Collaborative Computing for Advanced Tactile Internet Human-to-Robot (H2R) Communications in Integrated FiWi Multirobot Infrastructures

Mahfuzulhoq Chowdhury and Martin Maier, *Senior Member, IEEE*

***Abstract*—With the emergence of the Tactile Internet and advent of remote-controlled robots, proper task allocation among robots has attracted significant attention to enable robotic applications and services based on the human-to-robot communications paradigm. However, limited computing, energy, and storage resources of robots may hinder the successful launch of such applications. Task offloading to collaborative nodes is a promising approach to improve the task execution time and energy efficiency of robots. In this paper, we investigate a proper task allocation strategy by combining suitable host robot selection and computation task offloading onto collaborative nodes. We exploit conventional cloud, decentralized cloudlets, and neighboring robots as collaborative nodes for computation offloading in support of a host robot's task execution. More specifically, our proposed task allocation policy selects a suitable robot based on several key parameters, including robot availability, remaining energy, and task execution time. Furthermore, our proposed computation offloading strategy examines the suitability of collaborative nodes in terms of task response time and energy consumption and then chooses an appropriate collaborative node to conduct the requested computation. We introduce an adaptive resource allocation model and develop an analytical framework to evaluate the task allocation delay, energy consumption, and task response time for noncollaborative and collaborative task execution scheme across integrated fiber-wireless multirobot networks. The results show that the proposed collaborative task execution scheme outperforms the noncollaborative scheme in terms of task response time and energy consumption efficiency.**

***Index Terms*—Collaborative computing, fiber-wireless (FiWi), human-agent-robot teamwork (HART), Tactile Internet, task allocation.**

I. Introduction

With the emergence of 5G mobile networks and rapid development of smart devices, it is expected that a wide variety of real-time machine-centric applications are finding their way into our life. To unleash their full potential, some of those 5G applications (e.g., cognitive assistance) require low-latency communications with ultrareliable, ultra-responsive, and intelligent network connectivity. To meet the aforementioned requirements, we recently evaluated the performance gains obtained from unifying coverage-centric LTE-Advanced (LTE-A) heterogeneous networks (HetNets) with capacity-centric fiber-wireless (FiWi) access networks based on data-centric Ethernet passive optical network (EPON) and gigabit-class WLAN technologies [1]. We showed that very low latency on the order of 1 ms and ultrahigh reliability with almost guaranteed network connectivity can be achieved in FiWi enhanced LTE-A HetNets. Note that both very low latency and ultrahigh reliability are not only crucial for future 5G networks but are also essential design goals of the emerging *Tactile Internet* to remotely steer/control virtual and/or physical objects (e.g., remote-controlled robots) [2].

In order to realize Tactile Internet robotic applications, the feasibility of several key enabling technologies such as FiWi enhanced 4G mobile networks, cloudlets, mobile-edge computing (MEC), and cloud robotics were identified in [3]. Moreover, the subtle differences between 5G, Internet of Things (IoT), and Tactile Internet were outlined based on their different underlying communications paradigms. IoT and 5G leverage on machine-to-machine and human-to-human (H2H) communications, respectively. Conversely, the Tactile Internet relies on human-to-robot (H2R) communications to facilitate the interaction between human operators and tele-operated robots. Note that, in [3], we introduced the concept of FiWi enabled H2R communications for Tactile Internet applications and identified specific H2R communications requirements apart from low latency and high reliability, most notably efficient H2R task allocation. However, we did not elaborate on how H2R communications may be realized in technically sufficient detail.

Taking the respective areas where robots are strong and humans are weak into account, FiWi enabled H2R communications aims at leveraging on their "cooperative" and "collaborative" autonomy such that humans and robots may complement each other. This design approach is also known as human-agent-robot teamwork (HART) [4]. The potential benefits of real-time HART applications are immense, ranging from industrial applications (e.g., coal mining) to emergency response operations (e.g., food supply and human rescue). However, highly reliable and secure communication platforms along with intelligent task allocation and service coordination strategies are needed to meet their stringent quality-of-service requirements.

To ensure proper coordination in both local and nonlocal HART task allocation processes, integrated FiWi

Manuscript received April 8, 2017; accepted October 1, 2017. Date of publication October 10, 2017; date of current version December 11, 2017. This work was supported by the Natural Sciences and Engineering Research Council of Canada under Grant 2016-04521. *(Corresponding author: Mahfuzulhoq Chowdhury.)*

The authors are with the Optical Zeitgeist Laboratory, Institut National de la Recherche Scientifique, Montreal, QC H5A1K6, Canada (e-mail: mahfuzulhoq.chowdhury@emt.inrs.ca; maier@emt.inrs.ca).

Digital Object Identifier 10.1109/JIOT.2017.2761599

TABLE I
COMPARISON OF THIS PAPER WITH EXISTING TASK ALLOCATION SCHEMES

Scheme	**Main Features**	**Robot Selection and Computation Offloading**	**Resource Management**	**Coordination of Task Allocation and Offloading**	**Other Comments**
Minimum Distance based robot selection [7]	Used only robots distance value for task allocation, did not consider task re-allocation	Used only for robot selection based task execution, did not consider task offloading	Not addressed	Not considered	May suffer from higher task execution latency
Fixed task assignment scheme [8]	Used only robot identification number for task allocation, did not considered task re-allocation	Used only for robot selection based task execution, did not consider task offloading	Not addressed	Not considered	May suffer from higher task execution latency
Remote cloud based system [13, 14]	Minimum energy consumption of host mobile device, considered remote cloud server	Used only for location independent computation offloading task	Not addressed	Not considered	Not considered location dependent (e.g., sensing) task
Delay optimal computing [16]	Minimized computation task execution time, considered cloudlet for offloading	Used only for computation offloading task, did not consider any robot selection scheme	Not addressed	Not considered	Not considered location dependent H2R task execution
Hybrid cloud based system [15, 17]	Minimized energy consumption of host mobile device, considered remote cloud and decentralized cloudlet for offloading	Only offload computation task to suitable cloud server, did not consider any robot selection scheme	Not addressed	Not considered	Not considered location dependent H2R task execution
Our proposed scheme	Host robot selection for H2R task execution based on different parameters (e.g., distance, energy, skill, and availability)	Considered both suitable host robot selection and computation sub-task offloading to suitable collaborative node	Addressed	Considered	Considered both location dependent and independent H2R task
	Considered multiple collaborative cloud nodes for computation offloading	Offload computation sub-task to collaborative node that satisfies resource availability, task deadline, and minimum energy consumption criteria			

multirobot communication infrastructures play a crucial role, whereby humans (H), agents (A), and robots (R) perform the following respective functions. A human user delegates her task request to a robot through a nearby agent. The agent in turn coordinates the task allocation/robot selection process, while the selected robot executes the human's task. Note that typically the agent is placed at the optical-wireless interface of integrated FiWi multirobot networks.

For the efficient utilization of robotic resources, proper task allocation among robots is crucial by taking the different capabilities of robots and the specific task requirements such as task execution deadline and energy consumption of robots into account [5]. Most existing multirobot task allocation (MRTA) studies focus on only one or a few parameters for robot selection, e.g., the robot's energy [6], distance to task location [7], or task priority [8]. Clearly, in more advanced robot selection schemes, additional parameters need to be considered such as the robot's ability, availability, and task execution time. More importantly, note that the aforementioned MRTA schemes suffer from several shortcomings, most notably, the lack of control and coordination as well as task reallocation mechanisms, thus resulting in an increased task execution delay and energy consumption of selected robots. Further, some H2R tasks may involve location dependent sensing subtasks (e.g., image capturing) and/or location independent computation subtasks (e.g., processing of sensed data). For these types of task, different challenging scenarios may arise, where the available robots may fail to execute either the sensing or computation subtasks or both due to their limited computing and storage resources.

To address these shortcomings, mobile devices increasingly seek assistance from collaborative nodes (e.g., mobile-cloud computing and mobile device-to-device communications) for executing their computation tasks, a trend also known as *cyber-foraging* [9] or *collaborative computing* [10]–[12]. Despite recent progress on MRTA, the impact of collaborative/joint task execution schemes that consider both host robot and collaborative nodes (e.g., central cloud or cloudlet) for the H2R task execution process has not been examined in sufficient detail previously. For clarification, note that the H2R task execution process typically consists of two subparts. The first one comprises the initial processing or sensing subtask (e.g., capturing an image), which can be executed only by the selected host robot located in the given task area. The second subpart of the task involves the location independent computation/processing of the sensed data (e.g., image/face detection), which can be done by the host robot itself or alternatively be offloaded to collaborative nodes. Computation task offloading to collaborative nodes comes in three flavors: system-based, method-based, and optimization-based offloading. System-based offloading is used to decide whether to offload the computation task to an infrastructure-based cloud (e.g., central cloud [13], [14] and cloudlet [15], [16]) or an ad-hoc virtual cloud [17], [18]. Method-based offloading [19] involves application partitioning and code migration to an infrastructure-based cloud. Optimization-based task offloading, on the other hand, aims at achieving objectives related to minimizing energy consumption [14] and task response time [16] of mobile devices, though computation task offloading might not always be beneficial for mobile devices. Several studies showed that computation task offloading can help save energy of mobile devices only if they consume less energy during offloading the computation task to a central cloud than executing the task by themselves [13], [14].

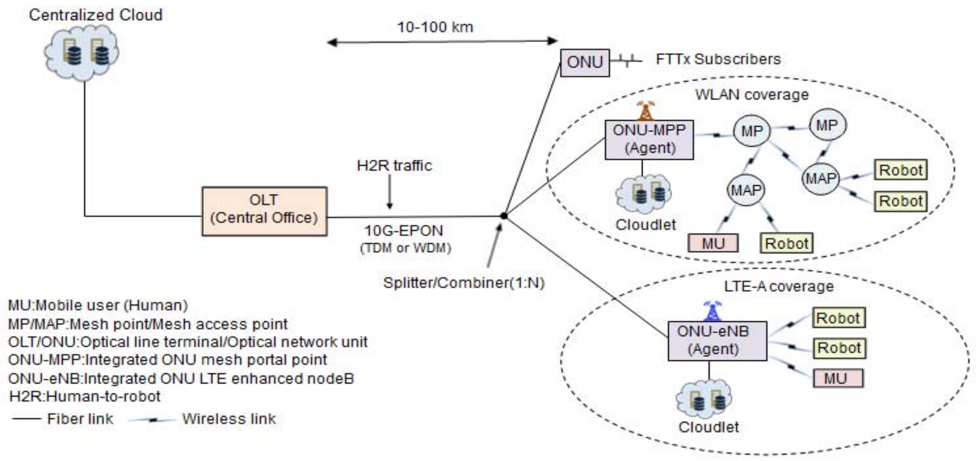


Fig. 1. Integrated FiWi multirobot network architecture for coordinating local and nonlocal H2R task allocation.

Most previous studies considered either full task (i.e., both sensing and computation subparts) allocation to a robot or only computation offloading (i.e., subpart of the full task) onto cloud nodes (central cloud and local cloudlet) for execution (also see Table I for details). Hence, the question of how to assign a local/nonlocal H2R task to a host robot considering different tasks and robot types with their respective energy consumption, availability, distance to task location, processing, and moving speed remains an open research challenge. Moreover, how to coordinate both task allocation among robots and computation subtask offloading onto collaborative nodes represents another unaddressed research challenge. Further, different challenging situations need to be investigated, where both collaborative cloud/cloudlet nodes may satisfy or fail to satisfy given computation offloading requirements.

Toward this end, in this paper we develop an efficient H2R task allocation strategy that includes both suitable host robot and collaborative node selection in integrated FiWi multirobot networks. We propose to use not only the central cloud and local cloudlets as collaborative nodes but also available neighboring robots for computation subtask offloading. To achieve energy savings of the robots and accomplish H2R tasks within their required time, the main objective of this paper is to select the proper policy for H2R task execution by evaluating the performance of a noncollaborative task execution scheme, in which the selected host robot executes the full H2R task, and the collaborative/joint H2R task execution, in which the selected host robot performs only the sensing subtask while the selected collaborative node executes the computation subtask via computation offloading.

The main contributions of this paper are threefold. First, we present a suitable host robot selection policy for H2R task allocation taking different parameters such as robot availability, remaining energy, and task execution time into account. Second, to improve the task response time and energy consumption of robots, we propose an efficient collaborative node selection scheme for computation offloading. Third, we investigate a unified resource management scheme that is able to handle coexisting conventional broadband traffic and computation offloaded data traffic.

The remainder of this paper is structured as follows. Section II describes our proposed FiWi multirobot network architecture, unified resource management, and H2R task allocation scheme in technically greater detail. Section III presents our developed analytical model. Section IV reports on our obtained results and findings. Finally, Section V concludes this paper.

II. FiWi Multirobot Network Infrastructure for H2R Task Allocation

A. Network Architecture

In this section, we redesign the generic FiWi network architecture introduced in [1] for coordinating the local and nonlocal allocation of H2R task that consist of both sensing and computation subparts, whereby humans, robots, and agents actively participate in the task allocation process, as shown in Fig. 1. We exploit the central cloud, cloudlets, and neighboring robots as collaborative nodes for computation subtask offloading. Note that the generic FiWi network architecture proposed in [1] was designed only for H2H communications and did not examine any H2R task allocation and computation offloading capabilities.

In our proposed collaborative computing-based FiWi multirobot network architecture, the optical fiber backhaul consists of an IEEE 802.3av 10 Gb/s EPON with an optical fiber range of 10–100 km between the central optical line terminal (OLT) and the remote optical network units (ONUs). The OLT is located at the central office and connects to three different subsets of ONUs via a typical tree-and-branch topology. The first subset of ONUs provides services to a single or multiple attached fixed (i.e., nonmobile) wired subscribers via FTTx access, e.g., fiber-to-the-home/business. To provide an interface with the wireless mesh network (WMN), the second subset of ONUs is attached to an IEEE 802.11s mesh portal point (MPP), referred to as ONU-MPP. The WMN consists of wireless mesh points (MPs) and mesh access points (MAPs), whereby intermediate MPs relay traffic between MPPs and MAPs. Each MAP serves associated mobile users (MUs) and robots within its wireless coverage area. Note that the integrated ONU-MPP is realized by using

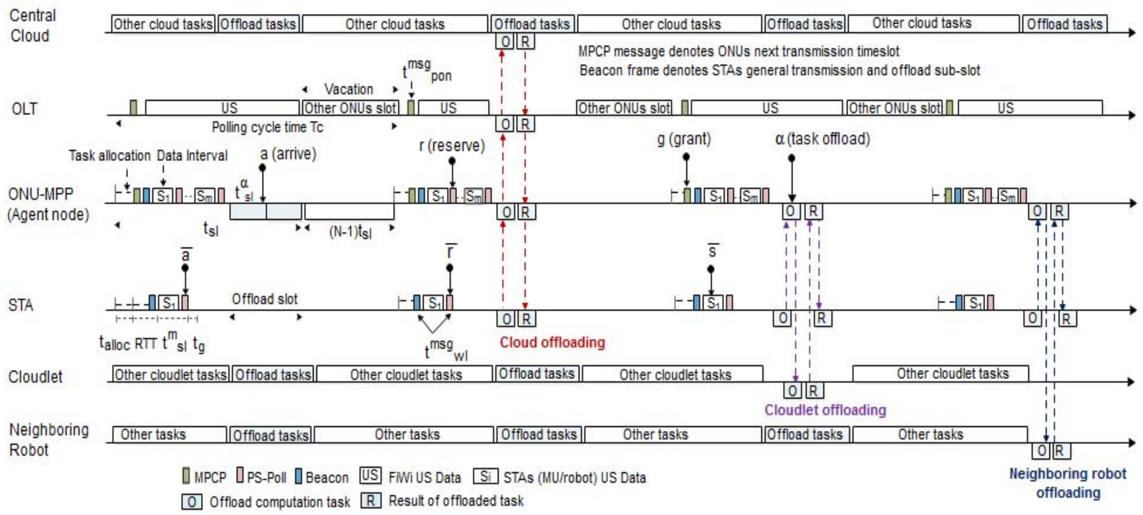


Fig. 2. Proposed resource management scheme.

so-called radio-and-fiber technologies with medium access control protocol translation taking place at the optical-wireless interface [20]. The third subset of ONUs, each connecting to an LTE enhanced nodeB (eNB) base station, giving rise to ONU-eNB, in order to provide 4G cellular services to MUs [21]. The central cloud servers are connected to the OLT via dedicated fiber links. In addition, to provide MEC services to MUs and robots at the edge of our integrated FiWi multirobot network, cloudlet servers are placed at the optical-wireless network edge and connected to separate ONU-MPPs or ONU-eNBs via dedicated fiber links.

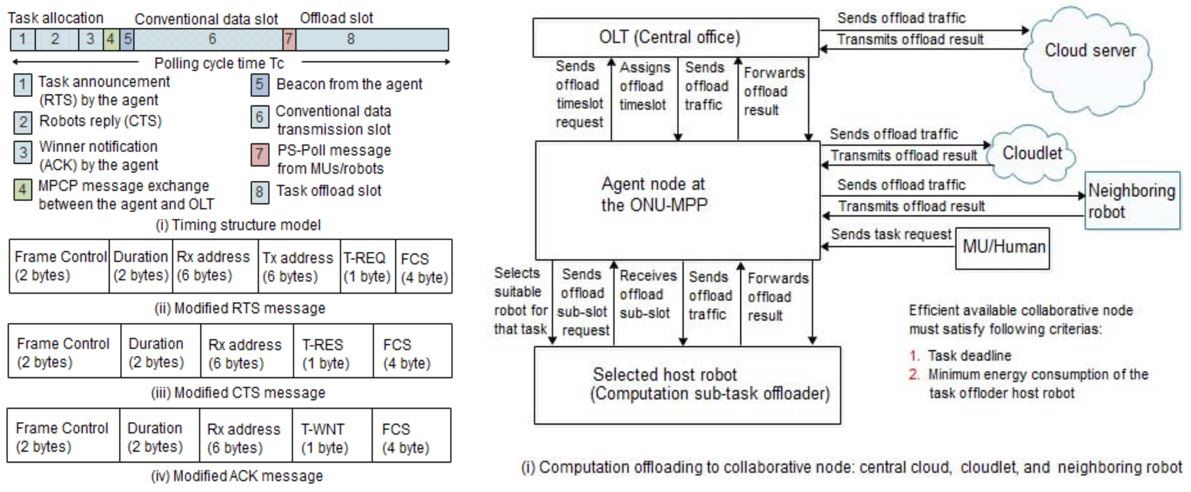
In the following, we focus on WLAN coverage only, whereby MUs and robots connect to the ONU-MPPs via IEEE 802.11ac WLAN, and leave cellular coverage for future work.

B. Resource Management Scheme

1) General Operation: Similar to [20], our proposed resource management scheme uses a two-layer time division multiple access (TDMA)-based operation in both optical and wireless subnetworks, as illustrated in Fig. 2. However, our proposed resource management scheme differs from [20] in several important ways. First, in the wireless part, we split the overall timing structure into three parts: i) initial robot selection for task allocation; ii) assigned time slot for associated users' conventional broadband traffic; and iii) computation offloaded data transmissions. Second, apart from the central cloud and local cloudlets, we consider available neighboring robots for computation subtask offloading. Third, our resource management scheme flexibly accommodates H2R task execution both with and without computation offloading onto a collaborative node.

To better understand the basic operation during a polling cycle, Fig. 3(a) illustrates the proposed resource allocation and control signal exchange process, which operates as follows.

- During the initial task allocation phase, the agent located at the ONU-MPP exchanges three control messages (`RTS`, `CTS`, and `ACK`) with its associated robots to select one suitable robot for each H2R task (to be described in greater detail in Section II-C).
- The resource management operation in the optical fiber backhaul uses IEEE 802.3av multipoint control protocol (MPCP) messages (`REPORT` and `GATE`), whereby the `REPORT` message is used by ONU-MPPs to report their upstream (US) transmission demands to the OLT and the `GATE` message is used by the OLT to inform ONU-MPPs about their US transmission windows (i.e., start time and duration) for the next polling cycle. The `REPORT` message contains the ONU-MPPs' US bandwidth requests in terms of buffer backlogs expressed in time units. In this paper, the traditional `REPORT` message is extended by using its reserved bits (32 bits) for carrying the computation offloading time slot request information, which contains the time instant up to which a given ONU-MPP can schedule its associated wireless users' computation offloaded data transmissions. The computation offloading bandwidth request information embedded in the `REPORT` message is used by the OLT to assign ONU-MPPs computation offloading transmission time slots in the next polling cycle. In the wireless part, the resource management operation is realized via the exchange of IEEE 802.11ac WLAN frames (i.e., `Beacon` and `PS-Poll`), whereby associated users (MUs/robots) report their bandwidth requests by sending an extended `PS-Poll` message to their corresponding ONU-MPP. The `PS-Poll` frame is extended by using its pad/reserved bits to include an offload flag bit. The offload flag bit is embedded in the `PS-Poll` frame to inform the ONU-MPP about the MUs/robots' computation offloading time slot requests. After receiving the `GATE` message from the OLT, the ONU-MPP assigns conventional broadband US traffic and computation offloaded data transmission opportunities to its associated users, resets its clock time, and then sends the broadcast `Beacon` frame to inform all associated MUs/robots about their US transmission time slots (i.e., start time and duration).


Task allocation
Conventional data slot
Offload slot
1 2 3 4 5 6 7 8
Polling cycle time Tc
1 Task announcement (RTS) by the agent
2 Robots reply (CTS)
3 Winner notification (ACK) by the agent
4 MPCP message exchange between the agent and OLT
5 Beacon from the agent
6 Conventional data transmission slot
7 PS-Poll message from MUs/robots
8 Task offload slot
(i) Timing structure model
Frame Control (2 bytes) | Duration (2 bytes) | Rx address (6 bytes) | Tx address (6 bytes) | T-REQ (1 byte) | FCS (4 byte)
(ii) Modified RTS message
Frame Control (2 bytes) | Duration (2 bytes) | Rx address (6 bytes) | T-RES (1 byte) | FCS (4 byte)
(iii) Modified CTS message
Frame Control (2 bytes) | Duration (2 bytes) | Rx address (6 bytes) | T-WNT (1 byte) | FCS (4 byte)
(iv) Modified ACK message
OLT (Central office)
Sends offload traffic
Transmits offload result
Cloud server
Sends offload timeslot request
Assigns offload timeslot
Sends offload traffic
Forwards offload result
Sends offload traffic
Transmits offload result
Cloudlet
Agent node at the ONU-MPP
Sends offload traffic
Transmits offload result
Neighboring robot
Sends task request
MU/Human
Selects suitable robot for that task
Sends offload sub-slot request
Receives offload sub-slot
Sends offload traffic
Forwards offload result
Selected host robot (Computation sub-task offloader)
Efficient available collaborative node must satisfy following criterias:
1. Task deadline
2. Minimum energy consumption of the task offloder host robot
(i) Computation offloading to collaborative node: central cloud, cloudlet, and neighboring robot

(a) (b)

Fig. 3. (a) Timing structure and control frame format. (b) Operational steps of computation offloading process.

- Subsequently the ONU-MPP receives the US transmissions from its associated users and forwards them to the OLT. At the same time, the ONU-MPP receives its intended downstream (DS) frames from the OLT and forwards them to its associated users. Note that each MU/robot sends/receives its US/DS data traffic to/from the ONU-MPP during its assigned time slot.

2) Computation Offloading Operation:

- In this paper, the transmission opportunity for computation offloading is kept separate from conventional broadband transmissions to permit both broadband and computation offloading operations within a polling cycle. Fig. 3(b) depicts the computation offloading process to a collaborative node, which might be a remote cloud server, decentralized cloudlet, or neighboring robot. Initially, the MU sends her H2R task execution request to the agent located at the ONU-MPP. The agent then selects a suitable host robot for the H2R task that contains both sensing and computation subparts via our task allocation algorithm (to be described in greater detail in Section II-C). Next, after completing the sensing subtask (e.g., image capturing), provided that the selected host robot requires assistance from a collaborative node (central cloud, cloudlet, or neighboring robot) to process the remaining computation subtask (e.g., image detection), it sends a computation offloading request to the ONU-MPP in an extended `PS-Poll` message.
- After receiving the computation offloading request from a given host robot, the ONU-MPP selects where (i.e., cloud node or neighboring robot) to offload the computation subtask onto subject to given computation subtask offloading requirements (see Section II-C) and sends an extended `REPORT` message to the OLT, which embeds the computation offloading request. Once the ONU-MPP receives the `GATE` message from the OLT that contains the ONU-MPPs' conventional broadband and computation offloading time slot map, it immediately schedules the host robots' computation offloading opportunities. The ONU-MPP then notifies all host robots about their computation offloading time slot information via a broadcast `Beacon` message (i.e., computation offloading time slot start time and duration). A given host robot then transmits the computation subtask data frame (see Section III-A for details) to the ONU-MPP via its assigned offloading time slot. After receiving the computation subtask input data frame from the task offloading host robot, the ONU-MPP forwards them to the selected collaborative node (central cloud/cloudlet/neighboring robot) for further processing.
- In case of cloudlet/neighboring robot offloading, the ONU-MPP sends the computation subtask input data frame to a cloudlet/neighboring robot via the fiber/wireless link for processing. Once the ONU-MPP receives the results of the computation subtask from the cloudlet/neighboring robot, it immediately sends them to the task offloading host robot. For central cloud offloading, the ONU-MPP sends the computation subtask input data frame to the OLT. Then, after receiving the computation subtask input data, the OLT transfers them to the central cloud. The OLT receives the computation subtask results from the central cloud after processing and sends them to the ONU-MPP. Once the ONU-MPP receives the computation subtask results from the OLT, it immediately forwards them to the corresponding host robot.

C. Proposed Task Allocation Algorithm

We consider the following two different task execution schemes: i) a noncollaborative scheme, where the suitable host robot executes the full task and ii) a collaborative scheme, where the suitable host robot and collaborative node (central cloud, cloudlet, or neighboring robot) conduct the sensing and computation subtask, respectively. Our proposed task allocation algorithm, which performs both suitable host robot and collaborative node selection, comprises the following four steps.

- *Step 1:* Initially, an MU sends her H2R task request message (T-REQ) to the agent node during her assigned US transmission time slot containing the following information: task location, task type, remaining energy threshold, and task deadline.
- *Step 2:* When the agent at the ONU-MPP receives the T-REQ from the MU, it first checks which robots in its wireless coverage area satisfy the given availability, energy threshold to conduct the task, and task execution deadline requirements. Toward this end, the agent first broadcasts a task announcement message (RTS) to all nearby robots. The RTS frame is extended by using its pad/reserved bits in order to include additional task request information (T-REQ), similar to [22]. After the reception of a RTS frame, the available robots in that network send reply messages (T-RES) embedded in CTS frames to the agent containing the following information: remaining energy, location, moving and processing speed, and precalculation of task execution time of each robot. After checking each robot's reply, the agent selects a suitable host robot according to the following criteria: robot availability, remaining energy, and minimum task execution time. The selected host robot is notified by the agent via a winner notification (T-WNT) embedded in the ACK message. The modified RTS, CTS, and ACK frames are depicted in Fig. 3(a).
- *Step 3:* The selected host robot first executes the sensing subpart of the task. If the host robot sends a computation subtask offloading request to the agent, the agent selects a suitable collaborative node for the computation subtask execution (remaining subpart).
- *Step 4:* The agent checks the computation subtask response time, resource availability, and energy consumption of all collaborative nodes (i.e., central cloud, cloudlet, and neighboring robots with minimum execution time and energy consumption value). The agent then selects the most suitable collaborative node for computation subtask execution based on the following criteria: i) computation subtask response time of the collaborative node is less than or equal to the computation subtask deadline; ii) sufficient resource availability; and iii) minimum energy consumption of task offloading host robot among all collaborative nodes.

The pseudocode (see Algorithm 1) of our proposed task allocation algorithm executed by the ONU-MPP (agent node) is described below in a more formal way by defining the various parameters used in our analytical model. First, the agent node broadcasts the H2R task request message RTS (total number of arrived tasks is n and available robots is m) to all available robots under the ONU-MPP and receives the robots' response CTS messages from all available robots (lines 1 and 2). Next, the agent node extracts the robots' residual energy information and precalculation of task execution time information from each robot's response message and allocates the H2R task to the robot that has sufficient residual energy to process the task ($e_r^j \geq e_{\text{th}}$) and satisfies the minimum task execution time requirements (lines 3–5). Then, if the selected host robot executes the sensing subtask of the full task and sends a PS-Poll message to agent node for collaborative node selection to perform the computation subtask execution, the agent checks the available resource type of all collaborative nodes (lines 6 and 7). If only one collaborative node (central cloud, cloudlet, or neighboring robot) satisfies the given task execution deadline, the agent at the ONU-MPP offloads the computation subtask onto that collaborative node. If more than one collaborative node is able to meet the given computation subtask execution deadline, the agent offloads the computation subtask onto the most suitable collaborative node based on its minimum energy consumption (lines 8–13). Conversely, if all available collaborative nodes fail to satisfy the given computation subtask deadline, the computation subtask is executed locally at the initially selected host robot (lines 14 and 15). Further, if the selected host robot fails to execute the computation subtask, a new host robot needs to be selected for the computation subtask execution (lines 16–18) and the process is repeated starting at Step 1.

III. Analytical Model

In this section, we first briefly discuss the assumptions of our analytical model. We then evaluate the performance of our suitable host robot selection scheme in terms of task allocation delay and task execution time. Afterwards, we analyze the end-to-end local/nonlocal task allocation and computation subtask offloading delay. Finally, we assess the performance of our proposed collaborative and noncollaborative task execution schemes in terms of task response time and energy consumption efficiency.

A. Assumptions

The H2R application is assumed to consist of one fine-grained task that includes both sensing (e.g., image capturing) and computation subtask (e.g., image/face detection), similar to [20] and [23]. Each robot is able to execute a single task at any given time. Unlike MUs, robots are assumed to be static, though they are able to move and change their position toward task location at low speed (e.g., pedestrian speed). Furthermore, robots are heterogeneous with regard to their remaining energy e_r^j (given in Joule), moving speed v_j (m/s), CPU clock speed μ_j (MHz), and position (x_j, y_j). The request from a given MU for a particular task i includes the task location (x_i, y_i), total or full task input size ($l_i = l_s + l_u$) that contains both sensing (l_s) and computing (l_u) subtask input data size (kB), total CPU clock cycles (megacycles) required to process the full task input data size, including both sensing and computation subtask CPU cycles ($t_{\text{cpu}} = s_{\text{cpu}} + c_{\text{cpu}}$), required robot energy threshold for each task (e_{th}), and deadline (t_d) to complete the full H2R task. We also assume that the output of the sensing subtask is the input of the computation subtask. The computation subtask considered for offloading is defined as follows: $c_i \triangleq (c_{\text{cpu}}, t_c^d, m_r, l_u, l_r)$, where c_{cpu}, t_c^d, m_r, l_u, and l_r indicate the number of required CPU cycles to

Algorithm 1 Task Allocation Algorithm

```
Considerations: Number of task arrives ($n$), number of available robots before task allocation ($m$), residual energy of robots ($e_r^j$), required energy threshold to process the task ($e_{th}$), required CPU cycles to process the sensing ($s_{cpu}$) and computation ($c_{cpu}$) subtask, full task execution time of robot ($t_j$), computation sub-task response time of central cloud ($t_c^{cl}$), cloudlet ($t_c^{ct}$), neighboring robot ($t_c^o$), and deadline ($t_c^d$), energy consumption of host robot considering own ($e_c^j$), cloud ($e_c^{cl}$), cloudlet ($e_c^{ct}$), and neighboring robot computation sub-task execution ($e_c^o$).
1:  for $i = 1$ to $n$ do
2:    for $j = 1$ to $m$ do
3:      The agent node at the ONU-MPP will check the residual energy ($e_r^j$) and pre-calculation of task execution time ($t_j$) of participating robots in that network via exchange of modified RTS and CTS messages
4:      if $e_r^j \geq e_{th}$ then
5:        Allocate task $i$ to host robot $j$ with minimum task execution time $t_j$
6:        The selected host robot executes the sensing sub-task ($l_s$) of the full task and sends a PS-Poll message to agent for computation offloading
7:        The agent checks the availability and resource type of central cloud ($cl$), cloudlet ($ct$), and neighboring robot ($o$) for computation sub-task ($s_{cpu}$) execution
8:        if $e_c^{cl} < e_c^{ct} \leq e_c^j$ & $e_c^{cl} < e_c^o \leq e_c^j$ & $t_c^{cl} \leq t_c^d$ then
9:          Offload the computation sub-task to central cloud
10:       else if $e_c^{ct} < e_c^{cl} \leq e_c^j$ & $e_c^{ct} < e_c^o \leq e_c^j$ & $t_c^{ct} \leq t_c^d$ then
11:         Offload the computation sub-task to cloudlet
12:       else if $t_c^o \leq t_c^d$ then
13:         Offload the computation sub-task to a suitable neighboring robot in that network
14:       else if $t_c^{cl} \geq t_c^d$ & $t_c^{ct} \geq t_c^d$ & $t_c^o \geq t_c^d$ then
15:         Execute the computation sub-task ($c_{cpu}$) locally in the selected host robot (see step 5)
16:       end if
17:     else
18:       Go to step 1
19:     end if
20:   end for
21: end for
```

process the computation subtask, deadline, required memory, input, and output size of the computation subtask, respectively.

B. Task Allocation Delay

The calculation of the average task allocation delay involves the transmission times of several control messages that are exchanged between the ONU-MPP (agent) and robots, namely task announcement via an extended RTS (t_{rts}), robot response via an extended CTS (t_{cts}), and winner notification via an extended ACK (t_{ack}) message, similar to the IEEE 802.11 DCF analysis in [24]. Hence, the probability that a given robot sends a response message in a random time slot is given by $p_k = (2/k + 1)$, where k denotes the constant backoff window size. For a total number m of robots participating in the task allocation, the probability that at least one robot transmits a response in a random time slot equals $p_{\text{tr}} = 1 - (1 - p_k)^m$. Thus, the probability that a single robot transmits a response in a time slot is then obtained as $p_{\text{cs}} = (mp_k(1 - p_k)^{m-1}/p_{\text{tr}})$.

For computing the average robot selection delay per task (t_{si}), we assume that the agent sends an RTS message to robots after waiting for DCF interframe space (t_{DIFS}), while robots send a CTS message after waiting for short interframe space (t_{SIFS}). After t_{SIFS}, the agent sends an ACK message to the selected robot. By using the values of t_{DIFS}, t_{SIFS}, t_{rts}, t_{cts}, and t_{ack}, both successful ($t_{sa} = t_{\text{DIFS}} + t_{\text{rts}} + mt_{\text{SIFS}} + mt_{\text{cts}} + t_{\text{SIFS}} + t_{\text{ack}}$) and unsuccessful task allocation time periods ($t_{ua} = t_{\text{DIFS}} + t_{\text{rts}} + mt_{\text{SIFS}} + mt_{\text{cts}}$) can be evaluated. Consequently, the average robot selection delay per task (t_{si}) is equal to $t_{si} = (p_e\sigma + p_s t_{sa} + p_u t_{ua}/p_s)$, where $p_e = (1 - p_{\text{tr}})$, $p_s = p_{\text{tr}}p_{\text{cs}}$, and $p_u = p_{\text{tr}}(1 - p_{\text{cs}})$ denote the empty, successful, and unsuccessful transmission probabilities in a time slot, respectively, and σ denotes the length of an idle time slot.

Next, we calculate the saturation throughput of our network. Assuming that n is the total number of task requests arriving at a given agent and m is the number of available robots in the network with sufficient remaining energy, the robot availability probability per task p_{avg} can be approximated by $p_{\text{avg}} = \min(1, (m/n))$. Further, by taking into account t_{si} and p_{avg}, the total task allocation delay (t_{alloc}) can be calculated as $t_{\text{alloc}} = t_{si}(n \cdot p_{\text{avg}})$. Additionally, if the total task allocation duration (t_{alloc}) is known, the total number of suitable robots (n_r) selected during task allocation is given by $n_r = (t_{\text{alloc}}/t_{si})$. Similarly, by using n, p_{avg}, and n_r, the total number of allocated tasks (n_a) during task allocation is given by $n_a = \sum_{i=1}^{n} \min(i, n_r) \cdot \binom{n}{i=1} (p_{\text{avg}})^i (1 - p_{\text{avg}})^{n-i}$. Thus, given the total data transmission duration (t_{total}), number of allocated tasks (n_a), and task allocation duration (t_{alloc}), the saturation throughput (s_{th}) is obtained as follows:

$$s_{\text{th}} = \frac{n_a(t_{\text{total}} - t_{\text{alloc}})}{t_{\text{total}}}. \tag{1}$$

C. Task Execution Time Without Offloading

The task execution/response time for different robots (t_j) is a crucial performance metric in the suitable host robot selection process that consists of the following four delay components: task allocation delay (t_{alloc}), time delay to reach the task location (t_r^j), time delay to process the sensing subtask (t_s^j), and time delay to process the computation subtask (t_c^j). Thus, t_j is given by

$$t_j = t_{\text{alloc}} + t_r^j + t_s^j + t_c^j = t_{\text{alloc}} + \frac{d_{ij}}{v_j} + \frac{s_{\text{cpu}}}{\mu_j} + \frac{c_{\text{cpu}}}{\mu_j} \tag{2}$$

where d_{ij}, v_j, μ_j, s_{cpu}, and c_{cpu} denote the distance between task i and location of robot j, moving and processing speed of robot j, required CPU cycles for sensing and computation subtask, respectively. The distance between task and robot location (d_{ij}) is calculated via the Euclidean distance as follows: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where (x_i, y_i) and (x_j, y_j) denote task i's and robot j's position in the 2-D space, respectively. For local (i.e., MUs and robots are associated with the same ONU-MPP) and nonlocal task (i.e., MUs and robots are associated with different ONU-MPPs) allocation, the calculation of the total task execution time considers the end-to-end local (t_{alloc}^l) and nonlocal task allocation delay

($t_{\text{alloc}}^{\text{nl}}$) instead of t_{alloc}. By accounting for both t_{alloc}^{l} and $t_{\text{alloc}}^{\text{nl}}$ (see Section III-D), the end-to-end local (t_{local}^{j}) and nonlocal (t_{nonlocal}^{j}) task execution time for selected host robots are given by $t_{\text{local}}^{j} = t_{\text{alloc}}^{l} + t_r^j + t_s^j + t_c^j$ and $t_{\text{nonlocal}}^{j} = t_{\text{alloc}}^{\text{nl}} + t_r^j + t_s^j + t_c^j$, respectively.

D. FiWi End-to-End Task Allocation and Offloading Delay

In this section, we present the end-to-end delay analysis for both local and nonlocal task allocation based on the US (from ONU-MPP to OLT) and DS (from OLT to ONU-MPP) frame transmission delay model in [21]. Note, however, that we had to modify the analytical model in [21] in order to accommodate our proposed resource allocation process, as explained in the following.

As shown in Fig. 2, the ONU-MPP time cycle (T_c) consists of the task allocation delay (t_{alloc}), MPCP message duration ($t_{\text{pon}}^{\text{msg}}$), conventional broadband traffic transmission (t_{sl}^{m}), computation offloading data transmission (t_{sl}^{α}), guard band (t_g), reservation ($V = t_{\text{wl}}^{\text{msg}} + t_g$), and vacation period. Recall that the considered FiWi network serves N ONUs and M users (humans and robots) at each ONU-MPP. We assume that H2R packets arrive at the ONU-MPP with an aggregate arrival rate of $\overline{\lambda}$ according to a Poisson process and M/G/1 queue. The random packet service time is given by the first moment ($\overline{X}$). Further, the aggregate H2R traffic load is equal to $\rho^{\text{h2r}} = \overline{\lambda X}$. During each cycle time T_c, the vacation duration equals $(N-1)t_{sl}$ and $\text{RTT} = 2t_{\text{prop}}$ denotes the round-trip time between OLT and ONU-MPPs. The nondata traffic transmission time during T_c is equal to $N(\text{MV} + t_{\text{alloc}} + t_{\text{pon}}^{\text{msg}})$. Thus, T_c is calculated as follows:

$$T_c = \frac{N(\text{MV} + \text{RTT} + t_{\text{alloc}} + t_{\text{pon}}^{\text{msg}})}{1 - \rho^{\text{h2r}}}. \tag{3}$$

Recall from above that an MU sends her US bandwidth request via a `PS-Poll` message. If the H2R task request is generated at the MU after the current `PS-Poll` message, it waits during the first delay component T_c (polling cycle time) for the next `PS-Poll` message to report to the ONU-MPP (agent) about her US subslot request (see Fig. 2). After the MU sends the task request frame to the ONU-MPP during the next time cycle subslot ($t_{sl}^{m} \geq t_{\text{wl}}^{\text{msg}}$), the total queuing delay time is equal to $2T_c - t_{\text{wl}}^{\text{msg}}$. The other two delay components associated with the US frame transmission are the US frame service time at the ONU-MPP ($\overline{X}_u \leq x_{\max}$) and the propagation delay (t_{prop}). Thus, the maximum US frame transmission delay (t_u) is given by $t_u = 2T_c - t_{\text{wl}}^{\text{msg}} + t_{\text{prop}} + x_{\max}$. Next, we compute the DS frame delay. If the OLT receives the MU's task request frame after the `GATE` message, the frame has to wait for $T_c - t_{sl}$. The other delay components are the DS frame service time delay at the OLT ($\overline{X}_d \leq x_{\max}$) and the associated propagation delay (t_{prop}). Given that $t_{sl} \geq 2t_{\text{prop}} + 2t_{\text{pon}}^{\text{msg}}$, the DS frame transmission delay (t_d) is equal to $t_d = T_c - 2t_{\text{pon}}^{\text{msg}} - t_{\text{prop}} + x_{\max}$.

By using US and DS frame delays, we are able to measure the end-to-end local and nonlocal task allocation delay as follows. If the MU requests a nonlocal task allocation, the associated ONU-MPP first sends the US task request frame to the OLT. The OLT then broadcasts the task request frame to all ONU-MPPs, whereby only the intended ONU-MPP processes the task request frame and initiates the nonlocal task allocation process. For the calculation of the nonlocal task allocation delay ($t_{\text{alloc}}^{\text{nl}}$), both US and DS frame transmission delays are required along with the task allocation delay (t_{alloc}). Thus, we have $t_{\text{alloc}}^{\text{nl}} = t_u + t_d + t_{\text{alloc}}$. Conversely, if the task request is local, the agent at the associated ONU-MPP receives the task request frame from the MU and starts the task allocation process immediately. Thus, for the calculation of the local task allocation delay (t_{alloc}^{l}), only the US frame transmission delay (t_u) is required along with the t_{alloc}. As a result, t_{alloc}^{l} is obtained as $t_{\text{alloc}}^{l} = t_u + t_{\text{alloc}}$.

Next, we analyze the maximum offload packet delay that incurs during the computation subtask offloading process. By assuming that a computation offload subslot is assigned to all devices M during each time cycle T_c and the collaborative nodes' traffic load is equal to ($\rho^{\text{cl}}, \rho^{\text{ct}}, \rho^{o}$), the computation offload subslot duration (t_{sl}^{α}) can be expressed as $t_{sl}^{\alpha} = (\rho^{\text{cn}} \cdot T_c/M)$, where subscript cn can be either the central cloud (cl), a local cloudlet (ct), or a neighboring robot (o), respectively. As shown in Fig. 2, if the computation offload request (a) is generated at the host robot after a `PS-Poll` message, it experiences the maximum computation offload packet buffering delay. For an upcoming transmission opportunity, the computation subtask offload packet needs to wait.

Note that the end-to-end maximum computation subtask offload packet delay consists of four delay components, as shown in Fig. 2. The first delay component is the time difference between the computation offload request arrival (a) and the transmission of offload reservation request (r) via a `PS-Poll` frame ($\overline{d}_1 = (M-1)t_{sl}^{\alpha} + (N-1)t_{sl} + t_{sl}^{m} + t_{\text{alloc}} + t_{\text{pon}}^{\text{msg}} + t_{\text{wl}}^{\text{msg}}$). The second delay component is the time gap between the transmission of offload reservation request (r) and the reception of offload grant (g) or `MPCP` frame ($\overline{d}_2 = (M-1)t_{sl}^{m} + Mt_{sl}^{\alpha} + (N-1)t_{sl} + t_{\text{alloc}} + t_{\text{pon}}^{\text{msg}}$). The third delay component is the time gap between the last grant (g) message and the start time of the offload (α) slot ($\overline{d}_3 = t_{\text{wl}}^{\text{msg}} + Mt_{sl}^{m}$). By summing up the first three delay components ($\overline{d}_1, \overline{d}_2, \overline{d}_3$), we observe from Fig. 2 that the maximum computation offload packet delay for cloudlet ($\overline{d}_{\text{ct}} \leq 2T_c$) and neighboring robot offloading ($\overline{d}_o \leq 2T_c$) is equal to $2T_c$. Whereas for central cloud offloading, the fourth delay component ($\overline{d}_4 = 2t_{\text{prop}}$) along with the first three components of the maximum offload packet delay calculation is given by $\overline{d}_{\text{cl}} \leq 2T_c + 2t_{\text{prop}}$, whereby ($t_{\text{prop}}$) denotes the one-way propagation delay between OLT and central cloud.

E. Performance Analysis of Computation Offloading and Collaborative Task Execution

In this section, we analyze the performance of noncollaborative and collaborative task execution schemes in terms of task response time and energy consumption efficiency. The first part of this section presents the calculation of computation subtask response time and energy consumption of a host robot for different offloading schemes to decide whether or

TABLE II
NOTATION

Symbol	Definition	Value/Unit
d_{ij}, e^i_j, e_{th}	Distance (robot to task location), initial energy of robots, and required energy threshold	1-10 m, 200 KJ, 50 J
p_c, p_v, p_s	Robot average power consumption for processing, moving, and sensing sub-task	.5 W, .7 W, .5 W
l_i, l_s, l_u, l_r	Total task input (full), sensing sub-task input, computation sub-task input, and output data size	KB (vary)
$v_j, t_{cpu}, s_{cpu}, c_{cpu}$	Robot moving speed, total task, sensing, and computation sub-task CPU cycles (workload)	.1-1 m/s, Mega cycles (vary)
m_{ct}, m_o, m_r	Available memory space of cloudlet, neighboring robot, and required size for offloading	MB (vary)
$\mu_j, \mu_{cl}, \mu_{ct}, \mu_o$	CPU clock frequency of host robot, central cloud, cloudlet, and neighboring robot	MHz (vary)
$t^{cl}_{prop}, t^{ct}_{prop}, t^{o}_{prop}$	Total propagation delay for cloud, cloudlet, and neighboring robot offloading	.6, .4, .4 ms
p_{idle}, p_u, p_r	Average energy consumed by robot during idle, data transmission, and reception	.001 W, .1 W, .05 W
$b_{wl}, b_{cl}, b_{ct},$	Transmission capacity of wireless and fiber link for cloud, cloudlet offloading	6900 Mb/s, 10 Gb/s, 10 Gb/s

not computation subtask offloading to a collaborative node is useful for the task offloading robot.

The computation subtask response time for central cloud offloading (t_c^{cl}) consists of two parts: offloading delay ($t_{\text{ofl}}^{\text{cl}}$) and computation subtask processing delay at the central cloud ($t_{\text{ex}}^{\text{cl}} = (c_{\text{cpu}}/\mu_{\text{cl}})$). The central cloud offloading delay considers the uplink communication delay ($t_{\text{cl}}^u = (l_u/b_{\text{wl}}) + (l_u/b_{\text{cl}}) + (l_u/b_{\text{cl}})$), downlink communication delay ($t_{\text{cl}}^r = (l_r/b_{\text{wl}}) + (l_r/b_{\text{cl}}) + (l_r/b_{\text{cl}})$), and total propagation delay, including both air (wireless part) and fiber (optical part) propagation delays. The uplink delay (t_{cl}^u) consists of the time required to transfer computation subtask offloading packets from the host robot to the agent across the wireless link and further from the agent (ONU-MPP) to the OLT and to the central cloud. The downlink delay (t_{cl}^r) measures the time period required to transfer the resultant offloading packet from the central cloud to the host robot via OLT and agent, respectively. Thus, t_c^{cl} is computed as follows:

$$t_c^{\text{cl}} = t_{\text{ofl}}^{\text{cl}} + t_{\text{ex}}^{\text{cl}} = \frac{2(l_u + l_r)}{b_{\text{cl}}} + \frac{(l_u + l_r)}{b_{\text{wl}}} + t_{\text{prop}}^{\text{cl}} + \frac{c_{\text{cpu}}}{\mu_{\text{cl}}}. \quad (4)$$

Similarly, the computation subtask response time for cloudlet offloading (t_c^{ct}) comprises the offloading delay ($t_{\text{ofl}}^{\text{ct}}$) and computation subtask processing delay at the cloudlet ($t_{\text{ex}}^{\text{ct}} = (c_{\text{cpu}}/\mu_{\text{ct}})$). The cloudlet offloading delay involves the uplink ($t_{\text{ct}}^u = (l_u/b_{\text{wl}}) + (l_u/b_{\text{ct}})$) and downlink ($t_{\text{ct}}^r = (l_r/b_{\text{wl}}) + (l_r/b_{\text{ct}})$) communication delay along with the total propagation delay ($t_{\text{prop}}^{\text{ct}}$) that incurs during the offloading process. The uplink delay (t_{ct}^u) includes the offload packet transmission time from the host robot to the agent and from the agent to the cloudlet by using the wireless and fiber link, respectively. The downlink delay (t_{ct}^r) captures the resultant offload packet transfer time period from the cloudlet to the host robot via the intermediate agent. Thus, t_c^{ct} is given by

$$t_c^{\text{ct}} = t_{\text{ofl}}^{\text{ct}} + t_{\text{ex}}^{\text{ct}} = \frac{(l_u + l_r)}{b_{\text{ct}}} + \frac{(l_u + l_r)}{b_{\text{wl}}} + t_{\text{prop}}^{\text{ct}} + \frac{c_{\text{cpu}}}{\mu_{\text{ct}}}. \quad (5)$$

The computation subtask response time for neighboring robot offloading (t_c^o) also accounts for both offloading delay (t_{ofl}^o) and computation subtask processing delay at the neighboring robot ($t_{\text{ex}}^o = (c_{\text{cpu}}/\mu_o)$). The neighboring robot offloading delay (t_{ofl}^o) includes the suitable neighboring robot selection delay (t_{alloc}), uplink delay ($t_o^u = (2l_u/b_{\text{wl}})$), downlink delay ($t_o^r = (2l_r/b_{\text{wl}})$), and total propagation delay (t_{prop}^o) that occurs during the offloading process. The uplink delay (t_o^u) comprises the offload packet transfer time from the task offloading host robot to the agent and from the agent to the selected neighboring robot by using the wireless link. The downlink delay (t_o^r) consists of the resultant packet transfer time from the neighboring robot to the task-offloading host robot across the agent. Thus, t_c^o is obtained as follows:

$$t_c^o = t_{\text{ofl}}^o + t_{\text{ex}}^o = t_{\text{alloc}} + \frac{2(l_u + l_r)}{b_{\text{wl}}} + t_{\text{prop}}^o + \frac{c_{\text{cpu}}}{\mu_o}. \quad (6)$$

Next, we analyze the total task response time of the following three collaborative task execution schemes: 1) host robot-central cloud; 2) host robot-cloudlet; and 3) host robot-neighboring robot. If the sensing subtask is executed by the selected host robot and the computation subtask is offloaded onto the central cloud, by using (2) and (4) the total task response time ($t_{j,\text{cl}}$) for the host robot-central cloud-based joint task execution is given by

$$t_{j,\text{cl}} = t_{\text{alloc}} + t_r^j + t_s^j + t_c^{\text{cl}}. \quad (7)$$

If the sensing subtask is executed by the host robot and the computation subtask is offloaded onto a nearby cloudlet for execution, by using (2) and (5) the total task response time for the host robot-cloudlet ($t_{j,\text{ct}}$)-based joint execution is obtained as follows:

$$t_{j,\text{ct}} = t_{\text{alloc}} + t_r^j + t_s^j + t_c^{\text{ct}}. \quad (8)$$

Further, if the sensing subtask is performed by the host robot and the computation subtask is offloaded onto a neighboring robot, then by using (2) and (6) the total task response time ($t_{j,o}$) for the host robot-neighboring robot-based joint execution is equal to

$$t_{j,o} = 2t_{\text{alloc}} + t_r^j + t_s^j + t_c^o. \quad (9)$$

By taking the local and nonlocal task allocation delays (see Section III-D) into account, the end-to-end nonlocal ($t_{\text{nonlocal}}^{j,\text{cn}} = t_u + t_d + t_{j,\text{cn}}$) and local task response time ($t_{\text{local}}^{j,\text{cn}} = t_u + t_{j,\text{cn}}$) for the case of collaborative task execution are obtained, whereby the subscript *cn* can stand for either cl, ct, or *o*, respectively. Using both collaborative total task response time ($t_{j,\text{cn}}$) and maximum offload packet buffering delay ($\overline{d}_{\text{cn}}$), we are able to assess the maximum total task response time for the host robot-central cloud ($\bar{t}_{j,\text{cl}} = \overline{d}_{\text{cl}} + t_{j,\text{cl}}$), host robot-cloudlet ($\bar{t}_{j,\text{ct}} = \overline{d}_{\text{ct}} + t_{j,\text{ct}}$), and host robot-neighboring robot ($\bar{t}_{j,o} = \overline{d}_o + t_{j,o}$)-based joint execution schemes. Note that in the noncollaborative host robot-based task execution the maximum response time ($\bar{t}_j$) is equal to t_j since there is no offloading delay.

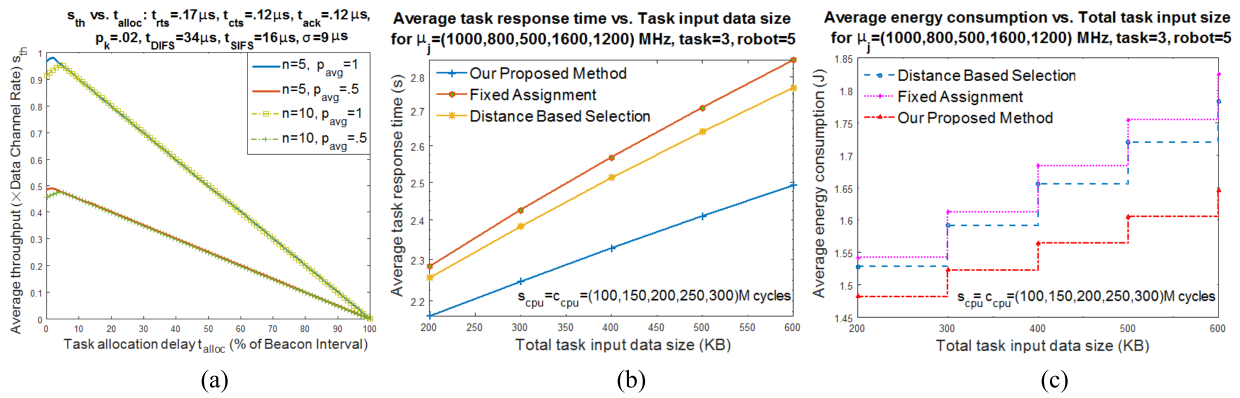


Fig. 4. (a) Average throughput variation of our proposed robot selection scheme. (b) Average task response time comparison. (c) Average energy consumption comparison of distance-based, fixed assignment-based, and our proposed robot selection scheme.

Energy consumption of host robot for computation subtask offloading onto the collaborative node (central cloud, cloudlet, or neighbor robot) consists both energy consumption for offloading communication delay and computation subtask processing delay. Next, by using (4)–(6) and Table II, the energy consumption of the host robot for computation subtask offloading onto the central cloud ($e_c^{\text{cl}} = e_{\text{ofl}}^{\text{cl}} + e_{\text{ex}}^{\text{cl}}$), cloudlet ($e_c^{\text{ct}} = e_{\text{ofl}}^{\text{ct}} + e_{\text{ex}}^{\text{ct}}$), and neighboring robot ($e_c^{o} = e_{\text{ofl}}^{o} + e_{\text{ex}}^{o}$) is obtained as follows:

$$e_c^{\text{cl}} = p_u\left(\frac{2l_u}{b_{\text{cl}}} + \frac{l_u}{b_{\text{wl}}}\right) + p_r\left(\frac{2l_r}{b_{\text{cl}}} + \frac{l_r}{b_{\text{wl}}}\right) + p_{\text{idle}}\left(\frac{c_{\text{cpu}}}{\mu_{\text{cl}}} + t_{\text{prop}}^{\text{cl}}\right) \tag{10}$$

$$e_c^{\text{ct}} = p_u\left(\frac{l_u}{b_{\text{ct}}} + \frac{l_u}{b_{\text{wl}}}\right) + p_r\left(\frac{l_r}{b_{\text{ct}}} + \frac{l_r}{b_{\text{wl}}}\right) + p_{\text{idle}}\left(\frac{c_{\text{cpu}}}{\mu_{\text{ct}}} + t_{\text{prop}}^{\text{ct}}\right) \tag{11}$$

$$e_c^{o} = p_u\left(\frac{2l_u}{b_{\text{wl}}}\right) + p_r\left(\frac{2l_r}{b_{\text{wl}}}\right) + p_{\text{idle}}\left(\frac{c_{\text{cpu}}}{\mu_o} + t_{\text{prop}}^{o}\right) + e_a^j \tag{12}$$

where e_a^j denotes the energy consumed during the task allocation process. From (10)–(12), we can calculate energy consumption of host robot for computation subtask processing delay at central cloud ($e_{\text{ex}}^{\text{cl}} = p_{\text{idle}} \cdot t_{\text{ex}}^{\text{cl}}$), cloudlet ($e_{\text{ex}}^{\text{ct}} = p_{\text{idle}} \cdot t_{\text{ex}}^{\text{ct}}$), and neighbor robot ($e_{\text{ex}}^{o} = p_{\text{idle}} \cdot t_{\text{ex}}^{o}$), where p_{idle} is the average idle energy consumption of host robot for computations subtask processing at collaborative node. Similarly, from (10)–(12), we can obtain energy consumption of host robot for central cloud ($e_{\text{ofl}}^{\text{cl}} = e_c^{\text{cl}} - e_{\text{ex}}^{\text{cl}}$), cloudlet ($e_{\text{ofl}}^{\text{ct}} = e_c^{\text{ct}} - e_{\text{ex}}^{\text{ct}}$), and neighbor robot ($e_{\text{ofl}}^{o} = e_c^{o} - e_{\text{ex}}^{o}$) offloading communication delay.

For the power consumption (p_c) and time (t_c^j) to process the computation subtask given by (2) and Table II, the energy consumption of the host robot for executing its own computation subtask (e_c^j) is equal to $e_c^j = p_c \cdot t_c^j = p_c \cdot (c_{\text{cpu}}/\mu_j)$.

In the following, we analyze the total energy consumption of the host robot for both collaborative and noncollaborative full-task execution schemes, which involves four parts. The first part of the task allocation process (e_a^j) corresponds to transmitting ($e_{tx}(l, d) = (\varepsilon_{\text{elec}} + \varepsilon_{\text{amp}} \cdot d^2) \cdot l$) and receiving ($e_{rx}(l) = \varepsilon_{\text{elec}} \cdot l$) the robot selection control packet (l bit) over a distance d, whereby $\varepsilon_{\text{elec}}$ and ε_{amp} represent the energy dissipation of the radio electronics ($\varepsilon_{\text{elec}} = 50$ nJ/bit) and transmit amplifier ($\varepsilon_{\text{amp}} = .0013$ pJ/bit), respectively [23]. Hence, e_a^j is given by $e_a^j = e_{tx}(l, d) + e_{rx}(l)$. The second part is the energy consumption (e_d^j) of selected host robot to reach the task location. For the average power consumption (p_v) and time (t_r^j) to reach the task location given in (2) and Table II, e_d^j is obtained as $e_d^j = p_v \cdot t_r^j = p_v \cdot (d_{ij}/v_j)$, where d_{ij} and v_j denote the distance between robot and task location and the speed of the moving robot, respectively. The third type of energy consumption is related to the sensing subtask. For a given average power consumption (p_s) and time (t_s^j) to process the sensing subtask, the energy consumption of the host robot for executing the sensing subtask (e_s^j) is given by $e_s^j = p_s \cdot t_s^j = p_s \cdot (s_{\text{cpu}}/\mu_j)$. Finally, the fourth type of energy consumption related to the computation subtask was computed above in (10)–(12).

By summing up the above four energy consumption parts, the total energy consumption of the robot (e_w^j) during the full-task execution is obtained as follows:

$$e_w^j = \sum_{i \in n} e_a^j + \sum_{i \in n} e_s^j + \sum_{i \in n} e_d^j + \sum_{i \in n} e_c^j + \sum_{i \in n} e_c^{\text{cn}} \tag{13}$$

where n is the number of processed tasks and the subscript cn stands for cl, ct, and o, respectively. Taking the host robot's initial energy (e_j^i) and total consumed energy (e_w^j) into account, the residual energy of a robot (e_r^j) equals $e_r^j = e_j^i - e_w^j$. Note that if the full task is executed by the host robot itself, the total energy consumption of the host robot for the resultant noncollaborative task execution is given by $e_j = e_a^j + e_s^j + e_d^j + e_c^j$. Hence, if the sensing subtask is performed by host robot itself while the computation subtask is offloaded onto a collaborative node, the energy consumption of the host robot for such a collaborative/joint task execution is equal to $e_{j,\text{cn}} = e_a^j + e_s^j + e_d^j + e_c^{\text{cn}}$.

F. Task Response Time and Energy Consumption Efficiency

As the total task response time and energy efficiency ratio are key performance metrics, we analyze both of them in this section. In our calculation of the total task response time efficiency (t_{eff}), we use the total task response time of the collaborative execution ($t_{j,\text{cn}}$) and noncollaborative host robot execution (t_j). The energy efficiency of the full-task execution (e_{eff}) is obtained from the energy consumption of the host robot for collaborative ($e_{j,\text{cn}}$) and noncollaborative task execution (e_j). The total task response time (t_{eff}) and energy consumption (e_{eff}) efficiency ratio taking both collaborative and noncollaborative task execution into account are given by $t_{\text{eff}} = (t_j - t_{j,\text{cn}}/t_j)$ and $e_{\text{eff}} = (e_j - e_{j,\text{cn}}/e_j)$, where subscript *cn* may stand for the central cloud (cl), cloudlet (ct), or neighboring robot (o), respectively.

Hence, the offload gain-overhead ratio (γ_{cn}) for computation subtask offloading to a collaborative node is given by the ratio of computation subtask offload gain ($t_c^j - t_c^{\text{cn}}$) and overhead ($\bar{d}_{\text{cn}} + t_{\text{ofl}}^{\text{cn}}$)

$$\gamma_{\text{cn}} = \frac{t_c^j - t_c^{\text{cn}}}{\bar{d}_{\text{cn}} + t_{\text{ofl}}^{\text{cn}}}. \tag{14}$$

IV. Results

In this section, we investigate the performance of our proposed suitable host robot selection, collaborative, and noncollaborative task execution schemes leveraging on the different capabilities of the central cloud, cloudlets, and robots. Table II summarizes the key system parameters and their assigned default values in compliance with previous studies [12], [18], [20], [23], [24].

A. Suitable Robot Selection

In this section, we discuss the performance of our proposed robot selection mechanism. Fig. 4(a) shows the variation of throughput for different task allocation delay, numbers of tasks, and robot availability probability per task. We compute the saturation throughput by taking the ratio of the MUs/robots data transmission time under the ONU-MPP and the total allocated time of each ONU-MPP [see (1)]. Recall that the allocated time slot duration of each ONU-MPP is divided into a task allocation phase and a data transmission phase, as shown in Fig. 3(a). Further, each task allocation to robot takes a certain amount of time due to control messages exchanged between an agent and the robots, as described above in Sections II-C and III-B. Note that when the number of H2R task allocation requests arriving at an agent is small, the task allocation phase may be carried out quickly after completing the required number of robot selections for all arriving task requests. Conversely, if the number of task requests is very high, not all tasks may be allocated to robots during the task allocation phase due to its limited (i.e., insufficient) duration. Thus, the duration of the task allocation phase needs to be set properly such that the required number of robots can be selected for all task requests arriving at the agent located in the corresponding ONU-MPP.

We observe from Fig. 4(a) that initially the average throughput increases with the task allocation delay and then levels off. At that time, the task allocation delay is sufficient to select a suitable robot for the requested task. After that, the average throughput decreases for further increasing task allocation delay. Note that for the same robot availability probability, a small number of the tasks require lower task allocation delay than that of larger numbers of tasks. This is because a small number of tasks requires the selection of a small number of robots to accomplish the task requests. Several control messages are exchanged between ONU-MPP (agent) and robots for selecting a suitable robot for each task, namely task announcement via an extended RTS, robot response via an extended CTS, and winner notification via an extended ACK message. The transmission times of these control messages determine the calculation of robot selection delay (t_{si}). Thus, by considering the total number of task requests (n), robot selection delay for each task (t_{si}), and robot availability probability for each task (p_{avg}), we observe that a small number of tasks (n) causes a lower total task allocation delay (t_{alloc}) than larger numbers of tasks, which is given by $t_{\text{alloc}} = t_{si}(n \cdot p_{\text{avg}})$. Further, a lower task allocation delay results in a higher data transmission time [see (1)], which in term increases average throughput. Moreover, for the same number of tasks, a lower robot availability probability per task leads to an inferior average throughput performance than higher robot availability probabilities.

In Fig. 4(b) and (c), we compare the average task response time and energy consumption of our proposed scheme with a traditional distance [7] and fixed assignment [8] based robot selection scheme by varying the total task input data size. Task response time for host robot own execution consists of task allocation delay, time to reach the task location, sensing, and computation subtask execution [see (2)]. Hence, host robot energy consumption for total task execution ($e_j = e_a^j + e_d^j + e_s^j + e_c^j$) is calculated by considering energy consumption for task allocation, moving to the task location, sensing, and computation subtask execution [see (13)].

As discussed earlier, our proposed scheme selects a suitable robot based on the robot's availability, remaining energy, and precalculation of minimum task execution time. Hence, the distance-based selection scheme assigns a task to the robot that offers the minimum distance to the task location. Meanwhile, the fixed assignment scheme chooses a suitable robot based on the lowest robot identifier or matching of task and robot identification number. From Fig. 4(b) and (c), we observe that the average task response time and energy consumption per task increases with the total task input data size. Importantly, both figures indicate that our proposed scheme significantly improves the average task response time and energy consumption per task than the alternative distance- and fixed assignment-based schemes. For instance, for a typical total task input data size of 600 kB, our proposed scheme achieves an average task response time per task that is 4.21% and 3.33% smaller than that of the fixed- and distance-based selection schemes, respectively. Moreover, for the same total task input data size, our proposed scheme shows an average energy consumption per task that is 3.29% and 2.61% below that of the fixed- and distance-based selection schemes, respectively.

B. Collaborative Versus Noncollaborative Task Execution

In this section, we compare the performance of the noncollaborative (i.e., without offloading) and collaborative/joint task execution schemes, whereby the sensing subtask is conducted by the selected host robot and the computation subtask is offloaded onto a collaborative node. To examine the impact

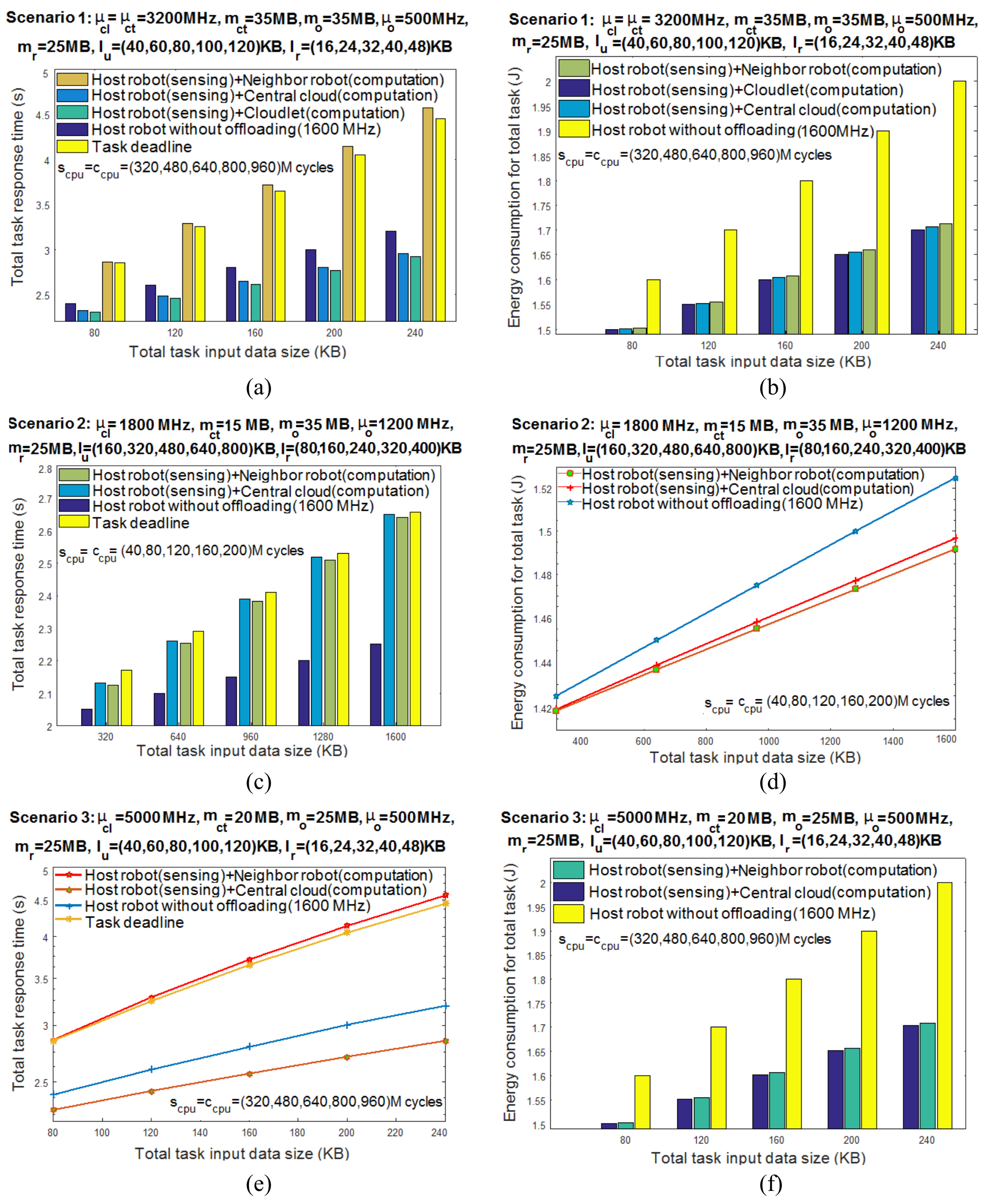


Fig. 5. Task response time and energy consumption variation of collaborative and noncollaborative task execution schemes versus total task input data size for three different scenarios.

of our proposed collaborative computing-based task execution scheme, we studied different evaluation scenarios based on different H2R task input and output data sizes, required workload (in terms of CPU cycles) to process the task, and collaborative nodes' resource conditions (i.e., processing power, available memory size, and availability), similar to [12], [18], and [20]. The parameter settings related to each particular scenario are given in Figs. 5–8. Moreover, for a particular H2R task that includes both sensing and computation subparts, four different types of task execution schemes were considered.

1) Selected host robot-based full-task execution without offloading.
2) Host robot (sensing subtask) with central cloud execution (computation subtask).
3) Host robot (sensing subtask) with cloudlet execution (computation subtask).
4) Host robot (sensing subtask) with neighboring robot execution (computation subtask).

The total task response time for the three collaborative and one noncollaborative schemes are calculated by using (7)–(9) and (2), respectively. The energy consumption of the host robot for the collaborative ($e_{j,\mathrm{cn}} = e_a^j + e_s^j + e_d^j + e_c^{\mathrm{cn}}$) and noncollaborative ($e_j = e_a^j + e_s^j + e_d^j + e_c^j$) total task execution is calculated by using (13) (see Section III-E).

Fig. 5(a) and (b) illustrates the total task response time and host robot energy consumption of the different task execution schemes for scenario 1. In this scenario 1, both central cloud and cloudlet are assumed to have the same

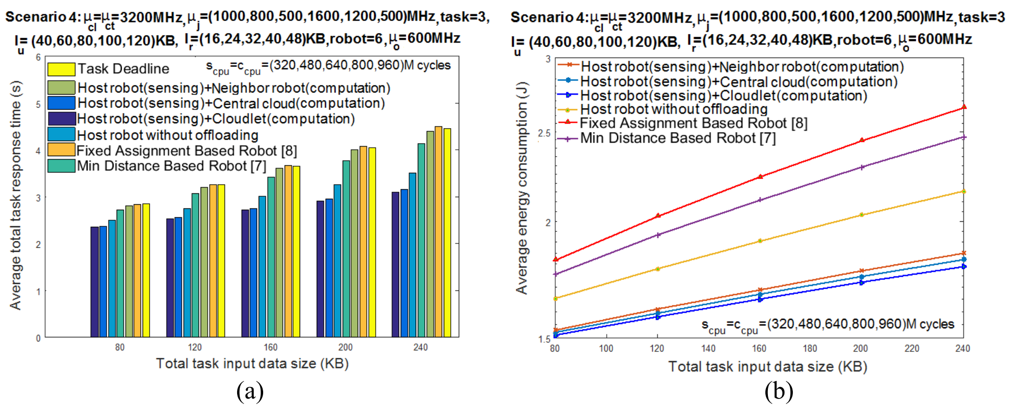


Fig. 6. Average task response time and energy consumption comparison of our collaborative (host robot-central cloud, host robot-cloudlet, and host robot-neighbor robot) and noncollaborative (host robot without offloading) schemes with existing schemes.

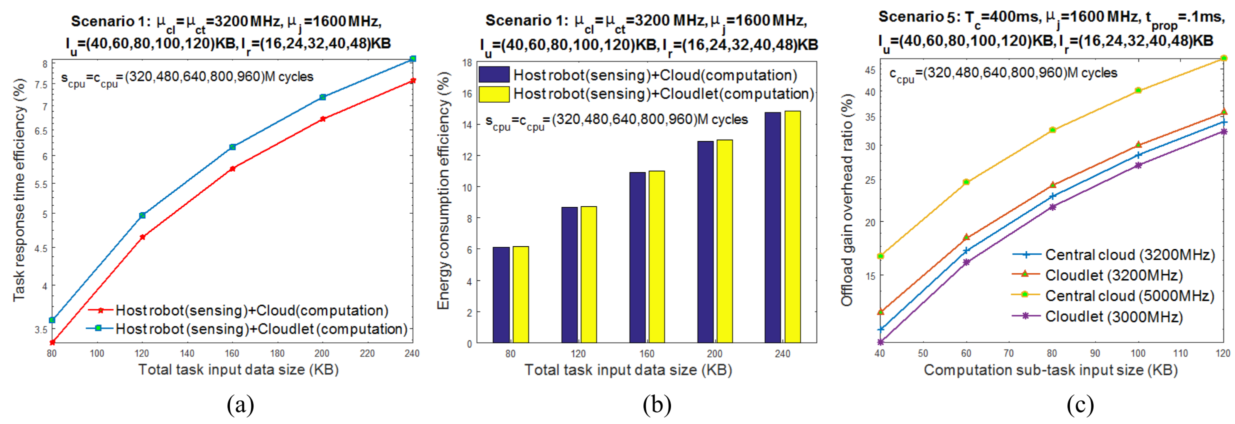


Fig. 7. (a) Task response time efficiency versus total task input data size. (b) Energy consumption efficiency versus total task input data size. (c) Offload gain overhead ratio versus computation subtask input data size.

computation capability/CPU power. The figures show that the task response time and energy consumption of host robot increase for increasing task input data size in all proposed task execution schemes. We notice that the host robot-neighboring robot-based joint task execution scheme shows a higher task response time than the host robot-central cloud scheme and fails to meet the task deadline requirement. The reason for this observation is the fact that the neighboring robot CPU power (500 MHz) is lower than the central cloud CPU power (3200 MHz). Thus, the computation subtask processing delay is much higher in the neighbor robot than that of the central cloud execution, which additionally results in a longer total task response time for the host robot-neighbor robot scheme. For instance, for a typical total task input size of 240 kB, the total task response time in the host robot-neighbor robot and host robot-central cloud scheme equals 4.56 and 2.95 s, respectively, whereas the computation subtask processing delay of the neighbor robot (t^{o}_{ex}) and central cloud ($t^{\mathrm{cl}}_{\mathrm{ex}}$) equals 1.92 and 0.3 s, respectively. Hence, the computation subtask offloading delay of the neighbor robot (t^{o}_{ofl}) and central cloud ($t^{\mathrm{cl}}_{\mathrm{ofl}}$) are equal to 0.049 and 0.056 s, respectively. However, the energy efficiency gain of the host robot-neighbor robot compared to the host robot-central cloud is negligible, less than 1%. This is because the difference between the energy consumption of the host robot for the central cloud ($e^{\mathrm{cl}}_{\mathrm{ex}} = p_{\mathrm{idle}} \cdot t^{\mathrm{cl}}_{\mathrm{ex}}$) and neighbor robot ($e^{o}_{\mathrm{ex}} = p_{\mathrm{idle}} \cdot t^{o}_{\mathrm{ex}}$) computation subtask processing delay is very small. The average energy consumption of the host robot (per second) is very low during the neighbor robot (in host robot-neighbor robot scheme) and central cloud (in host robot-central cloud scheme) computation subtask processing, e.g., $p_{\mathrm{idle}} = 0.001$ W (see Table II), as the host robot is idle at that time. Therefore, the difference between the host robot energy consumption for the host robot-neighbor robot and host robot-central cloud total task execution is also very low. Due to the lower energy consumption of the host robot for central cloud computation subtask processing in Fig. 5(b), the host robot-central cloud execution shows 1% higher energy efficiency gain compared to the host robot-neighbor robot scheme. For instance, for a typical total task input size of 240 kB, the host robot energy consumption in the host robot-neighbor robot and host robot-central cloud scheme equals 1.72 and 1.71 J, respectively, whereby the host robot energy consumption for computation subtask processing at the neighbor robot (e^{o}_{ex}) and central cloud ($e^{\mathrm{cl}}_{\mathrm{ex}}$) is equal 0.00192 and 0.0003 J, respectively. Note, however, that the host robot energy consumption for neighbor robot (e^{o}_{ofl}) and central cloud ($e^{\mathrm{cl}}_{\mathrm{ofl}}$) offloading equals 0.00418 and 0.0049 J, respectively.

TABLE III
TASK RESPONSE TIME AND ENERGY EFFICIENCY EVALUATION FOR COLLABORATIVE AND NONCOLLABORATIVE SCHEMES

Scenario No. and Description	**Optimal Scheme Based on Our Proposed Algorithm**	**Task Response Time Efficiency**	**Energy Consumption Efficiency**
Scenario 1: Total task input data size= 240KB, $l_u = 120$KB, $l_r = 48$KB, $s_{cpu} = c_{cpu}$=960M cycles, $\mu_{cl} = \mu_{ct}$=3200MHz, μ_o=500MHz, $m_o = m_{ct}$=35MB, m_r=25MB	Host robot (sensing)-cloudlet (computation) based scheme	36% gain over host robot-neighbor robot based scheme	2% gain over host robot-neighbor robot based scheme
		8% gain over host robot without offloading	15% gain over host robot without offloading
		1% gain over host robot-central cloud scheme	1% gain over host robot-central cloud scheme
Scenario 2: Total task input data size= 1600KB, $l_u = 800$KB, $l_r = 400$KB, $s_{cpu} = c_{cpu}$=200M cycles, μ_{cl}=1800MHz, μ_o=1200MHz, m_o=35MB, m_{ct}=15MB, m_r=25MB	Host robot (sensing)-neighbor robot (computation) based scheme	1% gain over host robot-central cloud scheme	1% gain over host robot-central cloud scheme
		15% lower than host robot without offloading scheme but satisfies task deadline (our criteria)	3% gain over host robot without offloading
Scenario 3: Total task input data size= 240KB, $l_u = 120$KB, $l_r = 48$KB, $s_{cpu} = c_{cpu}$=960M cycles, μ_{cl}=5000MHz, μ_o=500MHz, m_o=25MB, m_{ct}=20MB, m_r=25MB	Host robot (sensing)-cloud (computation) based scheme	38% gain over host robot-neighbor robot based scheme	1% gain over host robot-neighbor robot based scheme
		11% gain over host robot without offloading	15% gain over host robot without offloading
Scenario 4: Total task input data size= 240KB, $l_u = 120$KB, $l_r = 48$KB, $s_{cpu} = c_{cpu}$=960M cycles, $\mu_{cl} = \mu_{ct}$=3200MHz, $\mu_j = (1000, 800, 500, 1600, 1200, 500)$, μ_o=600MHz	Host robot (sensing)-cloudlet (computation) based scheme	30% gain over host robot-neighbor robot based scheme	2% gain over host robot-neighbor robot based scheme
		12% gain over host robot without offloading	18% gain over host robot without offloading
		1% gain over host robot-central cloud scheme	1% gain over host robot-central cloud scheme
		25% gain over minimum distance based robot task execution [7]	28% gain over minimum distance based robot task execution scheme [7]
		31% gain over fixed assignment based robot task execution [8]	33% gain over fixed assignment based robot task execution scheme [8]

Furthermore, we observe from both Fig. 5(a) and (b) that the host robot-cloudlet-based joint task execution scheme outperforms the host robot-central cloud-based joint scheme in terms of task response time and energy consumption of host robot. This is mainly due to the fact that the cloudlet implies a smaller computation offloading delay than the central cloud. The host robot-cloudlet-based scheme shows a 36%, 8%, and 1% increase of task response time efficiency and a 2%, 15%, and 1% higher energy efficiency than the host robot-neighbor robot, host robot without offloading, and host robot-cloud-based scheme, respectively. Thus, the host robot-cloudlet-based joint task execution scheme is optimal for scenario 1.

Interestingly, Fig. 5(c) and (d) indicates that the neighboring robot can also be selected as a collaborative node for computation subtask offloading since the sensing subtask is restricted to the initially selected host robot. In scenario 2, the central cloud CPU power and task workload (required CPU cycles to process the task) are smaller than in scenario 1. Hence, the cloudlet is unsuitable in this task execution scenario due to its insufficient available memory size. We notice that in Fig. 5(c), in comparison with the host robot without offloading scheme, the host robot-neighbor robot scheme experiences a longer response time (15% less gain than host robot without offloading scheme for 1600-kB total task input data size). This is because the host robot CPU power ($\mu_j = 1600$ MHz) is higher than that of the neighbor robot CPU power ($\mu_o = 1200$ MHz), which eventually causes a longer total task response time in the host robot-neighbor robot scheme compared to the host robot without offloading scheme. However, the host robot-neighbor robot scheme achieves a higher energy efficiency gain than host robot without offloading scheme by offloading the computation subtask to neighbor robot. Host robot consumes very little average idle energy consumption ($p_{\text{idle}} = .001$ W per second) during neighbor robot computation subtask execution in the host robot-neighbor robot scheme, which is smaller than the host robot average energy consumption ($p_c = .5$ W per second) for its own computation subtask processing in the host robot without offloading scheme. By contrast, Fig. 5(d) shows that the energy savings of the host robot for the host robot-neighbor robot scheme compared with host robot without offloading is not that significant for the following two reasons. First, the longer computation subtask processing time at the neighbor robot causes an increased idle energy consumption that reduces the host robot's energy savings in the host robot-neighbor robot scheme. Second, due to the smaller computation subtask response time during the host robot's own execution, the energy consumption of the host robot in the host robot without offloading scheme is less. Thus, in Fig. 5(d), the energy efficiency gain achieved by the host robot-neighbor robot scheme compared with host robot without offloading is very low. For instance, in Fig. 5(d), for a typical total task input size of 1600 kB, the host robot energy consumption for its own computation subtask execution in the host robot without offloading scheme equals 0.0625 J, while that of the neighbor robot computation subtask execution in the host robot-neighbor robot scheme is equal to 0.0292 J. Hence, the host robot energy consumption of the total task (sensing and computation) execution equals 1.53 J in the host robot without offloading and 1.49 J in the host robot-neighbor robot scheme, respectively.

Further, from both Fig. 5(c) and (d) we observe that the host robot-neighboring robot-based joint task execution scheme exhibits an improved task response time compared to the host robot-central cloud-based joint scheme (1%) due to its lower computation offloading delay. The host robot-neighboring robot scheme achieves a 1% and 3% higher energy efficiency than the host robot-cloud and host robot without offloading scheme, respectively. Thus, the host robot-neighboring robot-based joint task execution scheme is the most suitable one for this scenario by providing the lowest energy consumption while satisfying the task deadline.

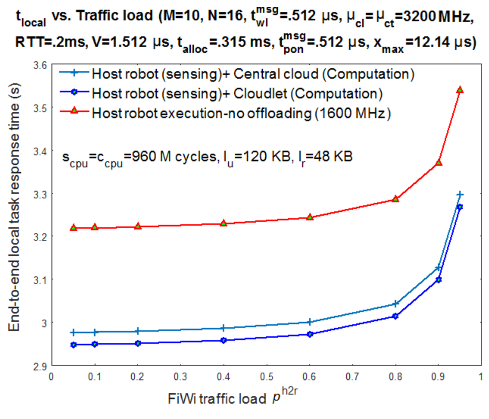
Fig. 5(e) and (f) depicts the task response time and energy consumption of our task execution schemes for a different scenario 3. In this scenario, the total task workload (CPU cycles to process the task) is higher than in the previously considered scenario 2. More specifically, the central cloud is assumed to be more powerful than the collaborative node (i.e., neighboring robot). We observe that the host robot-neighboring robot-based joint task execution is unable to meet the task deadline requirement. In addition, the cloudlet is unable to execute the task due to insufficient available memory. By contrast, the host robot-central cloud-based joint task execution scheme is the best choice for this scenario as it offers a smaller task response time and energy consumption of host robot than its counterparts. The host robot-cloud-based scheme shows a 38% and 11% higher task response time efficiency than the host robot-neighbor robot and host robot without offloading scheme and a 1% and 15% higher energy efficiency than the host robot-neighbor robot and host robot without offloading scheme, respectively.

In Fig. 6(a) and (b), we compare the performance of our collaborative and noncollaborative schemes with previously proposed minimum distance [7] and fixed assignment [8] based robot task execution schemes in a setting referred to as scenario 4 in Table III. Note that these task offloading schemes examined only computation task for execution, while location dependent sensing subtask was considered out of their scope. In this paper, the considered H2R task consists of both sensing and computation subtasks, whereby location-dependent sensing subtasks are restricted to robots and location-independent computation subtasks can either be done by robots or offloaded to a collaborative node (central cloud/cloudlet/neighbor robot) for execution. For fair comparison, we consider only existing robot-based full task (sensing and computation subtask) execution schemes. In scenario 4, both central cloud and cloudlet are assumed to have the same CPU power. However, the neighbor robot CPU power is assumed to be smaller than that of the central cloud and cloudlet. Other parameters settings are shown in Fig. 6(a) and (b). The distance [7] and fixed assignment [8] based task execution schemes were discussed in greater detail above in Section IV-A.

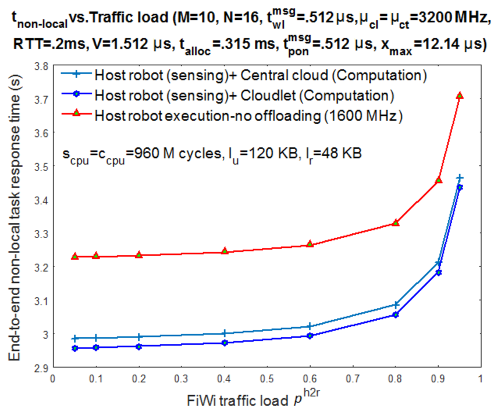
Fig. 6(a) and (b) clearly shows that for scenario 4 the host robot-cloudlet-based joint task execution achieves a significantly improved average task response time and energy consumption efficiency than the other schemes. For instance, for a typical task input size of 240 kB, the host robot-cloudlet-based joint execution shows a 30%, 12%, 1%, 25%, and 31% improved task response time with regard to the host robot-neighbor robot, host robot without offloading, host robot-cloud, minimum distance [7], and fixed assignment [8] based scheme, respectively. Moreover, for the assumed task input size of 240 kB, the host robot-cloudlet-based joint scheme achieves a 2%, 18%, 1%, 28%, and 33% higher energy efficiency than the host robot-neighbor robot, host robot without offloading, host robot-cloud, minimum distance [7], and fixed assignment [8] based scheme, respectively.

Next, we investigate the total task response time and energy efficiency of collaborative schemes that satisfies the task deadline for scenario 1. The task response time efficiency of our collaborative full task execution scheme is defined as the ratio of the collaborative task response time gain ($t_j - t_{j,\text{cn}}$) and the task response time of noncollaborative (t_j) execution. The energy consumption efficiency of our collaborative full task execution scheme is defined as the ratio of the energy consumption gain of the host robot for collaborative execution ($e_j - e_{j,\text{cn}}$) and the energy consumption of the host robot for noncollaborative (e_j) scheme. The host robot-central cloud and host robot-cloudlet-based joint task execution gain over noncollaborative host robot task execution scheme are depicted in Fig. 7(a) and (b), respectively. Both figures clearly indicate that for scenario 1, the host robot-cloudlet-based joint task execution achieves a superior task response time and energy efficiency than the host robot-central cloud-based joint scheme. For instance, for a typical total task input size of 240 kB, the host robot-cloudlet-based joint task execution shows an 8.75% improvement of task response time and a 14.98% improvement of energy efficiency than the host robot-based noncollaborative task execution scheme. Hence, the host robot-central cloud-based joint execution achieves a 7.81% decrease of task response time and a 14.72% increase of energy efficiency in comparison with the noncollaborative scheme. Further, in Fig. 7(c), the computation subtask offload gain-overhead ratio of both central cloud and cloudlet execution are shown. The computation subtask offload gain-overhead is defined as the ratio of the offload gain for collaborative node-based computation subtask execution ($t_c^j - t_c^{\text{cn}}$) and the offload overhead ($\bar{d}_{\text{cn}} + t_{\text{ofl}}^{\text{cn}}$) incurred by the communication protocols [see (14)].

Importantly, we observe from Fig. 7(c) that under the assumption that both central cloud and cloudlet have same computation power, the cloudlet-based computation subtask execution achieves a higher offload gain than the central cloud. For instance, for a typical computation subtask input data size of 120 kB, the offload gain overhead ratio of central cloud (3200 MHz) and cloudlet (3200 MHz) is 34% and 36%, respectively. However, if the computation power of the central



(a)



(b)

Fig. 8. End-to-end local and nonlocal task response time variation versus FiWi traffic load.

cloud (5000 MHz) is assumed to be higher than that of the cloudlet (3000 MHz), the central cloud shows a much better offload gain (48%) than the cloudlet execution (32%).

Finally, we evaluate the end-to-end local (t_{local}) and nonlocal (t_{nonlocal}) task response time under different FiWi traffic loads, as shown in Fig. 8(a) and (b). Local (MU and robot for task execution are located under the same ONU-MPP) task response calculation accounts for the US frame transmission delay of an MU's task request transmission (see Section III-D), task allocation delay for robot selection, time to reach the task location, sensing, and computation subtask execution. Conversely, nonlocal (MU and robot for task execution are located under different ONU-MPPs) task response time calculation takes into account both US and DS frame transmission delay of an MU's task request transmission (see Section III-D), robot selection delay for task allocation, required time for selected robots to reach the task location, sensing, and computation subtask execution. Recall from Section III-E that for the collaborative task execution scheme, the nonlocal and local task response time is equal to $t_{\text{nonlocal}} = t_u + t_d + t_{j,\text{cn}}$ and $t_{\text{local}} = t_u + t_{j,\text{cn}}$, respectively. by contrast, for the noncollaborative host robot task execution scheme, the nonlocal and local task response time is equal to $t_{\text{nonlocal}} = t_u + t_d + t_j$ and $t_{\text{local}} = t_u + t_j$, respectively.

Note that, both local and nonlocal task response times increase for increasing traffic loads in our considered FiWi network scenario in Fig. 8(a) and (b). Both figures clearly indicate that the host robot-cloudlet-based scheme provides an improved task response time than host robot-central cloud (2%) and host robot without offloading (10%) schemes. This is because the host robot-central cloud-based scheme incurs a higher computation offloading delay than the host robot-cloudlet-based execution. Moreover, the host robot-based noncollaborative task execution experiences a much higher task response time than the alternate collaborative schemes. This result is expected given that the host robot is less powerful than the central cloud and cloudlet. We also note that the end-to-end local task response of all compared schemes (collaborative and noncollaborative) are lower than their nonlocal task response time. The reason behind this is that beside task execution delay, the calculation of t_{nonlocal} involves both US and DS frame transmission delays for end-to-end task allocation, while t_{local} involves only the one-way US frame transmission delay.

V. Conclusion

Efficient task allocation among robots, computation offloading onto collaborative nodes, and adaptive resource allocation schemes represent key design challenges for reducing the end-to-end latency in advanced Tactile Internet H2R communications. In this paper, we presented a collaborative computing enhanced H2R task allocation mechanism that combines suitable host robot and collaborative node selection in integrated FiWi multirobot networks.

To improve the energy efficiency of the selected host robot while satisfying a given task deadline, we investigated both host robot-based noncollaborative and joint task execution schemes, in which the sensing subtask is conducted by a suitable host robot and the computation subtask is offloaded onto one of the collaborative nodes consisting of central cloud, cloudlets, and neighboring robots. In order to handle both conventional broadband and computation offloading traffic at the same time, we introduced a unified TDMA-based resource management scheme. Moreover, we developed an analytical framework to evaluate the performance of our proposed noncollaborative and collaborative task execution schemes in terms of task response time efficiency and energy efficiency of host robots. Unlike previous studies, we also analyzed the end-to-end local/nonlocal task response time for both collaborative and noncollaborative task execution schemes.

Our results provide insight into finding the optimal task execution scheme for a variety of use case scenarios with different task, robot, and collaborative node availability characteristics. The results of both collaborative/joint and noncollaborative task execution schemes demonstrate that for a typical task input size of 240 kB, the collaborative task execution scheme decreases the task response time by up to 8.75% and the energy consumption by up to 14.98% compared to the noncollaborative task execution scheme. The introduced collaborative computing-based H2R task allocation and resource management scheme represents a promising solution for enabling low-latency Tactile Internet applications.

References

[1] H. Beyranvand *et al.*, "Toward 5G: FiWi enhanced LTE-A HetNets with reliable low-latency fiber backhaul sharing and WiFi offloading," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 690–707, Apr. 2016.

[2] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-enabled tactile Internet," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 460–473, Mar. 2016.

[3] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The tactile Internet: Vision, recent progress, and open challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 138–145, May 2016.

[4] J. M. Bradshaw, V. Dignum, C. Jonker, and M. Sierhuis, "Human-agent-robot teamwork," *IEEE Intell. Syst.*, vol. 27, no. 2, pp. 8–13, Mar./Apr. 2012.

[5] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks*, vol. 604. Switzerland: Springer, 2015, pp. 31–51.

[6] A. Viguria, I. Maza, and A. Ollero, "S+T: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation," in *Proc. IEEE ICRA*, Pasadena, CA, USA, May 2008, pp. 3163–3168.

[7] S. Giordani, M. Lujak, and F. Martinelli, "A distributed algorithm for the multi-robot task allocation problem," in *Trends in Applied Intelligent Systems* (LNCS 6096). Heidelberg, Germany: Springer, Jun. 2010, pp. 721–730.

[8] R. M. de Mendonça, N. Nedjah, and L. de Macedo Mourelle, "Efficient distributed algorithm of dynamic task assignment for swarm robotics," in *Computational Science and Its Applications* (LNCS 7971). Heidelberg, Germany: Springer, Jun. 2013, pp. 500–510.

[9] P. Patil, A. Hakiri, and A. Gokhale, "Cyber foraging and offloading framework for Internet of Things," in *Proc. IEEE COMPSAC*, Atlanta, GA, USA, Jun. 2016, pp. 359–368.

[10] T. Penner, A. Johnson, B. Van Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *Proc. IEEE GLOBECOM*, Austin, TX, USA, Dec. 2014, pp. 2801–2806.

[11] T. Langford, Q. Gu, A. Rivera-Longoria, and M. Guirguis, "Collaborative computing on-demand: Harnessing mobile devices in executing on-the-fly jobs," in *Proc. IEEE MASS*, Hangzhou, China, Oct. 2013, pp. 342–350.

[12] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 190–194.

[13] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Comput.*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[14] I. Osunmakinde and V. Ramharuk, "Development of a survivable cloud multi-robot framework for heterogeneous environments," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 10, pp. 1–22, Oct. 2014.

[15] C. M. S. Magurawalage, K. Yang, L. Hu, and J. Zhang, "Energy-efficient and network-aware offloading algorithm for mobile cloud computing," *Comput. Netw.*, vol. 74, pp. 22–23, Dec. 2014.

[16] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[17] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *IEEE Netw.*, vol. 27, no. 5, pp. 34–40, Sep./Oct. 2013.

[18] H. Zhang, Q. Zhang, and X. Du, "Toward vehicle-assisted cloud computing for smartphones," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5610–5618, Dec. 2015.

[19] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. Int. Conf. Mobile Syst. Appl. Services*, San Francisco, CA, USA, Jun. 2010, pp. 49–62.

[20] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-edge computing vs. centralized cloud computing in fiber-wireless access networks," in *Proc. IEEE INFOCOM Workshop 5G Beyond Enabling Technol. Appl.*, San Francisco, CA, USA, Apr. 2016, pp. 991–996.

[21] D. P. Van, B. P. Rimal, S. Andreev, T. Tirronen, and M. Maier, "Machine-to-machine communications over FiWi enhanced LTE networks: A power-saving framework and end-to-end performance," *IEEE/OSA J. Lightw. Technol.*, vol. 34, no. 4, pp. 1062–1071, Feb. 15, 2016.

[22] B. Bellalta, J. Barcelo, D. Staehle, A. Vinel, and M. Oliver, "On the performance of packet aggregation in IEEE 802.11ac MU-MIMO WLANs," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1588–1591, Oct. 2012.

[23] F. Wang, G. Han, J. Jiang, and H. Qiu, "A distributed task allocation strategy for collaborative applications in cluster-based wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 10, pp. 1–16, Jun. 2014.

[24] S. C. Jha, U. Phuyal, M. M. Rashid, and V. K. Bhargava, "Design of OMC-MAC: An opportunistic multi-channel MAC with QoS provisioning for distributed cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3414–3425, Oct. 2011.

Mahfuzulhoq Chowdhury received the B.Sc. and M.Sc. degrees in computer science and engineering from the Chittagong University of Engineering and Technology, Chittagong, Bangladesh, in 2010 and 2015, respectively. He is currently pursuing the Ph.D. degree in Tactile Internet at the Institut National de la Recherche Scientifique, Montreal, QC, Canada.

His current research interests include collaborative computing, fiber-wireless multirobot networks, human-to-robot communications, and Tactile Internet.

Martin Maier (M'04–SM'09) received the M.Sc. and Ph.D. degrees (both with distinction) from the Technical University of Berlin, Berlin, Germany, in 1998 and 2003, respectively.

He is a Full Professor with the Institut National de la Recherche Scientifique, Montreal, QC, Canada, where he is the Founder and the Creative Director of the Optical Zeitgeist Laboratory. In 2003, he was a Post-Doctoral Fellow with the Massachusetts Institute of Technology, Cambridge, MA, USA. He was a Visiting Professor with Stanford University, Stanford, CA, USA, from 2006 to 2007. He authored *Optical Switching Networks* (Cambridge Univ. Press, 2008), which was translated into Japanese in 2009, and *FiWi Access Networks* (Cambridge Univ. Press, 2012).

Dr. Maier was a co-recipient of the 2009 IEEE Communications Society Best Tutorial Paper Award and the Best Paper Award presented at the International Society of Optical Engineers Photonics East 2000—Terabit Optical Networking Conference. He was a Marie Curie IIF Fellow of the European Commission from 2014 to 2015.