Hussein Al-Shatri, Sabrina Müller and Anja Klein, ”Distributed Algorithm for Energy Efficient Multi-Hop Computation Offloading,” in *Proc. IEEE Internation Conference of Communications*, May 2016.

Distributed Algorithm for Energy Efficient Multi-Hop Computation Offloading

Hussein Al-Shatri, Sabrina Müller and Anja Klein
Communications Engineering Lab, Technische Universität Darmstadt, Merckstrasse 25, 64283 Darmstadt, Germany
{h.shatri, s.mueller, a.klein}@nt.tu-darmstadt.de

***Abstract*—Computation offloading is a promising approach for reducing the computational load and extending the battery lifetime of mobile nodes. A network consisting of several wireless nodes accessing the cloud in a multi-hop fashion is considered. In multi-hop networks, offloading a computational task requires relaying the task by the intermediate nodes along the path towards the cloud. If the nodes are autonomous and rational, the intermediate nodes need to be incentivized for forwarding the tasks of other nodes. In this paper, a distributed decision algorithm which determines the set of tasks to be offloaded and the set of tasks to be locally computed for total energy minimization is proposed. Since a task needs to be sequentially forwarded by multiple nodes, each of which decides independently, decision conflicts on forwarding a task can occur. Accordingly, a novel coordination mechanism is proposed by which the forwarding nodes resolve their decision conflicts. In this coordination mechanism, nodes need only to exchange their forwarding decisions to resolve the conflicts. The results show that the proposed distributed algorithm achieves a performance close to the performance of the centralized algorithm.**

***Index Terms*—multi-hop, computation offloading, distributed decisions, coordination mechanism**

I. Introduction

Last years have witnessed significant advancements in the hardware and software development of wireless mobile nodes. However, mobile nodes are battery powered which limits their capabilities to run highly energy-consuming applications such as video processing, voice recognition and 3D localization/mapping processing [1], [2]. One emerging solution to this problem is mobile cloud computing [3], [4]. Mobile cloud computing aims at migrating the computational tasks and data storage from the battery powered mobile nodes to the resource-full cloud servers. A recent study by Cisco Inc. shows that cloud applications such as video/audio streaming, online gaming, social networking and online storage will occupy most of the mobile data traffic by 2019 [5].

One big advantage of computation offloading is that it relaxes the required storage and computation capabilities of future mobile nodes while being able to run computationally intensive applications. If the total network energy minimization is a target, a trade-off between offloading and locally computing tasks is appeared [3], [4]. Basically, the energy for computing a task is determined by the required number of CPU cycles whereas the energy for transmitting a task is determined by the amount of data needed to be transmitted to the cloud. Both the number of CPU cycles and the amount of transmitted data, are task-type depended. As a consequence, tasks which require few CPU cycles but a large amount of transmitted data for offloading will rather be computed locally. On the contrary, tasks with a low amount of offloading data are preferably being offloaded. Because of the scarcity of the radio resources, only a subset of nodes can offload simultaneously while the rest should locally compute to head their energy constraints and the delay constraints of their computation tasks. Thus, smart offloading decision algorithms need to be developed.

In the last few years, different computation offloading problems have been investigated. In [6], a node does not estimate the required execution time for its tasks before taking the offloading decision, but instead it initially computes the task locally and if the computation time exceeds a certain threshold, the node offloads the task to the cloud. The authors of [7] consider several interdependent tasks and use Lyapunov optimization to determine which task should be offloaded such that the execution time constraint is satisfied. In [8] and [9], the problem of deciding to which server a task should be offloaded is investigated. [10] considers the cloud architecture and develops a multi-objective service provisioning scheme to find a good trade-off between both the network computation and the nodes' battery lifetime. In [11], a multi-antenna node scenario is considered. For every node, the authors optimize jointly both the precoders and the assigned CPU cycles at the cloud.

The problem of deciding whether a node should compute its task locally or offload it to the cloud is considered in [12], [13]. Both [12] and [13] use game theory to develop a distributed algorithm where nodes are competing for the limited transmission resources for offloading their tasks.

As a summary, the above mentioned works consider only single hop wireless transmission scenarios. However, multi-hop transmission, where not all nodes have a direct radio link to the cloud, is of high importance in nowadays systems. To the best of the authors' knowledge, distributed multi-hop computation offloading is not well investigated in the literature. In [14], we formulate the multi-hop computation offloading problem as a multi-dimensional Knapsack problem and we propose a centralized algorithm to find the optimum node decisions.

This paper focuses on computation offloading in multi-hop networks. A novel distributed offloading algorithm aiming at minimizing the total network energy where nodes are modeled as decision makers is proposed. In multi-hop networks, the decision of offloading a task does not only depend on the

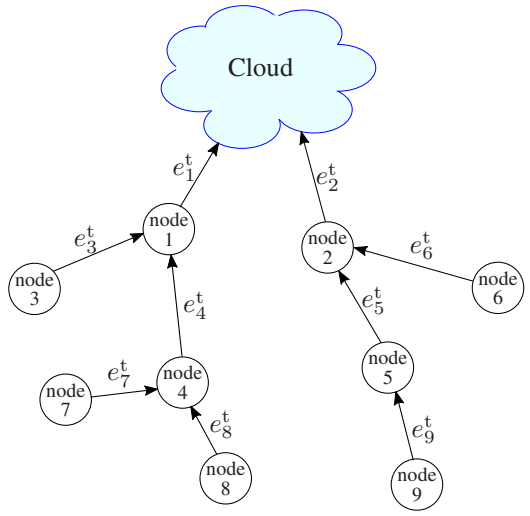

Fig. 1: An offloading tree where each node has a unique offloading route to the cloud.

corresponding node, but also on all the intermediate nodes on the path towards the cloud. These intermediate nodes need to simultaneously decide to forward this task so that it will successfully be offloaded. However, the intermediate nodes are energy limited and as well they should decide on their own tasks. Thus, they shall smartly select a subset of tasks to be forwarded such that the total network energy is minimized. Because intermediate nodes decide on forwarding tasks independently, there can be forwarding decision conflicts which will be resolved using a novel proposed coordination mechanism.

The rest of the paper is organized as follows. The next section describes the system model. Section III introduces the problem. The proposed algorithm is described in Section IV. The convergence of the proposed algorithm is analyzed in Section V. Numerical results are discussed in Section VI. Finally, the conclusion is drawn in Section VII.

II. System Model

A network consisting of N wireless nodes having access to the cloud is considered. Only a subset of the nodes has direct access to the cloud while other nodes access the cloud in a multi-hop fashion. Furthermore, the network is assumed to be always connected and the routing tree is a priori given in such a way that there is only a single route from each node to the cloud as shown in Fig. 1. Based on this, the considered network topology forms a rooted tree with the root and vertices being the cloud and the nodes, respectively. Let $\mathcal{P}_n$ be the set of predecessor nodes of node n which are all the nodes on the path between the n-th node and the cloud. Also, let $\mathcal{S}_n$ be the set of successor nodes of node n which are all nodes branching from the n-th node. Each edge in the tree departs from a single node to its nearest predecessor node. The radio links among the nodes have different qualities, i.e., different pathloss and shadowing effects. Then, the energies needed for transmitting a certain amount of data through different radio links are not the same. Accordingly, the amount of energy e_n^{t} the n-th node needs for transmitting a single bit to its nearest predecessor node is depicted next to the tree edges in Fig. 1.

In the considered network, nodes perform simultaneous computation sessions where in each session, every node is required to compute an independent task. It is assumed that the duration of a computation session is arbitrarily small so that node positions are fixed during this session. It is further assumed that a task is non-separable and does not have a hard time constraint. Therefore, the decision of whether to offload or not is based only on the energy consumption. The assumption that nodes are allowed to compute only their own tasks is considered in this paper.

A task n should be either computed locally at the corresponding node n or offloaded to the cloud for computation. Let L_n be the required number of instructions needed for computing the n-th task and let B_n be the required number of bits needed for offloading the n-th task. It is assumed that the cloud is resource-full and therefore, we do not consider the cloud computation. Furthermore, the transmission back to the nodes is not considered.

Concerning task computation, the energy of computing a task locally depends on the node's processor. Let e_n^{c} be the compute energy per instruction at the n-th node measured in Joule per bit. Then, the amount of energy consumed for computing the n-th task at the n-th node is given by

$$E_n^{\mathrm{c}} = L_n e_n^{\mathrm{c}}. \tag{1}$$

Concerning task transmission, let $h_{m,n} \in \mathbb{C}$ be the coefficient of the channel between the transmit node n and the receive node m. According to Shannon formula, the transmit power p_n^{t} of the n-th node adapted for achieving the required data rate R_m at the m-th node is calculated as

$$p_n^{\mathrm{t}} = \frac{\sigma^2}{|h_{m,n}|^2}\left(2^{R_m} - 1\right) \tag{2}$$

where σ^2 is the noise power at the m-th receive node. Then, the amount of transmit energy per bit the n-th node consumes is given by

$$e_n^{\mathrm{t}} = \frac{p_n^{\mathrm{t}}}{R_m}, \tag{3}$$

with a unit of Joule per bit. Given the number B_n of transmitted bits required for offloading the n-th task, the transmission energy of node n for transmitting its task to its nearest predecessor node is calculated as

$$E_n^{\mathrm{t}} = B_n e_n^{\mathrm{t}}. \tag{4}$$

If the n-th node is not directly connected to the cloud, predecessor nodes in $\mathcal{P}_n$ need to forward the n-th task towards the cloud. Hence, the required energy needed by the predecessor node $m \in \mathcal{P}_n$ for forwarding the n-th task is calculated as

$$E_{m,n}^{\mathrm{f}} = B_n e_m^{\mathrm{t}}. \tag{5}$$

Finally, every node n has an energy constraint E_n^{tot} in every computation session which is assumed to be enough for at least locally computing its task, i.e., $E_n^{\mathrm{tot}} \geq E_n^{\mathrm{c}}$.

III. Problem Statement

In this section, the computation offloading problem is introduced. Aiming at minimizing the total network energy, the problem of finding the optimum split of the tasks into two sets of offloaded tasks and locally computed tasks will be formulated. To this end, a single computation session where each node has a single task is considered. Let $\alpha_n \in \{0,1\}$ be the n-th node's decision variable, where $\alpha_n = 0$ implies that node n decides to compute its task and $\alpha_n = 1$ means that node n decides to offload its task to the cloud. Based on this, one can formulate the offloading optimization problem as

$$\left(\alpha_1^{\text{opt}}, \ldots, \alpha_N^{\text{opt}}\right) = \underset{\alpha_1,\ldots,\alpha_N}{\text{argmin}} \left\{ \sum_{n=1}^{N} \left(\alpha_n \left(E_n^{\text{t}} + \sum_{m \in \mathcal{P}_n} E_{m,n}^{\text{f}} \right) + (1-\alpha_n) E_n^{\text{c}} \right) \right\} \tag{6}$$

subject to

$$\alpha_n E_n^{\text{t}} + (1-\alpha_n) E_n^{\text{c}} + \sum_{k \in \mathcal{S}_n} \alpha_k E_{n,k}^{\text{f}} \leq E_n^{\text{tot}}, \ \forall n. \tag{7}$$

The objective function of (6) describes the total energy consumed in the network by either offloading or locally computing the tasks including the forwarding energy based on the nodes decision variables α_n, $\forall n$. The total energy constraint described in (7) states that the total energy consumed by every node for either offloading or locally computing its task and the additional energy consumed for forwarding other nodes' tasks is constrained by the node energy constraint E_n^{tot}. The optimization problem of (6)–(7) is a binary linear program and it is inseparable due to the task forwarding terms.

IV. Distributed Algorithm

A. Overview

The optimization problem of (6)–(7) can be solved using a centralized algorithm which can be implemented at the cloud or at a node. To avoid gathering all the network information into a single unit which requires a large amount of signaling overhead, a distributed algorithm aiming at solving the problem of (6)–(7) with nodes acting as decision makers is of interest.

Before explaining the proposed distributed algorithm, the required signaling in the network will be introduced. In the proposed distributed algorithm, a node is periodically informed of the network topology. More specifically, the n-th node needs to know both the set $\mathcal{P}_n$ of predecessor nodes and the set $\mathcal{S}_n$ of successor nodes. Moreover, a node n knows the transmit energy e_m^{t} per bit for all predecessor nodes $m \in \mathcal{P}_n$ on its path to the cloud. Finally, nodes have perfect decision information in the sense that they know the decisions of each other on their respective tasks of interest, i.e., own tasks and forwarding tasks. More specifically, a node knows the decisions of other nodes on its path to the cloud only for the common tasks in which this node and one or more other nodes need to forward a task jointly.

Basically, the distributed proposed algorithm consists of two main steps. In the first step, initial decisions are made where every node initially decides to offload its own task or not. If a node decides to offload, it sends a forwarding request to all its predecessor nodes. Then, all nodes who received a forwarding request make an initial forwarding decision. Because a task may need to be forwarded by multiple nodes and the nodes make their initial forwarding decisions without cooperating with each other, there could be some forwarding decision conflicts. In the second step, nodes should change their forwarding decisions aiming at resolving the decision conflicts. After resolving the decision conflicts, the forwarding nodes send back a forwarding acceptance or rejection to the requesting nodes. Only nodes who get a forwarding acceptance will offload their tasks whereas the rejected tasks will be computed locally at the corresponding nodes.

B. Initial Decisions

In a multi-hop network, a node cannot decide for offloading alone unless its predecessor nodes have the ability and willingness to forward its task to the cloud. Therefore, a node needs to incentivize its predecessor nodes by rewarding them for forwarding its task. Before defining the reward, the initial offloading decisions need to be introduced. Each node first makes an initial decision on offloading its task. To this end, the initial offloading decision at the n-th node is calculated as

$$\alpha_n^{\text{ini}} = \begin{cases} 1, & \text{if } E_n^{\text{c}} > E_n^{\text{t}} + \sum\limits_{m \in \mathcal{P}_n} E_{m,n}^{\text{f}} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

(8) implies that each node prefers offloading only if the total offloading energy $E_n^{\text{t}} + \sum\limits_{m \in \mathcal{P}_n} E_{m,n}^{\text{f}}$ including the forwarding energy is less than the local computing energy E_n^{c}. It can be noted that, in finding the initial decisions, the nodes consider not only their transmit energy but also the forwarding energies from the predecessor nodes. Therefore, the initial decisions are beneficial in minimizing the total energy of the whole network rather than in minimizing their own energy.

If a node n initially decides to offload, it will send a forwarding request to its predecessor nodes $m \in \mathcal{P}_n$ together with a rewarding offer. In other words, if all predecessor nodes in $\mathcal{P}_n$ accept to forward the n-th task, each of them will receive a reward which is calculated as

$$W_n = E_n^{\text{c}} - E_n^{\text{t}} - \sum_{m \in \mathcal{P}_n} E_{m,n}^{\text{f}}. \tag{9}$$

The reward function described in (9) represents exactly how much energy can be saved in the network by offloading the n-th task. Thus, nodes can use only the reward values for deciding which tasks to offload at best. For instance, the n-th node receives two forwarding requests and it can afford forwarding only one of them. Then, it will be beneficial for the network if the task with the higher reward is forwarded, which implies that a higher amount of energy will be saved in the network.

After taking the initial offloading decisions, every node n calculates its initial remaining energy as

$$E_n^{\mathrm{rest}(0)} = E_n^{\mathrm{tot}} - \left(\alpha_n^{\mathrm{ini}} E_n^{\mathrm{t}} + \left(1-\alpha_n^{\mathrm{ini}}\right) E_n^{\mathrm{c}}\right), \tag{10}$$

which will be used in forwarding the successor node tasks. It can be noted that the distributed initial decisions correspond to the optimum for the whole network only if there is enough forwarding energy available at all nodes. Nevertheless, nodes do not have always enough remaining energy to accept all forwarding requests. Therefore, finding the set of tasks to be forwarded for minimizing the total network energy will solve the problem [14]. It can be pointed out here that the information of the remaining energies $E_n^{\mathrm{rest}(0)}$ is not exchanged among the nodes.

Now, the nodes which receive forwarding requests need to make initial forwarding decisions. Note that nodes take the initial forwarding decisions without any exchange of information. Therefore, forwarding decision conflicts can occur. As will be described in the next section, nodes resolve their decision conflicts sequentially and iteratively. Let $\beta_{n,k}^{(i)} \in \{0,1\}$ be the forwarding decision variable of the k-th task at the n-th node in the i-th iteration. $\beta_{n,k}^{(i)} = 1$ implies that the node n accepts the k-th node's request in the i-th iteration whereas $\beta_{n,k}^{(i)} = 0$ means that the request of forwarding the k-th task is rejected by the n-th node in the i-th iteration, with $k \in \mathcal{S}_n$. Accordingly, the node n can decide initially which tasks it forwards by solving the following optimization problem:

$$\left(\beta_{n,k}^{(0)}\right)_{k\in\mathcal{S}_n} = \underset{\{\beta_{n,k}\}_{k\in\mathcal{S}_n}}{\operatorname{argmax}} \left\{ \sum_{k\in\mathcal{S}_n} \alpha_k^{\mathrm{ini}} \beta_{n,k} W_k \right\} \tag{11}$$

subject to

$$\sum_{k\in\mathcal{S}_n} \alpha_k^{\mathrm{ini}} \beta_{n,k} E_{n,k}^{\mathrm{f}} \leq E_n^{\mathrm{rest}(0)}. \tag{12}$$

Based on the description in this section, two simple scenarios can be solved distributedly right away. First, a single-hop network scenario where all nodes are connected directly to the cloud, i.e., $|\mathcal{P}_n| = 0, \forall n$ where $|.|$ denotes the size of a set. In this case, the optimum offloading decisions are $\alpha_n^{\mathrm{opt}} = \alpha_n^{\mathrm{ini}}$, $\forall n$ which implies that every node independently decides for the least energy consuming option. Second, a two-hop network scenario where all tasks can be forwarded at most by a single node, i.e., $|\mathcal{P}_n| \leq 1, \forall n$. In this case, each of the forwarding nodes has different set of tasks to be forwarded. Thus, the forwarding decision of a task k is taken by a single node n which solves the optimization problem of (11)–(12) for the optimum decisions $\beta_{n,k}^{\mathrm{opt}} = \beta_{n,k}^{(0)}$, $k \in \mathcal{S}_n$.

C. Resolving Decision Conflicts

This section discusses general scenarios where tasks can be forwarded by more than a single node, i.e., $|\mathcal{P}_n| \geq 2$. For a task which needs to be forwarded by more than one node, the forwarding nodes have to simultaneously accept or reject the forwarding requests. If there is a decision conflict in forwarding a task, it will not be offloaded and nobody will get the reward. However, the initial forwarding decisions are done blindly where every node solves the optimization problem of (11)–(12) regardless of the other nodes' decisions. Therefore, conflicts in forwarding decisions of a task can always happen and thus, a distributed iterative coordination mechanism needs to be employed at the nodes to resolve their forwarding decision conflicts.

As mentioned in Section IV-A, nodes have perfect decision information which means they know the decisions of each other in every iteration. Based on this information, a node can change its decision aiming at resolving its decision conflict with other nodes. To this end, an iterative coordination mechanism is proposed where a node changes its forwarding decision only by observing the others decisions.

In the proposed coordination mechanism, a node takes an action, i.e., changes its decision, only if it has both accepted tasks and rejected tasks under conflict with other nodes. Because a node takes an action only at the conflicting tasks, it will not loose anything by taking an action but rather it may get a reward if a conflict is resolved. Let $\gamma_n^{(i)}$ be the measure of nodes' unwillingness to forward the n-th task and it is calculated as

$$\gamma_n^{(i)} = |\mathcal{P}_n| - \sum_{m\in\mathcal{P}_n} \beta_{m,n}^{(i)}, \tag{13}$$

where $|\mathcal{P}_n|$ represents the total number of nodes needed to forward the n-th task for offloading and $\sum_{m\in\mathcal{P}_n} \beta_{m,n}^{(i)}$ represents the number of nodes who accept forwarding the n-th task in the i-th iteration. From (13), one can deduce that $\gamma_n^{(i)} = 0$ implies that the n-th task will be offloaded whereas $\gamma_n^{(i)} = |\mathcal{P}_n|$ implies that the n-th task will be locally computed. Moreover, the smaller $\gamma_n^{(i)}$, the higher the chances that the conflict can be resolved and vice versa. In the proposed mechanism, nodes take actions iteratively where in every iteration, a node deselects one of the conflicting accepted tasks, calculates the new remaining energy and selects one or more conflicting rejected tasks. Let $\mathcal{A}_n^{(i)}$ and $\mathcal{J}_n^{(i)}$ be the set of conflicting accepted tasks by the n-th node in the i-th iteration and the set of conflicting rejected tasks by the n-th node in the i-th iteration, respectively. Then, a node n takes an action if both sets $\mathcal{A}_n^{(i)}$ and $\mathcal{J}_n^{(i)}$ are nonempty. Taking an action can be explained in three steps. Firstly, node n deselects the task $t \in \mathcal{A}_n^{(i)}$ with the highest unwillingness $\gamma_t^{(i)}$. If $\mathcal{A}_n^{(i)}$ contains more than one task with the maximum $\gamma_t^{(i)}$, the one with the minimum ratio $W_t/E_{n,t}^{\mathrm{f}}$ will be deselected. Secondly, node n recalculates its remaining energy including the energy saved by deselecting the t-th task, i.e.,

$$E_n^{\mathrm{rest}(i)} = E_n^{\mathrm{rest}(i-1)} - \sum_{k\in\mathcal{S}_n} \alpha_k^{\mathrm{ini}} \beta_{n,k}^{(i-1)} E_{n,k}^{\mathrm{f}} + E_{n,t}^{\mathrm{f}}. \tag{14}$$

Finally, node n selects a subset of the rejected tasks in $\mathcal{J}_n^{(i)}$ such that the sum of the ratio $W_j/\gamma_j^{(i)}$ is maximized. Accordingly, the optimization problem is stated as

$$\left(\beta_{n,j}^{(i+1)}\right)_{j\in\mathcal{J}_n} = \underset{\{\beta_{n,j}\}_{j\in\mathcal{J}_n}}{\operatorname{argmax}} \left\{ \sum_{j\in\mathcal{J}_n} \frac{W_j}{\gamma_j^{(i)}} \right\} \tag{15}$$

subject to

$$\sum_{j\in\mathcal{J}_n} \alpha_j^{\text{ini}} \beta_{n,j} E_{n,j}^{\text{f}} \leq E_n^{\text{rest}(i)}. \tag{16}$$

It can be noted from the optimization problem of (15)–(16) that when $\beta_{n,j}^{(i+1)} = 0$, $\forall j \in \mathcal{J}_n^{(i)}$, node n takes no action as there is not enough energy to select a task in $\mathcal{J}_n^{(i)}$ and thus the t-th task will remain an accepted task. To avoid cycling, a node will not deselect the same task twice.

If no node takes an action, the coordination mechanism stops. Afterwards, the tasks with $\gamma_n^{(I)} \neq 0$ will be rejected and computed locally where I denotes the index of the last iteration. By rejecting the task n, the n-th node will compute locally and thus, it needs to recalculate its initial remaining energy $E_n^{\text{rest}(0)}$. If $E_n^{\text{rest}(0)}$ is not enough for forwarding the tasks the n-th node promised to forward, the algorithm will start from the beginning with the n-th node initially deciding to compute, i.e., $\alpha_n^{\text{ini}} = 0$.

V. Convergence Analysis

In the proposed distributed algorithm, nodes make their initial decisions simultaneously including the offloading and the forwarding decisions. However, the iterative coordination mechanism is run sequentially over the forwarding nodes. In particular, only nodes with nonempty sets $\mathcal{A}_n^{(i)}$ and $\mathcal{J}_n^{(i)}$ will participate in the coordination mechanism. In the coordination mechanism, the sizes of the sets $\mathcal{A}_n^{(i)}$ and $\mathcal{J}_n^{(i)}$ for the participating nodes are non-increasing. Based on this and because of the limited remaining energies at the nodes, the number of actions a node can take is limited and never increases. Therefore, the coordination mechanism converges.

Because of limited remaining energies at the nodes, nodes cannot freely select to forward any combinations of the requested tasks in the set $\mathcal{J}_n^{(i)}$ by deselecting any of the tasks in the set $\mathcal{A}_n^{(i)}$. Therefore, the coordination mechanism does not necessarily resolve all the forwarding conflicts due to the energy limitation. Therefore, tasks with unresolved conflicts will be rejected and the corresponding node will compute its task locally.

VI. Numerical Results

In this section, the performance of the proposed algorithm is investigated as a function of the ratio $\frac{B_n}{L_n}$ of the number B_n of bits of a task to the number L_n of instructions of the same task. The total network energy is considered as a performance measure. A scenario consisting of $N = 100$ nodes is considered. For the following results, Monte-Carlo simulations with 100 snapshots of different random trees, channel realizations and numbers of instructions are preformed. Furthermore, the simulation parameters are carefully tuned such that the trade-off between offloading and locally computing for the minimum total energy is not trivial, i.e., the optimum is not simply the initial decisions of (8) where each node decides for the least energy option. In every snapshot, a single computation session is preformed where every node has an independent non-separable task. The number of instructions of every task

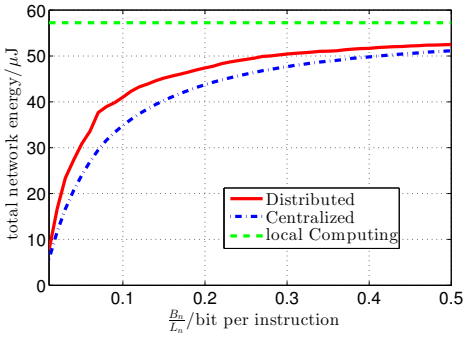

Fig. 2: Average total network energy versus $\frac{B_n}{L_n}$.

is randomly picked from a uniform distribution of the integers between 1 and 100. For any number of instructions and the given ratio $\frac{B_n}{L_n}$, the corresponding number B_n of bits can be calculated. Concerning the computing energy, we assume that nodes have different computing capabilities. Therefore, the compute energy per instruction e_n^{T}, $\forall n$ is drawn from a uniform distribution between 8 nJoule/instruction and 16 nJoule/instruction. Concerning the transmitting energy, the channel is modeled as a complex Gaussian channel with a unit average channel gain. Moreover, the receive noise is modeled as additive white Gaussian with zero mean and unit variance. Based on the channel gain in every channel realization, the transmit energy per bit can be calculated using (3). It is assumed that half of the nodes, randomly selected, have energy constraints exactly enough for computing, i.e., $E_n^{\text{tot}} = E_n^{\text{c}}$ while the other half of the nodes have higher energy constraints, i.e., $E_n^{\text{c}} \leq E_n^{\text{tot}} \leq 2E_n^{\text{c}}$.

To assess the performance of our proposed algorithm, two benchmark schemes are considered. First, a centralized algorithm which solves the optimization problem of (6)–(7) using the Gurobi solver [15] is considered. To see the advantage of offloading in terms of energy minimization, the local computing scheme where all nodes compute their tasks locally is considered as an upper bound of the total network energy consumption.

Fig. 2 shows the average total network energy as a function of $\frac{B_n}{L_n}$. Since in the local computing scheme all nodes compute locally, it consumes the same total energy of 57.3 μJ regardless of the ratio $\frac{B_n}{L_n}$. In general, at low $\frac{B_n}{L_n}$, it is preferable for the nodes to offload their tasks as the energy of transmission is less than the energy of computing. As the ratio $\frac{B_n}{L_n}$ gets close to 0.5, the energy of transmission is comparable to the computing energy and hence, more nodes compute locally. Furthermore, it can be noticed that the performance of our proposed distributed algorithm is close to the centralized algorithm with the highest loss of 15% at $\frac{B_n}{L_n} = 0.1$.

Fig. 3 shows the percentage of the number of tasks offloaded in the network to the total number of tasks as a function of $\frac{B_n}{L_n}$. In the low ratios of $\frac{B_n}{L_n}$, a large number of tasks is offloaded whereas few nodes offload their task at high ratio of $\frac{B_n}{L_n}$. Also, it can be noticed that using our distributed algorithm, less

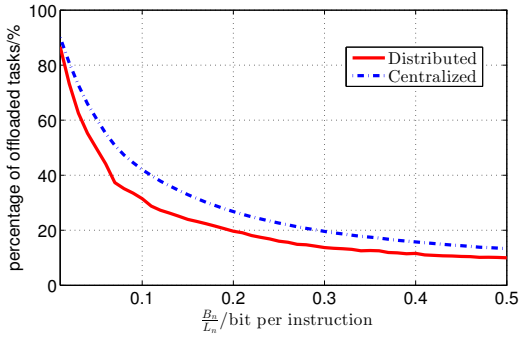

Fig. 3: Percentage of offloaded tasks to the total number of task versus $\frac{B_n}{L_n}$.

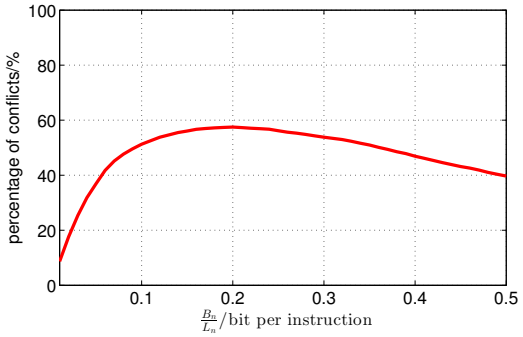

Fig. 4: Percentage of conflicting tasks to the total number of tasks versus $\frac{B_n}{L_n}$.

nodes will offload their tasks as compared to the centralized algorithm.

In Fig. 4, the percentage of the number of tasks with forwarding conflicts to the total number of tasks after the initial blind forwarding decisions is picked as a function of the ratio $\frac{B_n}{L_n}$. As can be seen in the figure, there is a peak at $\frac{B_n}{L_n} = 0.2$ where the number of conflicts reaches 60%. If the ratio $\frac{B_n}{L_n}$ decreases, less number of bits is required to be transmitted and the number of conflicts decreases since nodes have enough energy. If the ratio $\frac{B_n}{L_n}$ increases, more bits are required to be transmitted, and thus fewer nodes prefer offloading than computing.

VII. Conclusion

In this paper, a network consisting of several nodes accessing the cloud in a multi-hop fashion is considered. In this scenario, each node has an independent, non-separable computational task. Aiming at minimizing the total network energy, each node should decide to either compute its task locally or to offload it to the cloud. A new distributed algorithm where the individual nodes are the decision makers is proposed. In this algorithm, every node decides on its own either to offload its task or to locally compute it. If it decides to offload, it has to send a forwarding request to all intermediate nodes in its path to the cloud. All nodes which receive a forwarding request blindly make initial forwarding decisions. Moreover, a new coordination mechanism is proposed to resolve the decision conflicts. The results show that the performance of our proposed distributed algorithm is close to the performance of the centralized algorithm.

Acknowledgment

This work has been performed in the context of the DFG funded CRC 1053 MAKI – subproject B03. Authors would like to thank Mousie Fasil for his fruitful comments.

References

[1] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy ?" *Computer*, no. 4, pp. 51–56, April 2010.

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

[3] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, February 2013.

[4] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, April 2010.

[5] "Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019," February 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[6] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in *Proc. International Conference on Parallel and Distributed Systems*, Hsinchu, December 2007, pp. 1–8.

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

[8] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, Redondo Beach, August 2012, pp. 279–284.

[9] V. Cardellini, V. D. N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F. L. Presti, and V. Piccialli, "A game-theoretic approach to computation offloading in mobile cloud computing," *Mathematical Programming*, pp. 1–29, February 2015.

[10] M. Anastasopoulos, A. Tzanakaki, and D. Simeonidou, "Energy-aware offloading in mobile cloud systems with delay considerations," in *Proc. IEEE Global Communications Conference*, Austin, December 2014, pp. 42–47.

[11] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing." [Online]. Available: http://arxiv.org/abs/1412.8416

[12] E. Meskar, T. Todd, D. Zhao, and G. Karakostas, "Energy efficient offloading for competing users on a shared communications channel," in *Proc. IEEE International Conference on Communications*, London, June 2015, pp. 3192–3197.

[13] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, no. 4, pp. 974–983, April 2015.

[14] S. Müller, H. Al-Shatri, M. Wichtlhuber, D. Hausheer, and A. Klein, "Computation offloading in wireless multi-hop networks: Energy minimization via multi-dimensional knapsack problem," in *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, Hong Kong, September 2015, pp. 1907–1912.

[15] I. Gurobi, *Gurobi optimizer reference manual*. [Online]. Available: http://www.gurobi.com