IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 24, 2018, accepted October 14, 2018, date of publication October 17, 2018,
date of current version November 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2876600

An Energy-Efficient Resource Management System for a Mobile *Ad Hoc* Cloud

SAYED CHHATTAN SHAH, (Senior Member, IEEE)
Department of Information Communication Engineering, Hankuk University of Foreign Studies, Seoul 02450, South Korea

e-mail: shah@hufs.ac.kr

This work was supported in part by Hankuk University of Foreign Studies Research Fund of 2018 and in part by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science, ICT, and Future Planning under Grant 2017R1C1B5017629.

ABSTRACT Recently, mobile *ad hoc* clouds have emerged as a promising technology for mobile cyber physical system applications, such as mobile intelligent video surveillance and smart homes. Resource management plays a key role in maximizing resource utilization and application performance in mobile *ad hoc* clouds. Unlike resource management in traditional distributed computing systems, such as clouds, resource management in a mobile *ad hoc* cloud poses numerous challenges owing to the node mobility, limited battery power, high latency, and dynamic network environment. The real-time requirements associated with mobile cyber physical system applications make the problem even more challenging. Currently existing resource management systems for mobile *ad hoc* clouds are not designed to support mobile cyber physical system applications and energy-efficient communication between application tasks. In this paper, we propose a new energy-efficient resource management system for mobile *ad hoc* clouds. The proposed system consists of two layers: a network layer and a middleware layer. The network layer provides *ad hoc* network and communication services to the middleware layer and shares the collected information in order to allow efficient and robust resource management decisions. It uses: (1) a transmission power control mechanism to improve energy efficiency and network capacity; (2) link lifetimes to reduce communication and energy consumption costs; and (3) link quality to estimate data transfer times. The middleware layer is responsible for discovery, monitoring, migration, and the allocation of resources. It receives application tasks from users and allocates tasks to nodes on the basis of network- and node-level information.

INDEX TERMS Cloud robotics, local data center, mobile cloud, resource scheduling, sensor cloud.

I. INTRODUCTION

Recently, mobile ad hoc clouds have emerged as a promising technology for mobile cyber physical system (CPS) applications, such as mobile intelligent video surveillance and smart homes. In mobile CPS applications, various kinds of sensing and computing devices are deployed in order to monitor and analyze numerous situations and take the necessary actions in real time [1]. This involves sophisticated image and video processing algorithms, which require a vast amount of computing and storage resources. In order to address this issue, the conventional approach is to send the collected data to an application on an Internet cloud accessible through an infrastructure-based system, such as a cellular network [1]. This approach has several issues, such as high transmission energy consumption and communication latency. In addition, it cannot be used if preexisting network infrastructure is not available. In order to overcome the drawbacks of the conventional cloud-based approach, a new technology called mobile ad hoc clouds is currently being developed, in which multiple mobile devices, interconnected through a mobile ad hoc network, are combined to create a virtual supercomputing node or a small local cloud data center [1]. Mobile ad hoc clouds not only are deployable in network environments with no infrastructure but also significantly reduce the transmission energy consumption and communication latency.

A block diagram and the architecture of a mobile ad hoc cloud computing system are shown in Figs. 1 and 2. Nodes in mobile ad hoc clouds are divided into three categories: service master nodes (SMNs), service provider nodes (SPNs), and service consumer nodes (SCNs). SMNs are responsible for allocating tasks submitted by the SCNs to the SPNs on the basis of the resource allocation policy. Nodes communicate with each other through a mobile ad hoc network, which provides several communication services, including routing

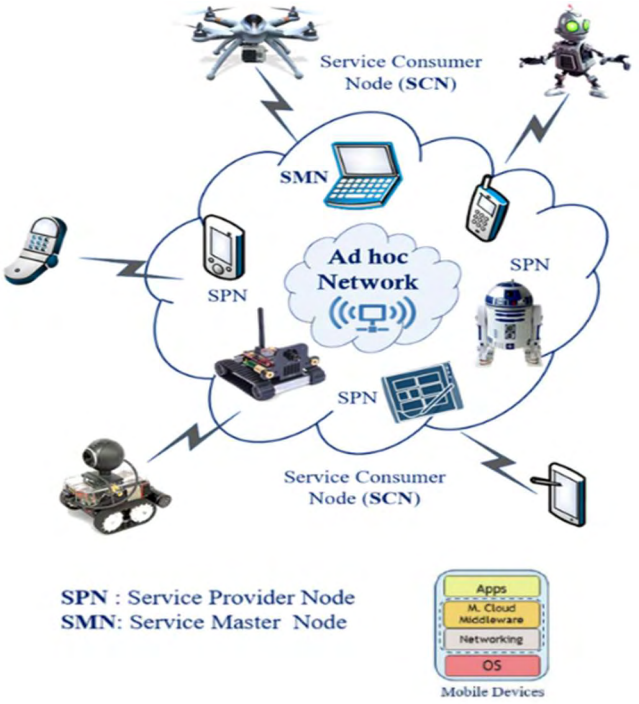


FIGURE 1. Block diagram of a mobile ad hoc cloud computing system.

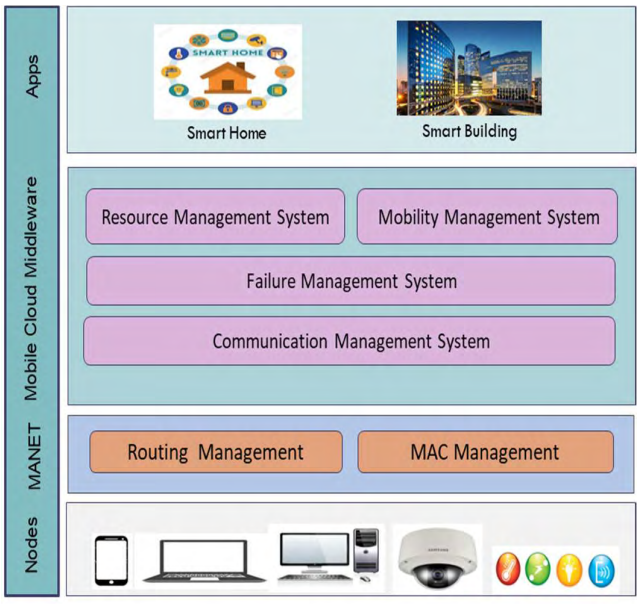


FIGURE 2. Architecture of a mobile ad hoc cloud computing system.

and medium access. The mobile cloud middleware layer is responsible for resource management, failure management, mobility management, communication management, and task management. In addition, the middleware hides all the complexities and provides a single system image to the user and applications running on the system.

A resource management system (RMS) is an integral part of any distributed computing system and is responsible for the discovery, monitoring, and allocation of network and system resources. Compared to traditional parallel and distributed computing systems, such as clouds, resource management in a mobile ad hoc cloud poses numerous challenges owing to the node mobility, limited battery power, high latency, and the dynamic network environment [1]. The real-time requirements associated with mobile CPS applications make the problem even more challenging. Research into resource management for mobile ad hoc clouds is still in a preliminary phase, and very few schemes based on a decentralized architecture have been proposed to address issues such as node mobility, energy management, and task failure [1].

In this paper, we propose an energy-efficient RMS for a mobile ad hoc cloud. The proposed system consists of two layers: a network layer and a middleware layer. The network layer provides ad hoc network and communication services to the middleware layer and shares the collected information in order to allow efficient and robust resource management decisions. It uses (1) a transmission power control mechanism to improve energy efficiency and network capacity, (2) link lifetimes to reduce communication and energy consumption costs, and (3) link quality to estimate data transfer times. The middleware layer is responsible for the discovery, monitoring, migration, and allocation of resources. It receives application tasks from users and allocates tasks to nodes on the basis of network- and node-level information.

Compared to already existing RMSs, the proposed system focuses on mobile CPS applications and energy-efficient communication between tasks. In addition, it aims to address task failure by adopting a failure avoidance mechanism. In order to make effective resource allocation decisions, a cross-layer design approach is used. Our research on transmission power control mechanisms is used to reduce transmission power energy consumption and increase concurrent transmissions in the network. The new system considers link quality and lifetime to reduce data transfer costs and communication energy consumption.

The key contributions of this paper are as follows:

- An energy-efficient RMS for mobile ad hoc clouds
- A network layer to provide energy-efficient and reliable ad hoc network and communication services
- A link lifetime aware routing protocol based on a transmission power control mechanism to reduce the transmission energy consumption and data transfer costs
- An efficient distance-based discovery mechanism to reduce the communication costs
- A Markov-chain-based LLPH model to predict link lifetimes
- A data transfer time estimation model that considers packet overhead, the number of dropped and lost packets, and traffic at neighboring nodes
- An energy consumption estimation model
- A task completion time estimation model

The rest of the paper is organized as follows. Section II focuses on the related work, Section III describes energy-efficient mobile ad hoc cloud RMS, Section IV describes the preliminary results, and Section V presents the conclusion.

II. LITERATURE REVIEW

Research into resource management for mobile ad hoc clouds is still in a preliminary phase, and very few schemes based on a decentralized architecture have been proposed in order to address issues such as node mobility, energy management, and task failure. This section is divided into two parts. The first part describes the work related to resource allocation, and the second part focuses on resource discovery and monitoring schemes.

A. RESOURCE ALLOCATION SCHEMES

A resource allocation scheme based on a distributed architecture was proposed in [2]. The scheme uses proactive and reactive failure management approaches and supports redundant execution of tasks to handle a failure. The scheme that was developed in [3] uses a program-controlled migration technique to address task failures. In order to allocate tasks, it uses a manager–worker model. Energy minimization and grid utility optimization problems are addressed in [4], and a scheme to reduce energy consumption was described in [5]. In order to balance the computational workload and energy consumption, the authors of [6] proposed a scheme that investigates the cooperation among mobile nodes. The scheme proposed in [7] addresses load balancing and scalability problems. In order to allocate a task, it uses a delayed response mechanism in which powerful nodes reply earlier than less powerful ones. A node mobility problem is addressed in [8] by profiling the mobility patterns of users. Li *et al.* [9] developed several online and batch scheduling schemes to offload CPU-intensive tasks onto a mobile ad hoc cloud. The MinHop scheme assigns tasks to a node on the basis of the number of hops, whereas the minimum execution time (MET) with communication scheme assigns a task to a node that would take the minimum time to execute. The minimum completion time (MCT) with communication scheme selects a node with the minimum expected completion time. MinMinComm and MaxMinComm, which are based on traditional MET, MCT, MinMin, and MaxMin schemes, have been proposed for batch scheduling. Compared to traditional schemes, in the proposed scheme, communication costs are considered. Task allocation problem in numerous network environments have been studied in [10]. A separate scheme has been proposed for each environment. A greedy scheme has been developed for an ideal network environment, which assigns a task to the node with the minimum task completion time.

Numerous job-stealing schemes (e.g., random, best rank aware, and worst rank aware) were proposed in [11]. These schemes are based on a centralized architecture and are aimed at reducing the energy consumption. The problems of a dynamic and unpredictable network environment, energy consumption, and node failure are addressed in [12]. The scheme proposed in [13] addresses the uncertainty problem using application waypoints. A node executing a task sends an estimate of the residual task completion time to a broker node. If the broker node fails to receive a task progress update at a specified waypoint, it assumes that the node has failed and takes necessary measures to complete the task.

In order to address the transmission energy consumption problem, an energy-efficient resource allocation scheme was proposed in [14]. This scheme uses a transmission power control mechanism to reduce transmission energy and a hybrid architecture for making efficient decisions. The node mobility problem was addressed in [15]. The scheme proposed in [15] uses the history of user mobility patterns, task characteristics, and distance information to reduce data transfer times. An efficient and robust resource allocation scheme that was developed in [16] aims to reduce task completion times and transmission energy consumption using a transmission power control mechanism and user mobility patterns.

Existing resource allocation schemes for mobile ad hoc systems do not consider the network environment, task queue size, or CPU overhead. Very few schemes consider the network environment, but they do not consider link quality, link lifetime, or the migration or reallocation of tasks.

This work is different from the schemes proposed in the literature, because it focuses on mobile CPS applications and energy-efficient communication between tasks. In addition, it aims to address task failure by adopting a failure avoidance mechanism. In order to make effective resource allocation decisions, a cross-layer design is used. Our previous research into transmission power control mechanisms is used to reduce the transmission energy consumption and increase concurrent transmissions in the network. The new system makes allocation decisions on the basis of CPU speed, task queue size, CPU overhead, link quality, and link lifetime.

B. RESOURCE DISCOVERY AND MONITORING

Numerous resource discovery protocols have been developed for wired and wireless networks and systems. In [17], [28], and [30] a survey of discovery technologies and a categorization based on search strategies and protocols are provided. The first category includes technologies such as Bluetooth Low Energy, which enables the discovery of resources in close spatial proximity. The client node sends a discovery message or may wait for an advertisement message. The second category includes technologies and protocols that enable the discovery of a resource on the network [19]–[21]. In order to discover a resource, a client multicasts a discovery request message over the network. A host receives the discovery message and replies via a response message. The third category uses a central or distributed directory [22]–[24], which stores information about resources. The information is registered by resource providers. A discovery request message is sent to the directory, which then replies via a discovery response message. In [18], a resource discovery and selection process based on matching parameters was proposed. The authors used an analysis matrix to describe multiple criteria in a decision analysis problem. The behavior of simple additive weighting and the TOPSIS and VIKOR multiobjective decision methods is analyzed. Normalization, the Euclidean distance, and sorting algorithms are used to optimize the

TABLE 1. Summary of resource management schemes.

	Transmission Energy Aware	Processing Energy Aware	Mobile CPS App Support	Cross Layer Design	Network Aware	Mobility Management	Load Balancing	Failure Management	Link Lifetime Aware	Link Quality Aware
Existing Schemes	[14] [16]	[4] [5] [6] [11] [13]	None	None	[9] [14] [15][16]	[8] [15] [16]	[6] [7] [11]	[2] [3] [12] [13] [15] [16]	None	None
Proposed Scheme	√		√	√	√	√		√	√	√

resource selection process. Li *et al.* [25] proposed a resource discovery mechanism in order to enhance the search efficiency over the social Internet of Things. The mechanism is based on preference and movement pattern similarity. A three-layer network structure is adopted, in which each node has some common interest. Based on these interests, subcommunities and global communities are formed. Every node has a preference vector, which is used to calculate the similarity between two nodes. Different parameters, such as weighting parameters, user-defined parameters, location coordinates, temporal features, and overlapping movement regions, are utilized. In order to address the challenges of the distributed nature, energy efficiency, scalability, and fast discovery, Huang *et al.* [26] proposed a device-to-device discovery protocol. This protocol allows neighboring nodes to form a group and to be synchronized. The nodes use table and time interval information in the group for discovery and beacon broadcasting. A device discovery approach and a connection establishment scheme for opportunistic networks were developed in [27], where a device announces its existence using a beacon stuffing method. The authors also proposed a score-based scanning schedule for device discovery to reduce the energy consumption.

The resource discovery protocols that were developed for wired and infrastructure-based wireless networks cannot be used in mobile ad hoc systems, which are characterized by a dynamic and unreliable network environment. Wired networks have stationary devices, whereas infrastructure-based wireless networks have nodes with restricted mobility.

The resource discovery protocols that were developed for mobile ad hoc systems work at either the application layer or the network layer. The application layer protocols focus on the architecture and the distribution of discovery components and are not aware of the communication mechanisms employed at the network layer [28]. They also do not consider node mobility and the dynamic network environment. Network layer discovery protocols are developed as part of a routing protocol and are used to discover limited network-level information, such as the node ID and number of hops [29].

III. AN ENERGY EFFICIENT RESOURCE MANAGEMENT SYSTEM FOR A MOBILE AD HOC CLOUD

The architecture of an energy-efficient mobile cloud RMS is shown in Fig. 3. The RMS is divided into two layers: a network layer and a middleware layer. The network layer provides network and communication services and also collects network-level information, such as link quality and lifetime, which is used by the middleware layer during the resource allocation process. The key components of each layer are described below.

A. RMS NETWORK LAYER

The network layer provides ad hoc network [43] and communication services to the middleware layer, and it shares the collected information in order to enable efficient and robust resource management decisions.

The network layer also plays a key role in the overall network and system performance. The key factors that affect the performance of the network layer are the transmission power, link lifetime, and link quality. The transmission power affects the transmission energy consumption and network capacity. The link quality determines the data transfer time, whereas the link lifetime determines the communication and energy consumption costs. Numerous protocols have been proposed in the literature, but they consider either the transmission power [36]–[41], link lifetime [42]–[47], or link quality [5], [35]. Transmission power aware routing protocols have poor discovery mechanisms. There are two categories of protocols based on the link lifetime. The first category includes protocols that rely on the current information about nodes, such as the location, distance, speed, residual energy, and transmission energy consumption. The schemes in the second category exploit historical information about nodes and users, such as mobility patterns and social relationships. Link quality aware routing protocols do not consider the packet overhead, the number of dropped and lost packets, or the traffic at neighboring nodes.

In this paper, an energy-efficient and link lifetime aware network layer is proposed in order to improve the energy efficiency and data transfer times. The proposed layer uses (1) a

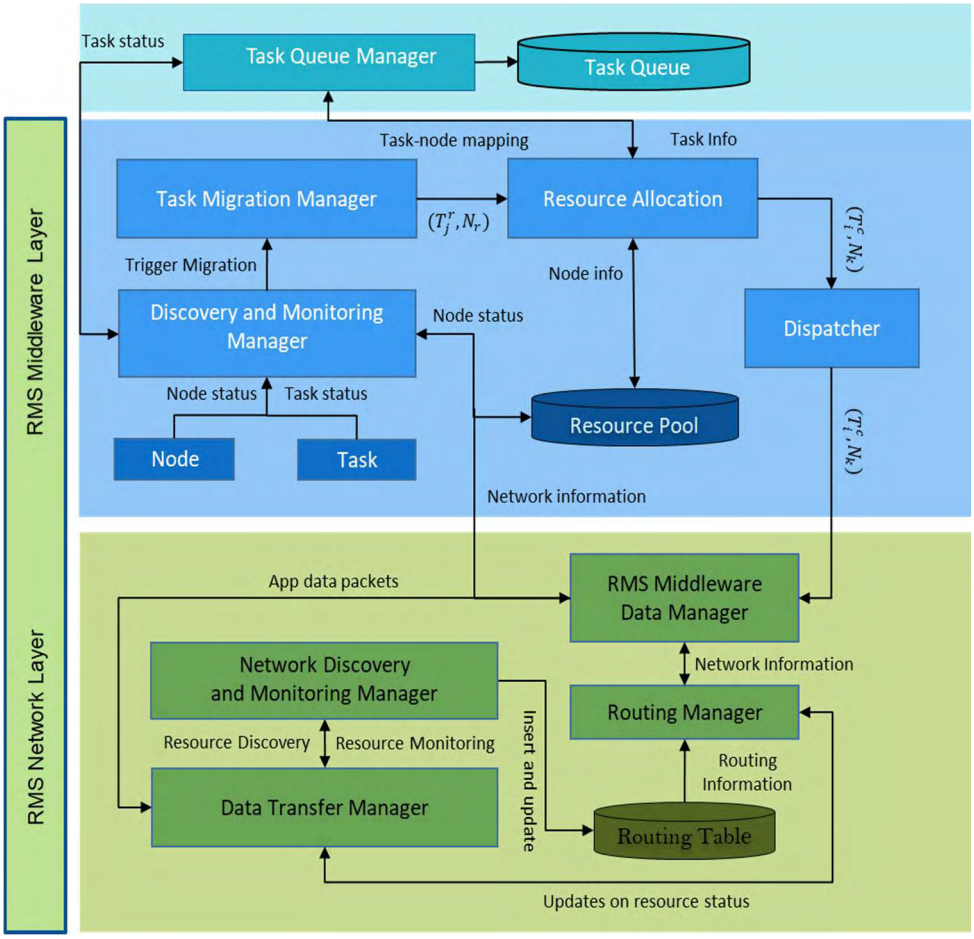


FIGURE 3. RMS architecture.

transmission power control mechanism to improve the energy efficiency and network capacity, (2) link lifetimes to reduce the communication and energy consumption costs, and (3) link quality to estimate data transfer times. The proposed process at the network layer is as follows:

- Receive a data transmission request from the middleware.
- Estimate the data transfer time.
- Predict the link lifetime.
- Select an energy-efficient route with a link lifetime greater than or equal to the estimated data transfer time.

The architectural elements of the network layer are described below.

1) MIDDLEWARE DATA MANAGER

The middleware data manager hides the network's complexity and provides an interface to the middleware for data transmission across the network.

2) NETWORK DISCOVERY AND MONITORING MANAGER

The network discovery and monitoring manager discovers and monitors the following network-level information:

1) Node ID
2) Average number of dropped and lost packets
3) Available bandwidth
4) Neighboring traffic
5) Self-traffic
6) Total bandwidth
7) Energy consumption per packet
8) Received signal strength intensity

The information collected by the discovery and monitoring manager is used to estimate the data transfer times and predict link lifetimes. The following sections describe the discovery and communication mechanism and the process of estimating the data transfer times and link lifetimes.

a: AN ENERGY-EFFICIENT DISCOVERY AND COMMUNICATION

In order to reduce the transmission energy consumption, a ClusterPow discovery and routing protocol was developed in [31]. This routing protocol significantly reduces the energy consumption but introduces a communication overhead owing to its poor discovery mechanism [16]. In this section, an efficient discovery mechanism is proposed to reduce this communication overhead. It is assumed that each node can transmit at multiple transmission power levels.

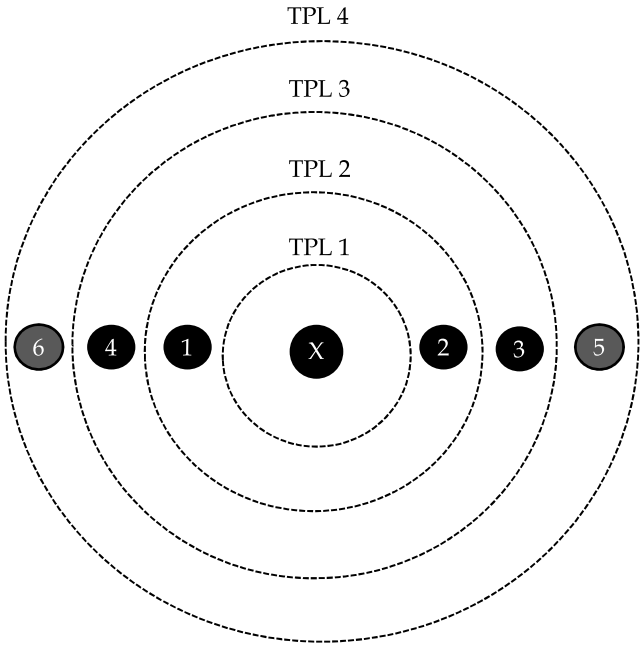


FIGURE 4. Illustration of the transmission power control mechanism.

For each transmission power level TPL_i (Fig. 4), each node maintains a separate routing table RTP_i. In order to discover nodes at each TPL, a source node uses the maximum transmission power to broadcast a discovery request packet.

The node that receives the discovery request packet calculates the distance to the source node using the following equation:

$$D_{ij} = 10^{\wedge} \frac{\left(TX_{power} - RSSI\right)}{10 \times n}, \tag{1}$$

where D_{ij} is the distance between nodes N_i and N_j, TX_{power} is the transmission power of TPL_i, RSSI is the received signal strength intensity, and n is a constant that depends on environmental factors.

Based on the distance, the node determines the transmission power level required to communicate with the source node. The node that received the discovery request packet then sends a reply packet at the selected transmission power level, which is also included in the packet.

When the source node receives the reply packet, it adds a new entry or updates an existing entry in the routing table. A routing entry comprises the node address and the transmission power level included in the reply packet.

In order to forward a packet to a destination, the routing manager accesses the lowest power routing table RTP_{min} in which the destination is present and forwards the packet at power level TPL_{min} to the next hop indicated in the routing table RTP_{min}.

To reduce communication costs, the discovery process should be started at a master node rather than at every node in the network.

b: ESTIMATING THE DATA TRANSFER TIME

The link quality reflects the available bandwidth and is used to estimate the data transfer times. In order to estimate the link quality, Model 2 that was developed in [5] and [35] is used. Compared to existing work, the model in [5] and [35] considers self-traffic and traffic at neighboring nodes:

$$\begin{aligned} LQ_{\mathrm{mk}} &= B_{available}(N_{\mathrm{mk}}) \\ &= B_{channel} - \sum\nolimits_{k \in N_{neighbors}(N_m)} B_{self}\ (N_k) \end{aligned} \tag{2}$$

where LQ_{mk} is the link quality between nodes N_m and N_k, $B_{self}(N_k)$ is the total traffic between node N_k and its neighbors, $B_{channel}$ is the total bandwidth, and $B_{available}(N_{mk})$ is the available bandwidth between nodes N_m and N_k.

The application or middleware layer sends a data transmission request, which includes the data size, to the network layer. Based on the data size, the network layer estimates the data transfer time using Model 3,

$$\begin{aligned} E_{DTT} &= (P_{kts} \times Pkt_{size}) \div LQ_{mk} \\ &\quad + (P_{kts\_AvgDL} \times Pkt_{size}) \div LQ_{mk} \end{aligned} \tag{3}$$

$$\begin{aligned} &Number\ of\ packets\ P_{kts} \\ &\quad = Data_{size} \div (Pkt_{size} - Pkt_{header\_size}) \end{aligned} \tag{4}$$

where P_{kts_AvgDL} is the average number of dropped and lost packets, $Data_{size}$ is the quantity of data to be transmitted, Pkt_{size} is the packet size, and Pkt_{header_size} is the packet header size.

Compared to existing work, the proposed model considers the packet overhead, the average number of dropped and lost packets, and the traffic at neighboring nodes. After estimating the data transfer time, the routing manager selects a route with an estimated lifetime greater than or equal to the estimated data transfer time. If multiple routes are available, then the route with the maximum probability is selected. For real-time data transmission, multiple routes can be selected to transfer data simultaneously to a destination node.

c: ESTIMATING THE LINK LIFETIME

In a previous study, we proposed a Markov-chain-based node location prediction scheme [16], which exploits the history of a user's mobility patterns to predict the next location. In this paper, a Markov-chain-based link lifetime estimation scheme is proposed. The main idea is to predict the available link lifetime on the basis of the history of link lifetime intervals. A route with an estimated lifetime greater than or equal to the application data transfer time is then selected for data transmission.

In order to predict the link lifetime, a Markov chain model is used. A Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property that, given the present state of the system, the future is independent of its past.

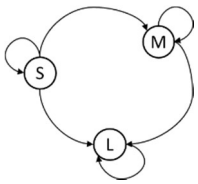
Formally,

$$\begin{aligned} &\Pr(X_{n+1} = x \,|X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) \\ &\qquad = \Pr(X_{n+1} = x \,|X_n = x_n) \end{aligned}$$

The random variable X_i takes values from a countable discrete set S. The elements of S are called states, and S is referred to as the state space.

The underlying Markov model represents each state as a link lifetime interval, and transitions represent the possible link lifetime intervals that follow the current interval. Whenever a link becomes available, the Markov transition probability matrix, which is used to describe the transitions, is updated. A row in the transition matrix includes the probabilities of link lifetime intervals. In order to make a prediction, the algorithm scans the row of the transition matrix corresponding to the current state or link lifetime interval and selects the entry with the maximum probability. The maximum probability interval is the interval that has most frequently followed the current interval in the past.

As a demonstration, Fig. 5 shows three link lifetime intervals: short, medium, and long. A transition probability matrix is also shown in Fig. 5. For each interval, the transition probability matrix contains the probabilities of moving from that interval to another. For example, a link L_{mn} with current state S or a short lifetime interval has a 0.5 probability that it will be available for S or a short duration, a 0.4 probability that it will be available for M or a medium duration, and a 0.1 probability that it will be available for L or a long duration.



	S	M	L
S	.5	.4	.1
M			.7
L			

FIGURE 5. State transition diagram and transition probabilities.

A transition probability matrix is maintained for each link. The prediction is made once the link becomes available. The possible transitions are $S \rightarrow S$, $S \rightarrow M$, $M \rightarrow M$, and $M \rightarrow L$. The following transitions are not logical and, therefore, are not permitted: $M \rightarrow S$, $L \rightarrow M$, and $L \rightarrow S$.

The main objective of predicting the link lifetime and considering it during link, route, and node selection is to reduce the energy consumption, data transfer time, and thus CPS application completion time. The proposed model can easily be adopted in vehicular ad hoc networks.

3) DATA TRANSFER MANAGER

The data transfer manager is responsible for data transmission using an underlying communication technology such as Wi-Fi Direct. The proposed network layer is independent of the communication technology. It can be modified to support one or several communication technologies, such as Wi-Fi or ZigBee.

4) ROUTING TABLE

A routing table maintains multiple routes to a destination. A routing entry in a routing table includes the following information:

1) Next node
2) Destination node
3) Average number of dropped and lost packets
4) Link quality
5) Link lifetime

5) ROUTING MANAGER

In order to reduce the transmission energy consumption, a transmission power control mechanism is used. It is assumed that every node can transmit at multiple transmission power levels. For each transmission power level TPL_i, each node maintains a separate routing table RTP_i. In order to discover nodes, the discovery mechanism described in the previous section is used. Based on the discovered information, the link quality and link lifetime for each node accessible at each transmission power level are determined. The procedure of estimating the link quality and lifetime was described in the previous section.

In order to forward data to a destination, a source node accesses the lowest power routing table RTP_{min} and selects a route whose lifetime is greater than or equal to the estimated data transfer time. If there are possible multiple routes, the route with the maximum probability is selected. The route selected from RTP_{min} can improve the energy efficiency and network capacity. This is because communication at lower transmission power reduces the transmission energy consumption and increases the number of parallel transmissions and therefore the network's capacity. If no route exists to the destination in the lowest power routing table RTP_{min}, then a routing table for the next transmission power level $\text{RTP}_{\text{min}+1}$ is accessed. The route selection mechanism is described below.

An alternative approach is to calculate the weight of each route on the basis of the link quality, link lifetime, and energy consumption and then to select the route with the maximum weight.

B. RMS MIDDLEWARE LAYER

Middleware layer is responsible for the discovery, monitoring, migration, and allocation of resources. It receives application tasks from users and allocates tasks to nodes on the basis of network- and node-level information. Each element of the middleware layer is described below.

1) TASK QUEUE MANAGER

The task queue manager manages the task queue. It receives tasks from users and inserts them into the task queue, and it also collects task results upon successful completion and communicates them to the users.

Algorithm 1 A Route Selection Mechanism

Receive data transmission request
Previously estimated data transfer time $E_{DTT\_previous} = INFINITY$
Probability of route lifetime $P\_Route_{LT\_Previous} = 0$
$RTP_{i=1}$
For each routing table RTP_i
  For each route in routing table RTP_i
    Estimate data transfer time E_{DTT} using data transfer time estimation model
    If ($\exists$ a route with link lifetime $Route_{LT} \geq$ Estimated data transfer time E_{DTT}) and
      ($E_{DTT} < E_{DTT\_previous}$) and (probability of route lifetime $P\_Route_{LLT} > P\_Route_{LT\_Previous}$)
        Select a route for data transmission
        $E_{DTT\_previous} = E_{DTT}$
        $P\_Route_{LT\_Previous} = P\_Route_{LLT}$
        RTP_{i++}

2) TASK QUEUE

The task queue has information about tasks and allocated nodes:

1) Task ID
2) Task type
3) Code size
4) Input data size
5) Output data size
6) Node ID
7) Task status
8) Task progress

3) RESOURCE POOL

The resource pool maintains the characteristics of each member of the mobile ad hoc cloud. It also maintains network-level information, such as the link quality and lifetime discovered and monitored by the network layer:

1) Destination node ID
2) Clock cycles per instruction
3) Clock cycle time
4) CPU energy consumption
5) System overhead
6) Task queue waiting time
7) Available memory
8) Available battery power
9) Transmission power level
10) Link quality
11) Link lifetime
12) Average number of dropped and lost packets

4) DISCOVERY AND MONITORING MANAGER

The discovery and monitoring manager integrated with the network layer is divided in two subcomponents, namely, node discovery and monitoring and network discovery and monitoring. The network discovery and monitoring manager is part of the network layer and is discussed in a related section.

The node discovery and monitoring manager discovers and monitors mobile devices in the communication range and tasks being executed on mobile nodes. The information discovered about nodes is stored in a resource pool, and tasks are stored in the task queue. Both the resource pool and the task queue are regularly updated by the monitoring manager, which triggers the migration manager under predefined conditions, such as node overutilization or underutilization or poor progress of a task.

The node discovery and monitoring manager has a distributed architecture. It is deployed on each node of the mobile ad hoc cloud. In order to discover and monitor resources, the discovery and monitoring manager on each node communicates the following messages with neighboring nodes.

a: NODE INFORMATION MESSAGE

A node information message (NIM) is used to discover and monitor nodes in the coverage area. It is also used to keep track of nodes in or out of the coverage range. Each node broadcasts a NIM every x intervals. The node that receives a NIM adds a new entry or updates an existing entry in a resource pool. If a node does not receive a NIM from a member node during m intervals, the member node is declared as unavailable in the resource pool.

Message type	Node ID	CPI	CCT	Broadcast ID

Node information message

b: NODE INFORMATION UPDATE MESSAGE

A node information update message (NIUM) is used to communicate dynamic information about a node, such as task queue size and available memory. In order to reduce the communication overhead, each node broadcasts a NIUM when the task queue size, available memory, or available battery power crosses a threshold. To communicate dynamic information to new nodes, the following mechanism is used.

On receiving a NIM:
If (node ID in the NIM does not exist in the resource pool)
Add a new entry to the resource pool

Send a node dynamic information request message (NIRM)

A node receiving a NIRM sends a NIUM

Message type	Source ID	Destination ID

Node information request message

Message type	Node ID	Queue waiting time	Available memory	Battery power	Destination node ID

Node information update message

c: MEMBERS INFORMATION MESSAGE

A members information message (MIM) is used to share resource pool information with neighboring nodes. Each node periodically broadcasts a MIM.

Message type	Node ID	Member ID	CPI	CCT
Member ID	CPI	CCT	Broadcast ID	

Members information message

In order to share dynamic information about member nodes, each node broadcasts a member information update message (MIUM) when the task queue size, available memory, or available battery power crosses a threshold. To communicate dynamic information to new nodes, the following mechanism is used.

On receiving a MIM:

If (member ID in MIM does not exist in the resource pool)

Add a new entry to the resource pool

Send members dynamic information request message

A node receiving an MDIRM sends a MIUM

Message type	Source ID	Member ID
Member ID	Member ID	Destination ID

Members dynamic information request message

Message type	Source ID	Member ID	Queue waiting time
Available memory	Available battery power	Destination or broadcast ID	

Members information update message

d: TASK INFORMATION MESSAGE

A task information message (TIM) is used to communicate the list of allocated or executing tasks at a node and the status of each task. This information is used by the resource allocation system to migrate or reallocate a task to another node in order to improve resource utilization, task performance, or energy efficiency. A TIM can also be used to detect task failure. If a node does not receive a TIM during z intervals, it will assume the failure of a task and can take the necessary action. A TIM is sent every z intervals to every node that has been sent the task for execution. If the task list cannot fit in a single message, then multiple messages are sent. With a centralized architecture, TIMs are sent to the node hosting the resource allocation service.

Message type	Node ID	Task ID	Task status
Task ID	Task status	Destination ID	

Task information message

5) RESOURCE ALLOCATION SERVICE

The resource allocation service assigns independent application tasks to nodes in mobile ad hoc clouds (Algorithm 2). It accesses task information from the task queue and node information from the resource pool and finds a suitable node that fulfills the task requirements. During matching, it considers the CPU speed, queue size, link quality, link lifetime, and transmission power required to send the data. The selected node ID and task ID are sent to the dispatcher, which then dispatches the selected task to the selected node.

Here is the step-by-step process for allocating a task (Fig. 6):

(1) A SCN submits task information to the task queue manager deployed on the SMN.
(2) The resource allocation service deployed on the SMN selects a node to execute the task.
(3) The ID of the selected node is sent to a dispatcher deployed on the SCN.
(4) The task code and related data are sent to a selected SPN.
(5) Upon successful completion of the task, the result is sent to the SCN.
(6) The task status is updated in the task queue deployed on the SMN.

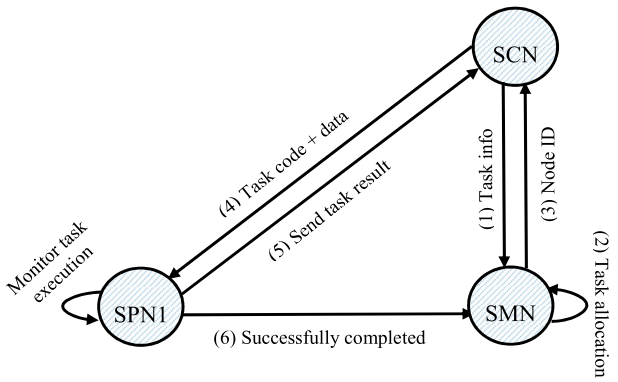


FIGURE 6. Step-by-step process for allocating a task.

Cost estimation models and the resource allocation algorithm that selects a node to execute a task are described below.

COST ESTIMATION MODELS

ESTIMATED COMPLETION TIME (E_{CT}) OF TASK T_i ON THE NODE N_k

The key factors that contribute to the task completion time are the task execution time (E_{ET}) and the data transfer time

Algorithm 2 The Proposed Resource Allocation Algorithm

Input: Set of tasks $T_i \in T$ in the task queue
Set of nodes $N_k \in N$ in the resource pool
Network information stored in RTP_i at the network layer

Output: Assignment of task T_i to service provider node $N_{\text{spn}} \in N$

```
Repeat until task queue is not empty
{
For each task T_i ∈ T submitted by SCN node N_C
{
    Previously estimated task completion time E_CT(T_i,N_pre) = 0
    RTP_i=1
     Probability of route lifetime P_Route_LLT_previous = 0
     Service provider node N_spn = null
     For each node N_k ∈ N in routing table RTP_i
       {
       For each route to N_k in routing table RTP_i
            Estimate task completion time using task completion time estimation model:
               E_CT(T_i,N_k) = E_ET(T_i,N_k) + E_DTT(T_i,N_k)
            Estimate processing energy consumption:
               E_EC(T_i,N_k) = (α × E_PT(T_i, N_k))
            If (∃ a route to N_k with lifetime Route_LT ≥ Estimated data transfer time E_DTT(T_i,N_k) &
                        (E_CT(T_i,N_k) < E_CT(T_i,N_pre)) & Probability of route lifetime P_Route_LLT >P_Route_LLT_previous
                  Assign T_i to N_k
                  E_CT(T_i, N_pre) = E_CT(T_i,N_k)
                  P_Route_LLT_Previous = P_Route_LLT
     }
     RTP_i++
     }
      N_spn = N_k
     Send task-node assignment (T_i,N_spn) to service consumer node N_C
     Update the task queue and resource pool with task-node assignment (T_i,N_spn) and
      completion time E_CT(T_i,N_spn)
}
```

(E_DTT). In order to estimate the completion time of task T_i on node N_k, we use

$$\text{E}_{CT}(\text{T}_\text{i}, \text{N}_\text{k}) = \text{E}_{ET}(\text{T}_\text{i}, \text{N}_\text{k}) + \text{E}_{DTT}(\text{T}_\text{i}, \text{N}_\text{k}). \tag{5}$$

The process of estimating the execution time (E_{ET}) of task T_i on node N_k, is as follows.

The task execution time consists of the CPU processing time and task queue time, which is defined as the time that a task waits in a queue. In order to estimate the execution time (E_{ET}) of task T_i on node N_k, we use

$$\text{E}_{ET}(\text{T}_\text{i}, \text{N}_\text{k}) = \text{E}_{PT}(\text{T}_\text{i}, \text{N}_\text{k}) + \text{E}_{QT}(\text{T}_\text{i}, \text{N}_\text{k}), \tag{6}$$

which considers the CPU processing time (E_{PT}) and the task queue time (E_QT).

The CPU processing time (E_{PT}) of task T_i on node N_k, is estimated using

$$\text{E}_{PT}(\text{T}_\text{i}, \text{N}_\text{k}) = \text{I}_\text{i} \times N^k_{CPI} \times N^k_{CCT}, \tag{7}$$

where I_i is the number of instructions in task Ti, N^k_{CPI} are the clock cycles per instruction of node N_k, and N^k_{CCT} is the clock cycle time of node N_k.

In order to estimate the task waiting time at queue, we use

$$\text{E}_{QT}(\text{T}_\text{i}, \text{N}_\text{k}) = \text{E}_{PTE}(\text{T}_\text{i}, \text{N}_\text{k}) + \text{E}_{PT\_TQ} + ((\text{m} + 2) \times \phi), \tag{8}$$

$$\text{E}_{PTE}(\text{T}_\text{i}, \text{N}_\text{k}) = (\text{I}_\text{i} - IE_\text{i}) \times N^k_{CPI} \times N^k_{CCT}, \tag{9}$$

where ϕ is the time spent by the system in allocating and deallocating tasks to a CPU and IE_i is the number of processed instructions.

The CPU processing time for tasks waiting in a queue is estimated through

$$E_{PT\_TQ} = \sum\nolimits_{n=1}^{m} E_{PT}(T_n, \text{N}_\text{k}), \tag{10}$$

which also considers the estimated execution time of tasks waiting for execution on the CPU. Here m is the number of tasks in the queue.

ESTIMATED DATA TRANSFER TIME (E_{DTT})

The data transfer time is estimated using

$$E_{DTT} = (P_{kts\times} Pkt_{size}) \div LQ_{mk} + (P_{kts\_AvgDL\times} Pkt_{size}) \div LQ_{mk} \quad (11)$$

where P_{kts} is the number of packets, Pkt_{size} is the packet size, LQ_{mk} is the link quality between nodes N_m and N_k, and P_{kts_AvgDL} is the average number of dropped and lost packets. For details, refer to Section III.A.2.

ESTIMATED ENERGY CONSUMPTION (E_{EC})

In order to estimate the energy consumption, we use

$$E_{\mathrm{EC}}(T_i, N_k) = (\alpha \times E_{PT}(T_i, N_k)) + (\beta \times P(T_i)), \quad (12)$$

where α is the CPU energy consumption per unit of time and β is the wireless channel energy consumption per transmitted packet.

The equation considers CPU energy consumption and communication energy consumption. CPU energy consumption is estimated using [48]

$$\alpha = \mathrm{P}_{static} + \mathrm{P}_{dynamic}, \quad (13)$$

where the dynamic CPU power consumption is

$$\mathrm{P}_{dynamic} = A \times C \times V^2 \times F, \quad (14)$$

where A is the number of active logic gates, C is the total capacitance load, V represents the voltage, and F represents the frequency.

6) DISPATCHER

A dispatcher is deployed on each node. It performs several functions, including the following:

(1) It dispatches task code and data to a node selected by the resource allocation service.
(2) It interfaces with the local execution and operating environment to enable the execution of a received task.
(3) Upon successful completion of the task, it sends the result to the SCN and status in the resource pool and the task queue.

7) TASK MIGRATION

A task migration process is initiated for the following events:

(1) A node is about to fail owing to mobility or low battery power.
(2) A node is underutilized or overutilized.
(3) A more powerful or suitable node joins the network.

The migration process consists of three phases (Fig. 7).

(1) *Task Migration Setup Phase*. The task migration service must decide which task to migrate and where to migrate it to. In order to find a suitable target node, the task migration service uses the resource allocation service. Simultaneously, a task selected for migration is cloned at a source node. The resource allocation service sends the ID of the selected node to a source node to handle the migration of the task.

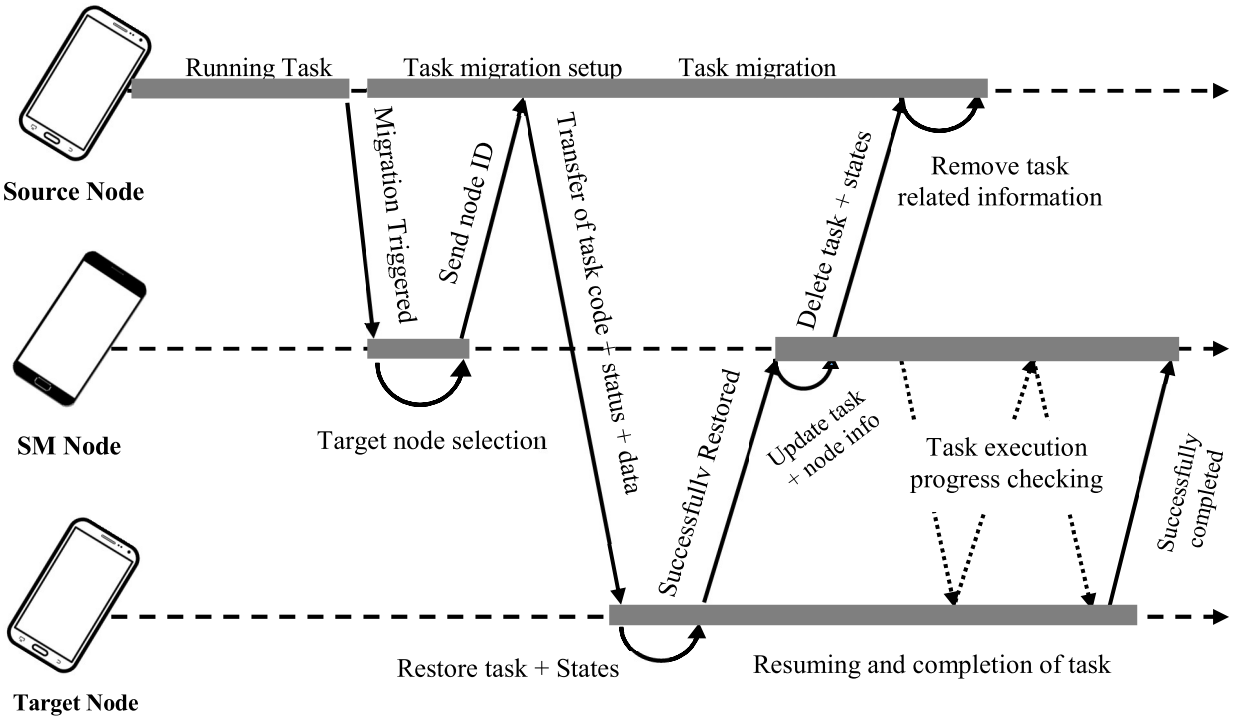


FIGURE 7. Phases of the migration process.

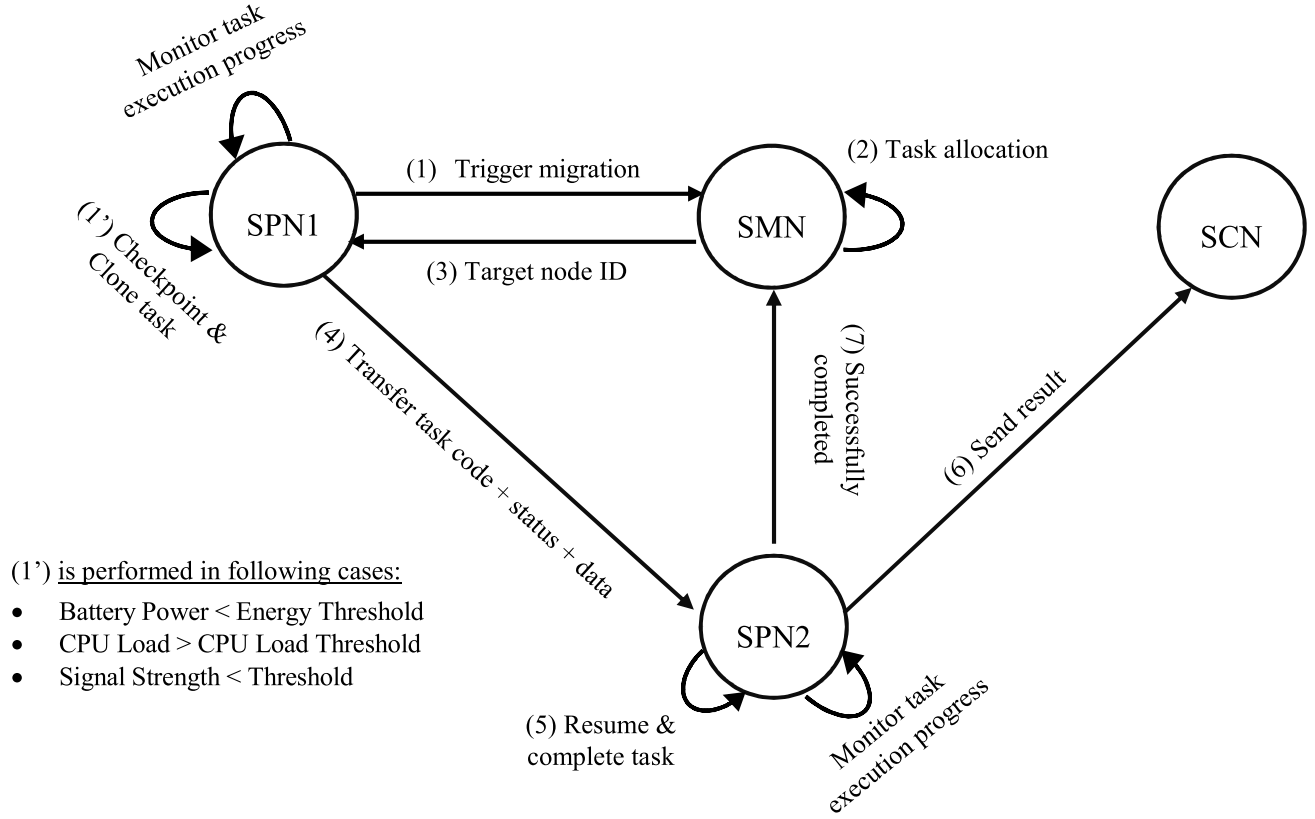


FIGURE 8. Step-by-step process for migration of a task.

(2) *Task Migration Phase*. During this phase, the source node dispatches the cloned task image and data to a target node, to be restored and resumed.
(3) *Resuming and Completion Phase*. As soon as the task information starts to arrive, the target node starts to restore the task. When all the information has been received, the execution of the task is resumed on the target node.

The step-by-step process of migrating a task (Fig. 8) is as follows:

(1) SPN1 monitors the node and task progress and triggers the migration process by sending a request to the resource allocation service deployed on the SMN. The request includes the ID of the existing SPN and task information.
(2) The resource allocation service deployed on the SMN selects another SPN2 to execute the task.
(3) The ID of the selected node is sent to the dispatcher on SPN1.
(4) The task code, related data, and current status are sent to SPN2.
(5) The task resumes execution on SPN2.
(6) Upon successful completion of the task, the result is sent to the SCN and the status is updated in the task queue deployed on the SMN.

C. WORKING OF RMS

In this section, we briefly describe the workings of the proposed RMS. Detailed mechanisms employed by each component are discussed in the related sections.

The user node or SCN submits task information to the task queue via the task queue manager. The node- and network-level information collected by the discovery and monitoring manager is stored in the resource pool. The resource allocation service accesses task information from the task queue and node information from the resource pool and selects an SPN that fulfills the task requirements. The IDs of the task and selected SPN are sent to the dispatcher deployed on the SCN, which transfers the task code and related data to the selected SPN. The dispatcher deployed on the SPN facilitates the execution of the received task and, upon successful completion, sends the result to the SCN and updates the status at the resource pool and the task queue.

IV. PRELIMINARY RESULTS

In order to demonstrate the feasibility of the proposed system, Contiki [32] was used to implement the resource allocation scheme and the Cooja [33], [34] simulator was used to implement network layer functions. Altogether, 20 nodes were randomly deployed, and numerous tasks of varying sizes were

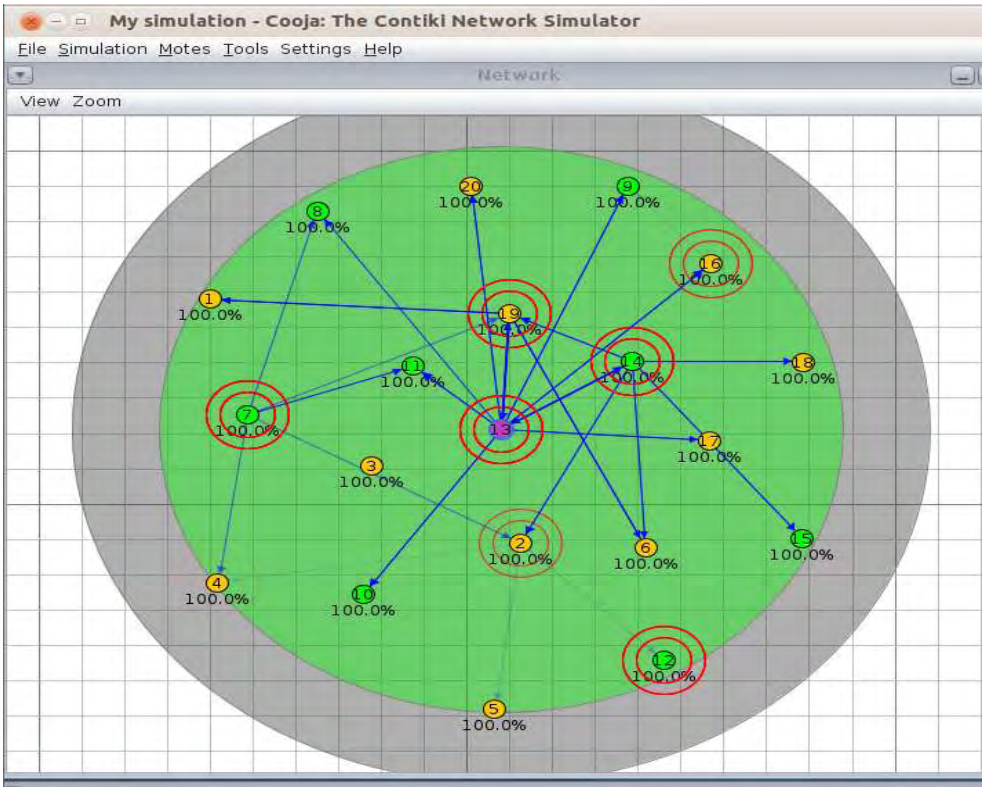


FIGURE 9. Network scenario.

used to reflect data-intensive applications, such as distributed face recognition. Nodes were divided into three categories: SMNs, SPNs, and SCNs. The SMN was responsible for allocating tasks submitted by SCNs. Based on the resource allocation policy, SPNs were selected to execute the tasks. Cooja's mobility plugin was used to generate group mobility patterns. The simulation parameters are given in Table 2, and the network scenario is shown in Fig. 9.

TABLE 2. Simulation parameters.

Platform	Cooja Simulator Contiki 3.0
Simulation time	3,600 s
Number of nodes	20
Mote types	3
Simulation area	1,200 m × 1,200 m
Packet size	512 bytes
MAC	IEEE 802.11
Transport protocol	User Datagram Protocol (UDP)
Radio medium	UDGM [33]
Transmission range	180 m
INT range	20 m

The performance of the proposed scheme is compared with that of a heterogeneity-aware task allocation (HTA) scheme [49], which allocates tasks on the basis of node processing power and available energy.

In HTA, nodes use the maximum transmission power for communication. The scheme also does not consider network-level information, such as link quality and lifetime.

The performance is evaluated in terms of accumulative task completion time (ATCT):

$$ATCT = \sum_{i=1}^{n} E_{CT}(T_i). \tag{15}$$

The completion time for task $E_{CT}(T_i)$ consists of the CPU execution time and the data transfer time. The task completion time estimation model is described in Section III.

Scenario 1. The proposed scheme makes allocation decisions on the basis of the estimated task completion time and energy consumption. The task completion time comprises the task execution time and the data transfer time. The task execution time includes the CPU processing time, queue time, and system overhead. The simulation results in Fig. 10 show that the performance of the proposed scheme is better by 10–15%. With 10 tasks, there is a minor improvement, but as the number of tasks increases, the performance of the scheme improves. The HTA scheme performs poorly because it does not consider data transfer costs. It selects nodes with

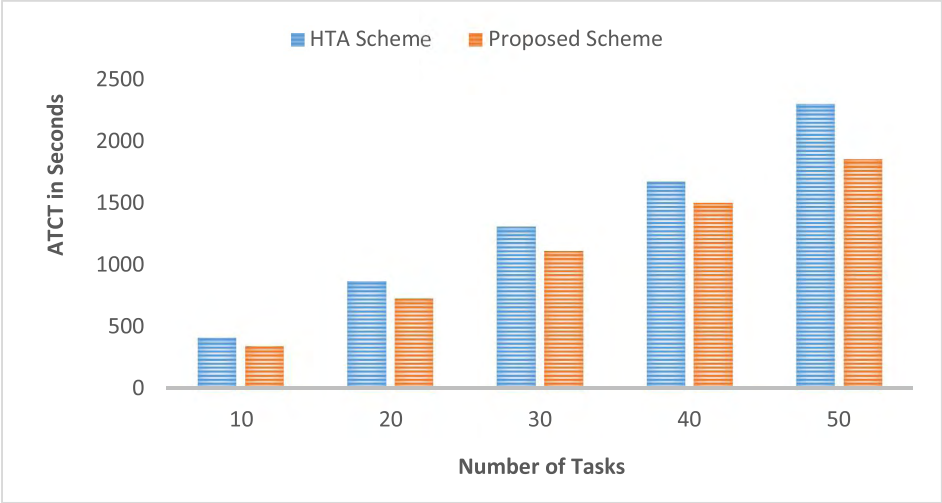


FIGURE 10. ATCT for Scenario 1.

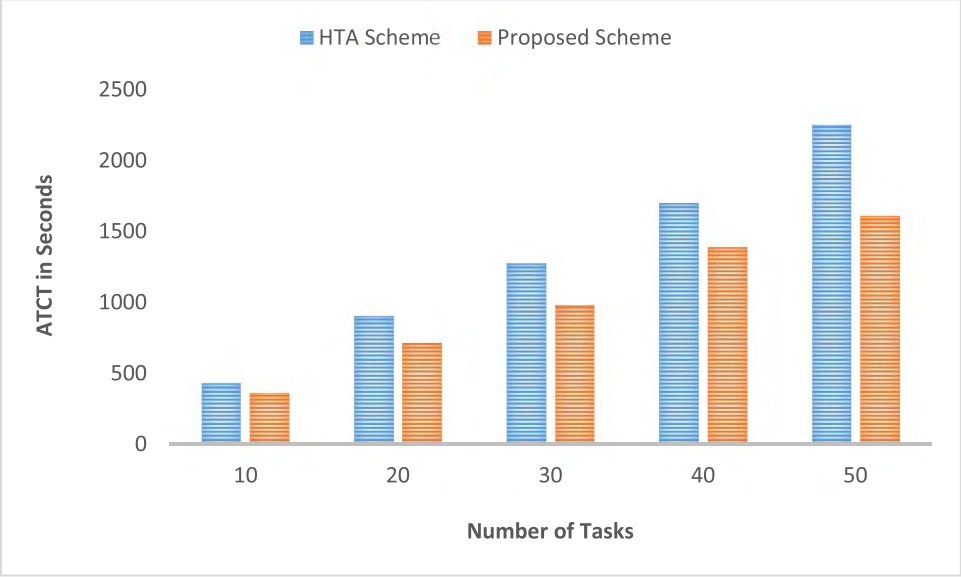


FIGURE 11. ATCT for Scenario 2.

the highest processing power and available energy. In several cases, nodes with the highest processing power and available energy were connected via low-quality communication links, which increased the data transfer time and thus the task completion time.

Scenario 2. The proposed scheme makes allocation decisions on the basis of the estimated task completion time, energy consumption, and link lifetime. The results for Scenario 2 in Fig. 11 show that, for the proposed scheme, the performance improves by 18–34%. This is because the proposed scheme selects SPNs with a route lifetime greater than or equal to the estimated data transfer time of tasks. This reduces route rediscovery and reselection costs in case of link failure or global node mobility. The transmission power control mechanism increases the number of parallel transmissions and thus the network capacity. This reduces the overall data transfer cost and thus the ATCT. HTA does not consider link lifetime and transmission power, so it select nodes with an unstable connectivity and reduced network capacity.

Transmission Energy Consumption: The transmission energy consumption of the proposed scheme was compared with that of a minimum hop-based task allocation (MinHop) scheme [9] as well as HTA [49]. The MinHop scheme assigns a task to a node on the basis of the number of hops. It uses the maximum transmission power. In the proposed scheme,

nodes use three transmission power levels for communication. Two scenarios were defined. In both scenarios, 20 nodes were deployed. However, the distance between the nodes in Scenario 3 was uniform, whereas in Scenario 4, nodes were deployed in groups of numerous sizes.

The transmission energy consumption for Scenarios 3 and 4 is given in Figs. 12 and 13, respectively. The performance of the proposed scheme in Scenario 3 is poor because most of the nodes were accessible only at the maximum transmission power. So, like MinHop and HTA, in the proposed scheme, the nodes used the maximum transmission power for communication. In addition, because of the multiple transmission power levels, the amount of control traffic in the proposed scheme was three times more than in the other schemes.

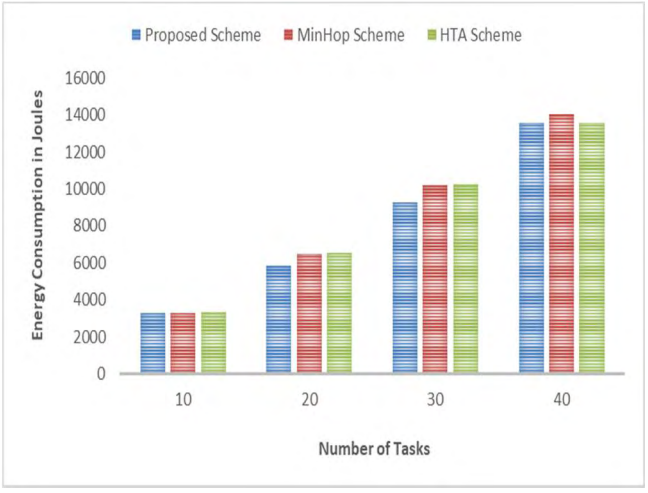


FIGURE 12. Transmission energy consumption for Scenario 3.

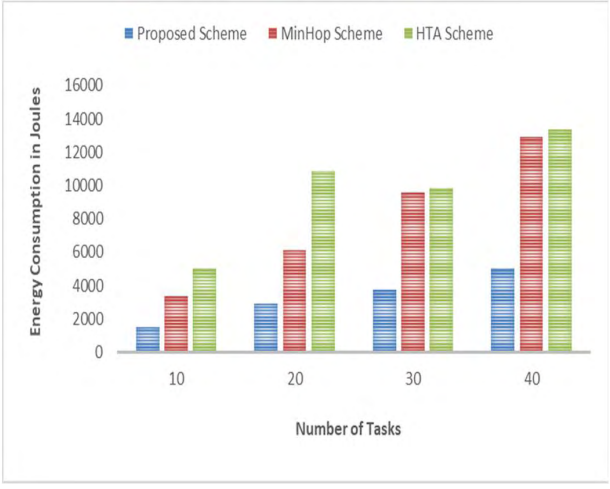


FIGURE 13. Transmission energy consumption for Scenario 4.

In Scenario 4, the proposed scheme outperforms the other two schemes. This is because nodes were deployed in groups. Several nodes were accessible at lower transmission power levels. The proposed scheme allocated tasks to nodes accessible at the minimum transmission power, which significantly reduced the transmission energy.

The transmission energy consumption of the proposed scheme using three transmission power levels compared to the proposed scheme always using the maximum transmission power is given in Fig. 14. The results indicate that the transmission power control mechanism significantly reduces the transmission energy.

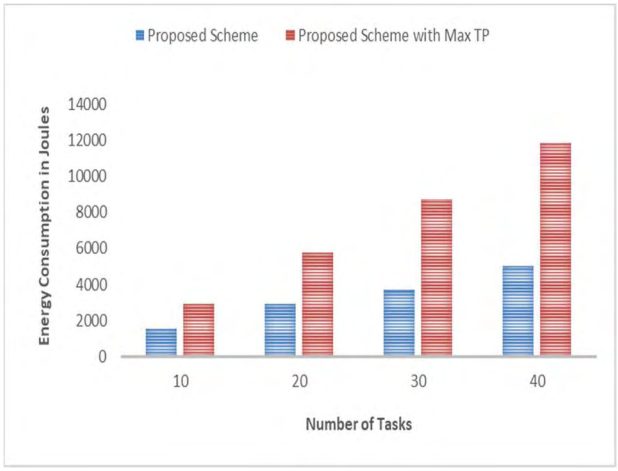


FIGURE 14. Transmission energy consumption for Scenario 4.

V. CONCLUSION

RMSs are responsible for the discovery, monitoring, migration, and allocation of network and system resources. They are an integral part of mobile ad hoc clouds and play a key role in the application and system performance. Research into resource management for mobile ad hoc clouds is still in a preliminary phase, and very few schemes based on a decentralized architecture have been proposed in order to address issues such as node mobility, energy management, and task failure. Most of these schemes do not consider the network environment, task queue size, or CPU overhead. Very few schemes consider the network environment, but they do not consider link quality, link lifetime, or the migration and reallocation of tasks.

In this paper, we proposed an energy-efficient RMS to support the execution of mobile CPS applications on a mobile ad hoc cloud. The proposed system consisted of two layers: a network layer and a middleware layer. The network layer provides ad hoc network and communication services to the middleware layer and shares the collected information in order to enable efficient and robust resource management decisions. The middleware layer is responsible for the discovery, monitoring, migration, and allocation of resources.

The new system is different from the systems proposed in the literature because it focuses on mobile CPS applications and energy-efficient communication between tasks. In addition, it aims to address task failure by adopting a failure avoidance mechanism, and it uses a cross-layer design to support

effective resource management decisions. Our research on transmission power control mechanisms is used to reduce the transmission energy consumption and increase concurrent transmissions in the network. The new system considers the processing power, queue size, system overhead, link quality, and link lifetime to reduce execution times, data transfer costs, and energy consumption.

ACKNOWLEDGMENTS

I would like to acknowledge contribution of Mr. Mpyana Mwamba Merlec of Korea University.

REFERENCES

[1] S. C. Shah, "Recent advances in mobile grid and cloud computing," *Intell. Automat. Soft Comput. J.*, vol. 24, no. 2, pp. 285–298, Feb. 2017.

[2] K. A. Hummel and G. Jelleschitz, "A robust decentralized job scheduling approach for mobile peers in ad-hoc grids," in *Proc. IEEE Int. Symp. Cluster Comput. Grid*, May 2007, pp. 461–470.

[3] D. C. Chu and M. Humphrey, "Mobile OGSI.NET: Grid computing on mobile devices," in *Proc. IEEE-ACM Workshop Grid Comput.*, Nov. 2004, pp. 182–191.

[4] C. Li and L. Li, "Utility-based scheduling for grid computing under constraints of energy budget and deadline," *Comput. Standards Interfaces*, vol. 31, no. 6, pp. 1131–1142, Nov. 2009.

[5] C. Li and L. Li, "Tradeoffs between energy consumption and QoS in mobile grid," *J. Supercomput.*, vol. 55, no. 3, pp. 367–399, 2011.

[6] C. Li and L. Li, "Collaboration among mobile agents for efficient energy allocation in mobile grid," *Inf. Syst. Frontiers*, vol. 14, no. 3, pp. 711–723, Jul. 2012.

[7] A. T. A. Gomes, A. Ziviani, L. S. Lima, and M. Endler, "DICHOTOMY: A resource discovery and scheduling protocol for multihop ad hoc mobile grids," in *Proc. 7th IEEE Int. Symp. Cluster Comput. Grid (CCGrid)*, May 2007, pp. 719–724.

[8] V. V. Selvi, S. Sharfraz, and R. Parthasarathi, "Mobile ad hoc grid using trace based mobility model," in *Advances in Grid and Pervasive Computing* (Lecture Notes in Computer Science), vol. 4459. Berlin, Germany: 2007, pp. 274–285.

[9] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *J. Supercomput.*, vol. 71, no. 8, pp. 3009–3036, 2015.

[10] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. 13th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2012, pp. 145–154.

[11] J. M. Rodriguez, C. Mateos, and A. Zunino, "Energy-efficient job stealing for CPU-intensive processing in mobile devices," *Comput. J.*, vol. 96, no. 2, pp. 87–117, 2012.

[12] H. Viswanathan, E. K. Lee, and I. Rodero, "Uncertainty-aware autonomic resource provisioning for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 8, pp. 2363–2372, Aug. 2015.

[13] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," *Comput. Elect. Eng.*, vol. 44, pp. 218–240, May 2015.

[14] S. C. Shah and M. S. Park, "An energy-efficient resource allocation scheme for mobile ad hoc computational grids," *J. Grid Comput.*, vol. 9, no. 3, pp. 303–323, Apr. 2011.

[15] S. C. Shah, Q.-U.-A. Nizamani, S. H. Chauhdary, and M.-S. Park, "An effective and robust two-phase resource allocation scheme for interdependent tasks in mobile ad hoc computational grids," *J. Parallel Distrib. Comput.*, vol. 72, no. 12, pp. 1664–1679, Dec. 2012.

[16] S. C. Shah, "Energy efficient and robust allocation of interdependent tasks on mobile ad hoc computational grid," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 5, pp. 1226–1254, May 2014.

[17] A. Bröring, S. K. Datta, and C. Bonnet, "A categorization of discovery technologies for the Internet of Things," in *Proc. 6th Int. Conf. Internet Things*, Nov. 2016, pp. 131–139.

[18] L. H. Nunes, J. C. Estrella, C. Perera, S. Reiff-Marganiec, and A. C. B. Delbem, "Multi-criteria IoT resource discovery: A comparative analysis," *J. Softw. Pract. Exper.*, vol. 47, no. 10, pp. 1325–1341, 2016.

[19] *Stuart Cheshire and Mark Krochmal*. document RFC 6762, IETF, 2013. [Online]. Available: https://tools.ietf.org/html/rfc6762

[20] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny Internet nodes," *IEEE Internet Comput.*, vol. 16, no. 2, pp. 62–67, Mar./Apr. 2012.

[21] Y. Goland, T. Cai, P. Leach, and Y. Gu. (Oct. 28, 1999). Internet Draft: Simple Service Discovery Protocol/1.0 Operating Without an Arbiter. IETF. [Online]. Available: https://tools.ietf.org/html/draft-cai-ssdp-v1-03

[22] Z. Shelby, M. Koster, C. Bormann, and P. van der Stok. (Jul. 7, 2018). Internet Draft: CoRE Resource Directory. IETF. [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-resource-directory-07

[23] T. Jaffey, J. Davies, and P. Beart. (2016). Hypercat 3.00 Specification. Hypercat Limited. [Online]. Available: http://www.hypercat.io/standard.html

[24] S. Jirka and D. Nüst, "Sensor instance registry,"Open Geospatial Consortium, Wayland, MA, USA, OGC Discussion Paper 10-171, 2010.

[25] Z. Li, R. Chen, L. Liu, and G. Min, "Dynamic resource discovery based on preference and movement pattern similarity for large-scale social Internet of Things," *IEEE Internet Things J.*, vol. 3, no. 4, pp. 581–589, Aug. 2016.

[26] P.-K. Huang, E. Qi, M. Park, and A. Stephens, "Energy efficient and scalable device-to-device discovery protocol with fast discovery," in *Proc. IEEE Int. Workshop Internet-Things Netw. Control (IoT-NC)*, Jun. 2013, pp. 1–9.

[27] S. Lu, S. Shere, Y. Liu, and Y. Liu, "Device discovery and connection establishment approach using ad-hoc Wi-Fi for opportunistic networks," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PERCOM Workshops)*, Mar. 2011, pp. 461–466.

[28] A. N. Mian, R. Baldoni, and R. Beraldi, "A survey of service discovery protocols in multihop mobile ad hoc networks," *IEEE Pervasive Comput.*, vol. 8, no. 1, pp. 66–74, Jan. 2009.

[29] W. Sun, Z. Yang, X. Zhang, and Y. Liu, "Energy-efficient neighbor discovery in mobile ad hoc and wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1448–1459, 3rd Quart., 2014.

[30] C. N. Ververidis and G. C. Polyzos, "Service discovery for mobile ad hoc networks: A survey of issues and techniques," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 3, pp. 30–45, 3rd Quart., 2008.

[31] V. Kawadia and P. R. Kumar, "Power control and clustering in ad hoc networks," in *Proc. IEEE INFOCOM, 22nd Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Mar./Apr. 2003, pp. 459–469.

[32] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, Nov. 2004, pp. 455–462.

[33] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 641–648.

[34] A. Velinov and A. Mileva, "Running and testing applications for contiki OS using COOJA simulator," in *Proc. Int. Conf. Inf. Technol. Develop. Educ. (ITRO)*, Jun. 2016, pp. 279–285.

[35] Q. Xue and A. Ganz, "Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks," *J. Parallel Distrib. Comput.*, vol. 63, no. 2, pp. 154–165, Feb. 2003.

[36] E. Paraskevas, K. Manousakis, S. Das, and J. S. Baras, "Multi-metric energy efficient routing in mobile ad-hoc networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2014, pp. 1146–1151.

[37] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless Ad-hoc networks," in *Proc. 7th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, vol. 1, 2001, pp. 97–107.

[38] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 11, pp. 1122–1133, Nov. 2001.

[39] S. Doshi, S. Bhandare, and T. X. Brown, "An on-demand minimum energy routing protocol for a wireless ad hoc network," *ACM SIGMOBILE Mobile Comput. Commun.*, vol. 6, no. 3, pp. 50–66, Jul. 2002.

[40] S. Banerjee and A. Misra, "Minimum energy paths for reliable communication in multi-hop wireless networks," in *Proc. 3rd ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, vol. 2, Jun. 2002, pp. 146–156.

[41] J. Gomez, A. T. Campbell, M. Naghshineh, and C. Bisdikian, "PARO: Supporting dynamic power controlled routing in wireless ad hoc networks," *Wireless Netw.*, vol. 9, no. 5, pp. 443–460, Sep. 2003.

[42] M. Maleki, K. Dantu, and M. Pedram, "Lifetime prediction routing in mobile ad hoc networks," in *Proc. IEEE Wireless Commun. Netw.*, Mar. 2003, pp. 1185–1190.
[43] X. Wang, C. Wang, G. Cui, and Q. Yang, "Practical link duration prediction model in vehicular ad hoc networks," *Int. J. Distrib. Sensor Netw.*, vol. 2015, Jan. 2015, Art. no. 2.
[44] E. Y. Hua and Z. J. Haas, "An algorithm for prediction of link lifetime in MANET based on unscented kalman filter," *IEEE Commun. Lett.*, vol. 13, no. 10, pp. 782–784, Oct. 2009.
[45] Q. Guan, F. R. Yu, S. Jiang, and G. Wei, "Prediction-based topology control and routing in cognitive radio mobile ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 9, pp. 4443–4452, Nov. 2010.
[46] P. Vijayalaxmi, E. Ravindra, V. V. Kohir, and V. D. Mytri, "Mobility Prediction algorithm to improve the Routing performance in MANET," *Int. J. Comput. Sci. Netw. Secur.*, vol. 14, no. 9, pp. 39–44, Sep. 2014.
[47] H. Yang, M. Yu, and X. Zeng, "Link available time prediction based GPSR for vehicular ad hoc networks," in *Proc. IEEE 14th Int. Conf. Netw., Sens. Control (ICNSC*, May 2017, pp. 293–298.
[48] L. Wang *et al.*, "Energy-aware parallel task scheduling in a cluster," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1661–1670, Sep. 2013.
[49] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, and M. Imran, "Heterogeneity-aware task allocation in mobile ad hoc cloud," *IEEE Access*, vol. 5, pp. 1779–1795, Feb. 2017.

SAYED CHHATTAN SHAH (SM'14) received the Ph.D. degree in computer science from Korea University in 2012 and the M.S. degree in computer science from the National University of Computer and Emerging Sciences in 2008. He held faculty positions with the Seoul National University of Science and Technology, Korea University, Dongguk University, Hamdard University, and Isra University. He is currently an Assistant Professor of computer science with the Department of Information Communication Engineering, Hankuk University of Foreign Studies (HUFS), Seoul, South Korea. He is also the Director of the Mobile Grid and Cloud Computing Laboratory. Prior to joining HUFS, he was a Senior Researcher with the Electronics and Telecommunications Research Institute, South Korea, and an Engineer with the National Engineering and Scientific Commission, Pakistan. His research interests lie in the fields of parallel and distributed computing systems, mobile computational clouds, and ad hoc networks.

He is a member of the IEEE Communications Society, the International Telecommunication Union, and the European Research Council. He was the conference chair and on the program committees of various international conferences. He is currently an Associate Editor of *Information Processing Systems and Intelligent Automation and Soft Computing*.