doubles the rate as compared to no-cooperation, which is a significant improvement.

For the same setup in Fig. 4(a), we also evaluate the impact of energy control part of EaCC on the average rate performance. In particular, Fig. 4(c) shows the average rate versus battery level at the receiver device R_1. In these results, we used Android based Nexus 5 smartphones. The average rate is calculated as the average over 10 trials (with different seeds). The computation under consideration in this experiment is $O(n^2)$, which counts the number of bytes in a packet for each byte in the packet. We consider that if the battery level of a device reduces below 40% threshold, then energy credits are not generated for the processing of the received packets. This makes $Q_{n,k}$ large over time, and after some point no packets are transmitted to that device for the processing task. In Fig. 4(c), when the battery level of R_1 reduces below 40%, then it stops receiving packets for processing. If there is no cooperation, then the rate towards R_1 reduces to 0. On the other hand, with EaCC, the rate is still higher than 0 thanks to having helper. The helper device with larger energy level (for the sake of this experiment), receives packets from the source, processes them, and forwards them to R_1, which receives already processed data. After 40% threshold, both EaCC and no-cooperation improve, because R_1 starts processing packets. This result shows the importance of energy-awareness in our cooperative computation setup.

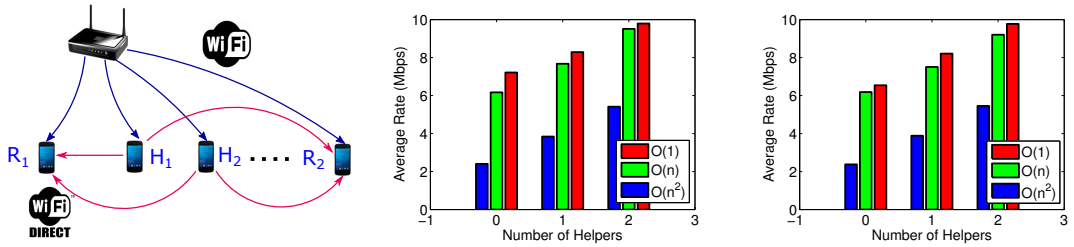
Now, we consider the impact of the number of helpers to overall rate performance. In particular, we develop a setup shown in Fig. 5(a), where there is one source, one receiver, and a varying number of helpers. In this setup, the source device, receiver, and the first two helper devices are Nexus 5 smartphones, while the other helpers are Nexus 7 tablets. Fig. 5(b) shows the average rate (averaged over 10 seeds) when EaCC is employed versus the number of helpers for different computational complexities such as $O(1)$, $O(n)$, and $O(n^2)$, where the processing task is counting the number of bytes in a packet. As expected, when complexity increases, the rate decreases. More interestingly, the increasing number of helpers increases the rates of all complexity levels. There are two reasons for this behavior. First, even if complexity level is low, *e.g.,* $O(1)$, processing power is still a bottleneck, and it can be solved by increasing the number of helpers. Note that after the number of helpers exceeds a value, the achievable rates saturate, which means that processing power is not a bottleneck anymore, but bandwidth is. The second reason is that receiving data over multiple interfaces increases diversity. In other words, when the channel condition over one interface (*e.g.,* between source and the mobile device) degrades, the other interface (*e.g.,* between two mobile devices) can still have a better channel condition.

In order to understand the real impact of processing power in a cooperative system, we tested both EaCC and cooperation (without computation and energy control) in the setup shown in Fig. 5(a). The results are provided in Fig. 5(c) when the complexity is $O(n^2)$. As can be seen, while EaCC significantly increases the rate with increasing number of helpers, cooperation slightly increases the rate (due to diversity). The improvement of EaCC over cooperation is as high as 83%, which is significant.

Finally, we consider a scenario that there are multiple receivers interested in different files. Fig. 6(a) shows the system model with one source, two receivers, and multiple helpers. In this setup, the source, two receivers, and the first helper is Android based Nexus 5 smartphone, while the rest of the helpers are Nexus 7 tablets. Fig. 6(b) and (c) show the average rate (averaged over 10 seeds) measured at R_1 and R_2 when EaCC is employed with respect to the increasing number of helpers, respectively. Similar to previous setups, $O(1)$, $O(n)$, and $O(n^2)$ correspond to different computational complexities, where the processing task is counting the number of bytes in a packet. As can be seen, the measured rate at both R_1 and R_2 increases with increasing number of helpers. This shows that our EaCC algorithm successfully accommodates multiple flows and receivers.

VII. Conclusion

We considered that a group of cooperative mobile devices, within proximity of each other, (i) use their cellular or Wi-Fi (802.11) links as their primary networking interfaces, and (ii)



(a) System Model (b) Rate at R_1 vs. Number of Helpers (c) Rate at R_2 vs. Number of Helpers

Fig. 6. (a) System model consisting of a source device, two receivers, and multiple helpers. Wi-Fi is used between the source and the receiver devices (R_1, R_2) and the helper devices ($H_1, \ldots$), while Wi-Fi Direct is used to connect the receiver devices with the helper devices. (b) EaCC: Average rate measured at receiver R_1 versus the number of helpers. (c) EaCC: Average rate measured at receiver R_2 versus the number of helpers.

exploit their D2D connections (Wi-Fi Direct) for cooperative computation. We showed that if mobile devices cooperate to utilize their aggregate processing power, it significantly improves transmission rates. Thus, for this scenario, we developed an *energy-aware cooperative computation* framework to effectively utilize processing power and energy. This framework provides a set of algorithms including flow, computation and energy controls as well as cooperation and scheduling. We implemented these algorithms in a testbed which consists of real mobile devices. The experiments in the testbed show that our *energy-aware cooperative computation* framework brings significant performance benefits.

REFERENCES

[1] "Cisco visual networking index: Global mobile data traffic forecast update," 2014-2019.

[2] "Ericsson mobility report," February 2015.

[3] "Wi-fi direct," http://www.wi-fi.org/discover-and-learn/wi-fi-direct.

[4] "Android," http://developer.android.com/develop/index.html.

[5] "Nexus tech specs," https://support.google.com/nexus/answer/6102470?hl=en.

[6] A. Singh, Y. Xing, and H. Seferoglu, "Cooperative compuration in device-to-device networks." [Online]. Available: http://nrl.ece.uic.edu and via [cs.NI] arXiv:1602.04400

[7] T. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. Wong, J. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5g cellular: It will work!" *Access, IEEE*, vol. 1, pp. 335–349, 2013.

[8] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with h.264/avc: tools, performance, and complexity," *Circuits and Systems Magazine, IEEE*, vol. 4, no. 1, pp. 7–28, First 2004.

[9] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/avc baseline profile decoder complexity analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 704–716, July 2003.

[10] J. K. Sundararajan, D. Shah, M. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets tcp: Theory and implementation," *Proceedings of the IEEE*, pp. 490–512, March 2011.

[11] P. Vingelmann, P. Zanaty, F. Fitzek, and H. Charaf, "Implementation of random linear network coding on opengl-enabled graphics cards," in *Wireless Conference, 2009. EW 2009. European*, May 2009, pp. 118–123.

[12] H. Shojania, B. Li, and X. Wang, "Nuclei: Gpu-accelerated many-core network coding," in *INFOCOM 2009, IEEE*, April 2009, pp. 459–467.

[13] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[14] M. Chiang, S. T. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: a mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, January 2007.

[15] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas, "Bits and coins: Supporting collaborative consumption of mobile internet," in *Proc. IEEE Infocom*, Hong Kong, April 2015.

[16] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," April 2014, technical report - arxiv:1310.0720v6[cs.GT].

[17] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," in *Proc. of ACM MobiHoc*, Hong Kong, May 2008.

[18] C. Boldrini, M. Conti, and A. Passarella, "Exploiting users' social relations to forward data in opportunistic networks: The hibop solution," in *Proc. of Pervasive and Mobile Computing*, October 2008.

[19] L. Keller, B. Cici, A. Le, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: Cooperative video streaming on smartphones," in *Proc. of ACM MobiSys*, June 2012.

[20] H. Seferoglu, L. Keller, A. Le, B. Cici, C. Fragouli, and A. Markopoulou, "Cooperative video streaming on smartphones," in *Proc. Allerton*, 2011.

[21] M. Ramadan, L. E. Zein, and Z. Dawy, "Implementation and evaluation of cooperative video streaming for mobile devices," in *Proc. of IEEE PIMRC*, Cannes, France, September 2008.

[22] S. Li and S. Chan, "Bopper: wireless video broadcasting with peer-to-peer error recovery," in *Proc. of IEEE ICME*, Beijing, China, July 2007.

[23] J. Chesterfield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez, "Exploiting diversity to enhance multimedia streaming over cellular links," in *Proc. of IEEE INFOCOM*, March 2005.

[24] ——, "A system for peer-to-peer video streaming in resource constrained mobile environments," in *Proc. of ACM U-NET*, December 2009.

[25] S. Ioannidis, A. Chaintreau, and L. Massoulie, "Optimal and scalable distribution of content updates over a mobile social network," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[26] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: a case study," in *Proc. of ACM Workshop on Challenged Networks (CHANTS)*, Chicago, IL, September 2010.

[27] R. K. Lomotey and R. Deters, "Architectural designs from mobile cloud computing to ubiquitous cloud computing - survey," in *Proc. IEEE Services*, Anchorage, Alaska, June 2014.

[28] T. Penner, A. Johnson, B. V. Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *Proc. IEEE GLOBECOM*, Austin, TX, December 2014.

[29] M. Satyanarayanan, S. Smaldone, B. Gilbert, J. Harkes, and L. Iftode, "Bringing the cloud down to earth: Transient pcs everywhere," in *MobiCASE'10*, 2010, pp. 315–322.

[30] E. Miluzzo, R. Caceres, and Y. Chen, "Vision: mclouds - computing on clouds of mobile devices," in *ACM workshop on Mobile cloud computing and services*, Low Wodd Bay, Lake District, UK, June 2012.

[31] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in mul- tihop radio networks," *IEEE Trans. on Automatic Control*, vol. 37, no. 12, December 1992.

[32] ——, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. on Information Theory*, vol. 39, no. 2, March 1993.

[33] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE Trans. on Networking*, vol. 16, no. 2, April 2008.

[34] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE/ACM Transactions on Information Theory*, vol. 34, no. 5, March 2000.