RANCE: A Randomly Centralized and On-Demand Clustering Protocol for Mobile Ad Hoc Networks

Xi Chen, Gang Sun, *Member, IEEE*, Tao Wu, Ling Liu, Hongfang Yu, *Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—LEACH-like clustering protocols focus mainly on the low-power, low-rate, and low-wakeup network applications, and work in a multiround clustering strategy that causes frequent handovers of cluster heads (CHs), thus less support for real-time services that require stable cluster topologies. Besides, these protocols are faced with respective drawbacks, such as suboptimality of selected heads, costly node-base station (BS) energy overheads, lack of runtime cluster maintenance, etc. This article proposes RANCE, a randomly centralized and on-demand clustering protocol, aiming at prolonging nodes' clustered time to support internodes collaboration while being energy efficient in mobile ad hoc networks. First, RANCE designs a randomly centralized CH selection mechanism in which every node in the local wireless network is eligible to initiate the centralized CH selection, so that the self-organizing characteristics of mobile ad hoc nodes can be utilized for head selection optimization. Second, taking into account the wireless volatility caused by changes of topology, obstacles, signal strength, etc., the fine-grained cluster relationships maintenance is provided by means of multilevel aliveness and adaptive bidirectional heartbeat packets. Third, RANCE works in an event-driven and on-demand manner instead of a time-triggered manner in LEACH-like protocols, to reduce the impact on continuous services caused by frequent CH handovers among all nodes. Simulation results show that RANCE provides longer clustered time (over 99% of nodes' lifetime in networks more than 100 nodes) and good clustering scalability with high consistency at minimum energy cost, and exhibits good potentials in mobile wireless environments that are infrastructureless/poor for continuous missions.**

Manuscript received 9 December 2021; accepted 29 June 2022. Date of publication 5 July 2022; date of current version 21 November 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFC1507005; in part by the China Postdoctoral Science Foundation under Grant 2018M643448; in part by the Sichuan Science and Technology Program under Grant 2022YFG0208, Grant 2022YFG0161, and Grant 2021YFQ0056; and in part by the Fundamental Research Funds for the Central Universities, Southwest Minzu University under Grant ZYN2022003. *(Corresponding author: Xi Chen.)*

Xi Chen is with the School of Computer Science and Engineering, Southwest Minzu University, Chengdu 610041, China, and also with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: cx@swun.edu.cn).

Gang Sun and Hongfang Yu are with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: gangsun@uestc.edu.cn; yuhf@uestc.edu.cn).

Tao Wu is with the School of Computer Science, Chengdu University of Information Technology, Chengdu 610225, China (e-mail: wut@cuit.edu.cn).

Ling Liu is with the College of Electronic and Information, Southwest Minzu University, Chengdu 610041, China (e-mail: lingliu@mail.ustc.edu.cn).

Mohsen Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/JIOT.2022.3188679

***Index Terms*—Cluster head (CH) selection, cluster maintenance, clustering, mobile ad hoc networks (MANETs).**

I. Introduction

MOBILE ad hoc network (MANET) and wireless sensor network (WSN) are widely used in environment monitoring, information gathering, mobile surveillance, disaster rescue, battlefield communication, etc., [1], [2], and are usually deployed in harsh environments hardly within the reach of humans. In such environments, because wireless nodes are usually distant from the base station (BS), direct delivery of messages from nodes to BS can be costly in terms of energy consumption. Clustering is often an effective measure in MANET/WSN to construct a hierarchical logical topology [3], in which member nodes in a cluster only communicate with the cluster head (CH) that conducts distant communication with BS on behalf of the whole cluster. In this way, member nodes do not have to conduct long-range energy-consuming direct communication with BS frequently, so as to enhance energy efficiency and prolong lifetime. LEACH [4] is a typical self-organized clustering protocol that adopts a distributed algorithm to select CHs and construct clusters in a multiround manner where each round has a fixed duration, in which way, energy consumption is expected to be evenly distributed among nodes. LEACH has been actively studied and applied since its emergence. Recent years have seen improvements to LEACH in various aspects, such as how CHs can be optimally selected [5]–[7], how multihop communication can be introduced to further enhance energy efficiency [6], [8]–[12], how node mobility can be supported [9], [13], how machine learning and various swarm intelligence algorithms can be integrated [14]–[19], how special purpose scenarios (such as underwater, IoT, etc.) can be accommodated [11], [20]–[26], etc., thus a series of LEACH-like protocols. Nevertheless, there still exist some shortcomings that prevent their broader applications in volatile MANETs.

1) *Orientation of LEACH-Like Protocols Is Mainly to Low-Continuous Nonrealtime Network Applications:* LEACH-like protocols adopt the time-triggered multiround strategy for CH selection [27], with the purpose to distribute the heavy-duty role of CH among all participating nodes so that energy consumption is expected to be evenly distributed. This results in frequent (and sometimes unnecessary) handovers of CHs among all nodes.

Therefore, these protocols are usually applicable in low-power, low-rate, and low-wakeup network applications, such as environmental data collection in WSNs. The frequent CH handovers in LEACH-like protocols are not conducive to the routing and forwarding of real-time or continuous streaming services (such as video surveillance), nor to the long-term or time-critical collaboration between nodes.

2) *Suboptimality of CH Selection:* The original LEACH adopts a distributed algorithm that allows for every node to self-elect for the role of CH. In this pure distributed framework, nodes are unaware of important metrics (e.g., residual energy) of one another, which are critical for optimal CH selection. Therefore, elected CHs might not always be the best-suited ones. On the other hand, the BS-based centralized CH selection [e.g., LEACH centralized (LEACH-C) [4]] that periodically collects node metrics is expected to handle the suboptimality problem. However, the delivery of node metrics to the distant BS requires higher power long-range communication that introduces extra bandwidth and energy overheads, shortening nodes' lifetime [28].
3) *Lack of Cluster Relationship Maintenance:* Topological dynamics and time variability after cluster formation are not well taken into consideration by LEACH-like protocols. Wireless mobile networks are highly volatile in that there exist instabilities in signal strength and radio coverage caused by distance, terrain, obstacles, mobility, etc., thus, the possible temporary or long-term changes of head-member relationships in a cluster during runtime. These changes must be discovered in a timely fashion so that cluster relationships can be kept as consistent as possible with underlying link-layer or physical-layer topologies. This requires a fine-grained cluster relationship maintenance that is missing in current LEACH-like protocols.

In order to overcome these shortcomings, this article proposes RANCE, a randomly centralized and on-demand clustering protocol for MANETs. We envision that in MANETs, while energy is still a critical metric for a wireless network to operate properly, flexible and long-term internodes collaboration is also a key factor for sophisticated collaborative tasks. To this extent, RANCE pays a special focus on mobile applications that require long-term internodes collaboration organized by clusters in MANETs. RANCE aims at prolonging nodes' clustered time and enhancing consistency between cluster relationships and underlying topologies, while still being energy efficient. This work is summarized as follows.

1) First, RANCE designs a randomly centralized CH selection mechanism in which every node in the local wireless network is eligible to initiate the centralized CH selection, so that the self-organizing characteristics of ad hoc networks can be utilized for CH selection optimization. It is a centralized CH selection without the evolvement of the remote BS, so as to sustain energy efficiency.
2) Second, taking into account the wireless volatility caused by changes of topology, signal strength, mobility, etc., cluster relationships are timely and dynamically maintained by multilevel aliveness and bidirectional heartbeat packets, to provide high accuracy and consistency for cluster relationships.
3) Third, RANCE works in an event-driven and on-demand manner instead of a time-triggered manner commonly used in LEACH-like protocols, to reduce the impact on continuous services caused by unstable forwarding or routing due to frequent CH handovers among all nodes.

The remainder of this article is organized as follows. Related works are summarized in Section II. Section III mathematically analyzes problems when applying LEACH-like protocols in MANETs. Section IV specifies the details of RANCE, including initial clustering, maintenance, hitchhiking, rebuilding, etc., together with the protocol analysis. Section V envisions possible applications of RANCE in mobile surveillance, IoT, etc. Section VI conducts various OMNeT++ simulations to test the functionalities and performance of RANCE, with comparison with LEACH-like protocols. Finally, this article is concluded and future works are envisioned in Section VII.

II. Related Works

LEACH is applied in the distributed clustering in WSNs, where each node generates a threshold $T(n)$ periodically. A stochastic approach is adopted to generate $T(n)$, as follows:

$$T(n) = \begin{cases} \frac{p}{1-p\times\left(r \mod \frac{1}{p}\right)}, & \text{if } n \in G \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where n refers to the nth node, p refers to the percentage of CHs required in the network, r refers to the ongoing round, and G refers to the set of nodes that have not become CHs in the last $1/p$ rounds. Every node also generates a random value and compares it with $T(n)$. If it is smaller than $T(n)$, this node elects itself as a CH and propagates the advertisement to other nodes, which can choose to join the cluster. For every $1/p$ rounds, $T(n)$ is gradually getting larger round by round. Therefore, stochastically all nodes have a very probability to become CHs every $1/p$ rounds. LEACH works in a multiround manner, so that the energy-consuming role of CH is expected to be distributed among all nodes for load balancing. LEACH might suffer from suboptimality during CH selection due to its distributed stochastic approach. In addition, CHs are not uniformly distributed, which leads to nonuniform cluster formation [29]. Therefore, many improvements are made based on LEACH.

A. Centralized Improvements

Introducing centralization during CH selection is an effective measure to solve the suboptimality issue seen in LEACH. LEACH-C [4] is a centralized version of LEACH, where the remote BS collects metrics of nodes and decide the best-suited ones to be CHs in a centralized manner. LEACH-C conducts CH selection by using simulated annealing algorithm to minimize the sum of squared distances between all nodes to reduce energy consumption. The major issue of LEACH-C is the remarkably more energy used for metrics reporting from nodes to BS. Centralized energy-efficient clustering routing protocol (CEECR) [30], which is also a centralized protocol based

TABLE I
CLUSTERING PROTOCOL COMPARISON

Protocol	Method of CH Selection	Criteria for CH Selection	Timing of CH Selection	Super CH	Multi-Hop Communication	Energy Consumption	Cluster Relationship Maintenance
LEACH [4]	distributed	T(n)	round-based	no	no	medium	no
LEACH-C [4]	centralized	sum of squared distances between all nodes	round-based	no	no	high	no
CEECR [30]	centralized	energy, speed, etc.	round-based	no	no	high	no
DMH-LEACH [9]	distributed	T(n), mobility, etc.	round-based	no	yes	medium	no
Fuzzy Logic LEACH [6]	distributed	fuzzy logic	round-based	yes	yes	medium	no
LEACH-R [8]	distributed	T(n), energy rank, etc.	round-based	no	yes	medium	no
DHRP [27]	distributed	residual energy, etc.	hyper-round-based	no	yes	low	no
WINCH [31]	distributed	similar to LEACH-C	round-based	no	no	high, but rechargable	no
RANCE (the proposed protocol)	randomly centralized	application-determined	on-demand and event-driven	no	yes	medium	yes

on LEACH-C, considers node mobility during clustering. The major criteria for CH selection in CEECR are reliability (i.e., nodes with energy above average) and stability (i.e., nodes with speed below average). Another problem faced with CEECR and other centralized protocols is its comparatively poor scalability in larger wireless mobile networks [27].

B. Distributed Improvements

Dynamic multihop LEACH (DMH-LEACH) [9] introduces multihop communication to LEACH, so that the need for energy-consuming long-range single-hop communication between CHs and BS can be reduced. However, simply introducing the multihop communication causes another problem that CHs nearer to BS have to relay traffic sent from further CHs, in addition to the local cluster traffic. This results in severe energy drainage that compromises network lifetime. DMH-LEACH takes advantage of node mobility, and nodes with stronger mobility are more likely to be selected as CHs. This is based on the intuition that nodes with stronger mobility move toward to or away from BS more frequent, so that they do not always stay nearer to BS to relay traffic as CHs. This keeps energy consumption low in multihop communications.

Verma *et al.* [6] proposed the fuzzy logic LEACH. Fuzzy logic is used to select a super CH (i.e., SCH) among all ordinary CHs based on residual energy, mobility, centrality, etc. SCH communicates with BS on behalf of CHs, so that CHs do not have to conduct frequent long range BS-CH communications.

LEACH Relay (LEACH-R) [8] divides every round into two phases, namely, clustering phase and relay phase. The clustering phase is further divided into two subphases: 1) CH selection, which selects CH by means of random value $T(n)$ and local energy rank $D(n)$ and 2) cluster formation, which works almost identical to LEACH except for the exchange of energy information between CH and members to calculate $D(n)$. The relay phase is further divided into two subphases as well: 1) relay selection, which determines the shortest route that consists of several relay nodes (cluster members might also become relays in addition to CHs) and 2) routing, which sends half of the traffic cached locally in the first step, and sends the rest if the ACK of the first half returns. This caching strategy enhances the packet delivery ratio yet increases latency due to multiple transmissions, thus unfriendly to real-time services.

C. Energy Improvements

Neamatollahi *et al.* [27] proposed the dynamic hyper round policy (DHRP), which schedules clustering task to extend the network lifetime and reduce energy consumption, as opposed to the round-based policy adopted in both distributed and centralized LEACH-like protocols. Clustering is only performed at the beginning of each dynamic hyper round (i.e., coarser grained and dynamically reorganized atomic rounds). This reduces energy overheads of round-based CH handovers among nodes by eliminating the unnecessary reclusterings. Baroudi [31] proposed a wirelessly energy-charged scheme (WINCH) to enable robot-assisted wireless energy transfer in WSNs. In WINCH, CHs are selected using LEACH-C, then robots visit the sites frequently as needed, and place themselves in the optimal positions to conduct wireless charging to extend the network lifetime.

Table I summarizes a brief comparison between different LEACH-like clustering protocols. Compared with these protocols, RANCE has remarkable differences in several aspects (see the last line of Table I). It selects CHs in a randomly centralized manner, and considers various application-oriented criteria (e.g., energy and bandwidth) during CH selection that reflect application characteristics. RANCE also selects CHs in an event-driven (e.g., events that indicate lower energy of current CHs) and on-demand (e.g., when cluster members become detached) manner, other than the time-triggered round-based method. Therefore, RANCE reduces unnecessary pauses of continuous services caused by frequent CH handovers, and improves energy efficiency. Last but not least, RANCE provides cluster relationship maintenance seldom seen in previous LEACH-like protocols to eliminate at large the inconsistency

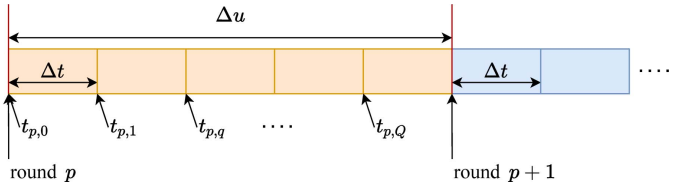


Fig. 1. Round duration of LEACH-like protocols.

between logical clusters and physical topologies. Technical details of RANCE can be found in successive parts.

III. Mathematical Analysis and Problem Modeling

Previous works of LEACH-like protocols focus mainly on extending the lifetime of the network. For a MANET, nodes are required to collaborate to accomplish complex tasks. To achieve this, it is also desirable that nodes stay attached in some cluster long enough to offer data relay and various functions. We mathematically analyze key metrics of LEACH-like protocols in a mobile wireless and collaborative environments, and demonstrate step by step the inapplicability of simply applying LEACH-like protocols in such environments given their static round-based (i.e., fixed duration for each round) clustering workflows, using a simplified example as shown in Fig. 1. The duration of every clustering round is a fixed value Δu. Notations and definitions used in mathematical analysis hereinafter are given in Table II.

A. Clustered Time

Clustered time represents the duration one node stays attached to some cluster. We first deduct the mathematical representation of clustered time of a node k. Suppose node k is clustered with CH h. Let t denote time, and p denote the pth round. Let $x_k(p)$ and $y_k(p)$ denote the x-y coordinates at the beginning of round p, and $s_k(t)$ and $\theta_k(t)$ denote the time-varying speed and angle of node k, respectively. The coordinates for round $p+1$ can be acquired using the following equations:

$$\begin{aligned} x_k(p+1) &= x_k(t_p+\Delta u) \\ &= \int_{t_p}^{t_p+\Delta u} s_k(t)\cos\theta_k(t)dt + x_k(p) \qquad (2) \\ y_k(p+1) &= y_k(t_p+\Delta u) \\ &= \int_{t_p}^{t_p+\Delta u} s_k(t)\sin\theta_k(t)dt + y_k(p). \qquad (3) \end{aligned}$$

Therefore, the distance $d_{k,h}$ between node k and CH h for the beginning of round p can be calculated through the following equation. Taking (2)–(3) into 4, $d_{k,h}$ can be obtained in a recursive manner given deterministic $x_k(0)$, $y_k(0)$, $s_k(t)$, and $\theta_k(t)$

$$d_{k,h}(p) = \sqrt{(x_k(p)-x_h(p))^2 + (y_k(p)-y_h(p))^2}. \qquad (4)$$

TABLE II
Notations and Definitions for Adaptive Intervals

Notations	Definitions
k	The k-th node among all nodes
h	Usually refers to a CH unless specified otherwise
K	The total number of nodes
H	The total number of CHs
P	The total number of rounds for a node k
Q	The total number of ticks in a round p
$t_{p,q}$	The time corresponding to the q-th tick of round p
t_p	The time corresponding to the beginning of round p
Δu	The duration of very round
Δt	The duration after every tick inside a round
$x_k(t_{p,q})$	The x-axis coordinate of node k at time $t_{p,q}$
$y_k(t_{p,q})$	The y-axis coordinate of node k at time $t_{p,q}$
$x_k(p)$	The x-axis coordinate of node k at the beginning of round p
$y_k(p)$	The y-axis coordinate of node k at the beginning of round p
X	The upper bound of x-axis coordinate
Y	The upper bound of y-axis coordinate
$s_k(t)$	The time-varying speed of node k
$\theta_k(t)$	The time-varying angle of node k
$d_{k,h}(t_{p,q})$	The distance between node k and h at time $t_{p,q}$
$d_{k,h}(p)$	The distance between node k and h at the beginning of round p
D_k	The radius of radio coverage of node k
$c_{k,h}(p)$	The 0-1 variable indicating whether node k is inside the radio coverage of node h at the beginning of round p
Δct_k	The clustered time of node k
ct_{av}	The average of clustered time of all nodes
ctp_{av}	The average percentage of clustered time of all nodes
$V_h(t_{p,q})$	The cluster relationships stored at CH h at time $t_{p,q}$
$V_h^{un}(t_{p,q})$	The un-consistent (i.e., false) cluster relationships stored at CH h at time $t_{p,q}$
$V_h^{con}(t_{p,q})$	The consistent (i.e., true) cluster relationships stored at CH h at time $t_{p,q}$, i.e., $V_h^{con}(t_{p,q}) = V_h(t_{p,q}) - V_h^{un}(t_{p,q})$
$con_h(t_{p,q})$	The consistency of CH h's cluster relationships at time $t_{p,q}$
con_h	The overall consistency throughout the lifetime of CH h
con_{av}	The average consistency of all CHs
E_{extra_av}	The average extra energy consumption
Δt_{min}	The minimum aliveness probing interval
Δt_{rng}	The variable range of aliveness probing interval
Δt_{adapt}	The adaptive aliveness probing interval

Let $c_k(p)$ denote a 0-1 variable indicating whether node k is inside the radio coverage D_h of CH h, as the following:

$$c_{k,h}(p) = \begin{cases} 1, & d_{k,h}(p) < D_h \\ 0, & d_{k,h}(p) \geq D_h. \end{cases} \qquad (5)$$

The clustered time of node k can be calculated using (6), because once a node k is inside the radio coverage of some CH h at the beginning of some round p (that is when clustering occurs), node k receives CH advertisements, and is able to participate in the clustering process thus getting attached. During the whole duration Δu, node k is considered to be clustered with CH h due to the fixed round-based strategy adopted by LEACH-like protocols

$$\Delta ct_k = \sum_{p=0}^{P} c_{k,h}(p)\Delta u. \qquad (6)$$

The average clustered time ct_{av} and its percentage ctp_{av} in a MANET are shown as follows, where K denotes the total number of nodes, and $life_{av}$ denotes the average lifetime of nodes

$$ct_{av} = \overline{\sum_{k=1}^{K} \Delta ct_k}$$

$$ctp_{av} = \frac{ct_{av}}{\text{life}_{av}}. \tag{7}$$

Assume CHs are perfectly evenly distributed for each round, if all nodes are static or moving to the same direction with the same speed, i.e., $s_k(t) = s_l(t)$ and $\theta_k(t) = \theta_l(t)$ for any nodes $k, l \in \{1, \ldots, K\}$, $k \neq l$, it holds true that $d_{k,l}(p) = d_{k,l}(p+1)$ according to (2)–(7), thus very long clustered time once they are initially clustered. Therefore, the primary factor that affects the clustered time is the initial distribution (i.e., positions) of nodes, and LEACH-like protocols will work just fine in such "static" scenarios. However, for an ad hoc and randomly mobile wireless network, $s_k(t)$ and $\theta_k(t)$ where $k \in \{1, \ldots, K\}$ are very different from node to node, and from time to time, thus higher probability for $c_k(p)$ to be assigned 0, leading to shorter clustered time when simply applying LEACH-like protocols.

B. Consistency

Another thing worth noticing in a mobile wireless environment is that a node k might move out of the radio coverage of CH h and back inside later during Δu, i.e., physically detached for a short period of time Δt while seemingly clustered throughout the duration Δu of a round, where $\Delta t < \Delta u$ as shown in Fig. 1. CH h might not be aware of this short-term physical detach since LEACH-like protocols work on the basis of fixed round duration Δu. Therefore, there is probability that the clustered status discovered by CH h might be different with the real underlying physical topology during Δt, thus the inconsistency. If data are forwarded by CH to inconsistent cluster members during Δt, packet delivery fails, harmful to continuous streaming-styled services that require stable wireless links.

Consistency represents the consistent states between the clustered relationships discovered by the CH and physical topologies. We deduct the mathematical representation of consistency. Suppose consistency can be checked every Δt seconds (which, however, is usually missing in LEACH-like protocols), and each Δu contains Q Δts, i.e., the following:

$$Q = \frac{\Delta u}{\Delta t}. \tag{8}$$

Let p denote the pth round and q denote the qth tick (i.e., the beginning of every Δt) inside the round. We have the following relationships:

$$t_{p,q} = t_{p,q-1} + \Delta t. \tag{9}$$

The coordinates of node k at time $t_{p,q}$ (i.e., the qth tick of the pth round) can be acquired using the following equations:

$$\begin{aligned} x_k(t_{p,q}) &= x_k(t_{p,q-1} + \Delta t) \\ &= \int_{t_{p,q-1}}^{t_{p,q-1}+\Delta t} s_k(t)\cos\theta_k(t)dt + x_k(t_{p,q-1}) \end{aligned} \tag{10}$$

$$\begin{aligned} y_k(t_{p,q}) &= y_k(t_{p,q-1} + \Delta t) \\ &= \int_{t_{p,q-1}}^{t_{p,q-1}+\Delta t} s_k(t)\sin\theta_k(t)dt + y_k(t_{p,q-1}). \end{aligned} \tag{11}$$

Inconsistency occurs when the following condition holds true during Δt, i.e., a cluster member moves out of the CH's radio coverage before the next round starts

$$\begin{aligned} d_{k,h}(t_{p,q}) &= \sqrt{\left(x_k(t_{p,q}) - x_h(t_{p,q})\right)^2 + \left(y_k(t_{p,q}) - y_h(t_{p,q})\right)^2} \\ &\geq D_h, \text{ where } 0 \leq q \leq Q. \end{aligned} \tag{12}$$

Let $V_h^{un}(t_{p,q})$ denote the set for any cluster member k of CH h whose $d_{k,h}(t_{p,q}) \geq D_h$, and $V_h(t_{p,q})$ denote the set of all cluster members of CH h at time $t_{p,q}$. The instantaneous consistency of CH h at time $t_{p,q}$ can be acquired by the following:

$$\text{con}_h(t_{p,q}) = 1 - \frac{\left\|V_h^{un}(t_{p,q})\right\|}{\left\|V_h(t_{p,q})\right\|}. \tag{13}$$

The overall consistency throughout the lifetime of CH h can be obtained by the following:

$$\text{con}_h = \overline{\sum_{p=0}^{P} \overline{\sum_{q=0}^{Q} \text{con}_h(t_{p,q})}}. \tag{14}$$

The average consistency in a MANET is as follows, where H denotes the total number of nodes that have been CHs:

$$\text{con}_{av} = \overline{\sum_{h=1}^{H} \text{con}_h}. \tag{15}$$

Despite of its importance, consistency checking is usually missing in LEACH-like protocols. That is why the cluster relationships in a mobile ad hoc environment can be inaccurate.

C. Extra Energy Consumption

To improve consistency between cluster relationships and physical topologies for LEACH-like protocols, intuitively, introducing periodic consistency check with smaller Δt seems promising, because inconsistent cluster relationships can be removed in time once discovered. Nevertheless, it hardly captures the wireless dynamics in MANETs, and incurs higher energy consumption caused by more packet exchanges. Now, we analyze energy consumption. We assume a simplified consistency check model that works in round-trip manner: the CH multicasts a request packet for the consistency check, and cluster members unicast a reply upon reception. Any failed reply within Δt implies that (12) holds, thus the inconsistency. The radio model we adopted for energy consumption is similar to that of [32]

$$\begin{aligned} E_{tx}(l, d) &= \begin{cases} lE_{\text{elec}} + l\epsilon_{fs}d^2, d < d_0 \\ lE_{\text{elec}} + l\epsilon_{mp}d^4, d \geq d_0 \end{cases} \\ E_{rx}(l) &= lE_{\text{elec}}. \end{aligned} \tag{16}$$

In (16), d is the distance between sender and receiver nodes, $E_{\text{elec}} = 50$ nJ/bit is the energy dissipated per bit to run the transceiver circuitry, l is the length of packet by bit, whereas $\epsilon_{fs} = 10$ pJ/bit/m^2 and $\epsilon_{mp} = 0.0013$ pJ/bit/m^4 are amplifier energy parameters corresponding to the free space channel model and the multipath channel model, respectively. During round p with duration Δu, if consistency check is to be executed every Δt, the energy consumed by CH h can be

calculated by (18) where $E_{tx}(l, D_h)$ represents the energy consumption of the multicast by the CH, and $\|V_h^{\text{con}}(t_{p,q})\|E_{rx}(l)$ represents the CH's energy consumption of receiving of packets returned by cluster members. The energy consumed by all cluster members of the cluster can be obtained by (19), where $E_{tx}(l, d_{k,h})$ represents the energy consumption of the unicast responded to the CH by every member, and $\|V_h^{\text{con}}(t_{p,q})\|E_{rx}(l)$ represents cluster members' energy consumption of receiving of multicast issued by the CH. Together, the extra energy consumption by simple consistency check during a round can be obtained through (20) as follows:

$$V_h^{\text{con}}(t_{p,q}) = V_h(t_{p,q}) - V_h^{un}(t_{p,q}) \tag{17}$$

$$E_h(p) \geq \sum_{q=0}^{Q}\left[E_{tx}(l, D_h) + \left\|V_h^{\text{con}}(t_{p,q})\right\|E_{rx}(l)\right] \tag{18}$$

$$E_{\text{mem}}(p) \geq \sum_{q=0}^{Q}\left[\sum_{k=1}^{\|V_h^{\text{con}}(t_{p,q})\|} E_{tx}(l, d_{k,h}) + \left\|V_h^{\text{con}}(t_{p,q})\right\|E_{rx}(l)\right] \tag{19}$$

$$E_{\text{extra}}(p) = E_h(p) + E_{\text{mem}}(p). \tag{20}$$

As seen from (20), simply introducing static periodic consistency check to LEACH-like protocols incurs extra energy consumption to enhance consistency of plain LEACH-like protocols for a cluster during a round. In a mobile ad hoc environment, distances between nodes can be large, especially for centralized clustering such as LEACH-C, thus energy consuming if the consistency checking is not deliberately designed. The average extra energy consumption for every node throughout its lifetime can be obtained by the following:

$$E_{extra\_av} = \frac{\sum_{p=0}^{P} E_{\text{extra}}(p)}{K}. \tag{21}$$

D. Problem Modeling

Based on the previous mathematical analysis, simply introducing frequent checking every Δt where $\Delta t < \Delta u$ can hardly improve clustered time and consistency while still being energy efficient. Therefore, a dedicated protocol should be designed considering the requirements of clustering and the dynamics of MANETs.

The key problem is twofold, prolonging the clustered time at minimum energy cost to sustain continuous services, and offering a fine-grained maintenance mechanism to provide high consistent cluster relationships, which can be modeled as a mathematical programming problem as follows:

$$\max \quad ct_{av} \text{ and } \text{con}_{av} \tag{22}$$

$$\min \quad E_{extra\_av} \tag{23}$$

$$\text{s.t.} \quad c_{k,h}(p) \in \{0, 1\}, \qquad h \neq k, h, k \in \{1, \ldots, K\} \tag{24}$$

$$\sum_{h=1}^{H} c_{k,h}(p) \leq 1, \qquad h \neq k, h, k \in \{1, \ldots, K\} \tag{25}$$

$$\exists t, s_k(t) > 0, \qquad k \in \{1, \ldots, K\} \tag{26}$$

$$\exists t, \exists \epsilon, \theta_k(t) \neq \theta_k(t + \epsilon), |\epsilon > 0|, k \in \{1, \ldots, K\} \tag{27}$$

$$0 < x_k(t) < X, \qquad k \in \{1, \ldots, K\} \tag{28}$$

$$0 < y_k(t) < Y, \qquad k \in \{1, \ldots, K\} \tag{29}$$

$$0 < D_k \ll \min(X, Y), \qquad k \in \{1, \ldots, K\}. \tag{30}$$

Conditions (24) and (25) indicate that a node can be clustered with at most one CH for every round. Conditions (26) and (27) emphasize the network mobility with changing speed and angle, and conditions (28) and (29) indicate the deployed dimensions of the MANET. Condition (30) indicates a low-energy network that nodes can only communicate with nearby peers locally. In order to solve this problem, RANCE, a randomly centralized and on-demand clustering protocol, is proposed.

TABLE III
TYPES OF PDUS AND ROLES

PDUs/Roles	Definitions
cluster_build	PDU that initiates the clustering
cluster_join	PDU that requests to join a cluster under construction
cluster_role	PDU that assigns the role to a node in the cluster under construction
cluster_confirm	PDU that confirms the assigned role as member in the cluster under construction
cluster_ack	PDU that finally acknowledges other nodes' joining to cluster
cluster_alive_req	PDU issued by the head for aliveness maintenance
cluster_alive_resp	PDU replied by members for aliveness maintenance
cluster_detach	PDU issued by the head to notify the destruction of the cluster
cluster_hitch	PDU issued by detached nodes to swiftly join a readily constructed cluster
unspec	Unspecified role
peer	Role for nodes that are not currently part of any existing cluster
initiator	Role for the node that initiates the clustering
member	Role for nodes that are under control of the head
head	Role for the node that is selected as the controller of the cluster

IV. RANCE CLUSTERING PROTOCOL

RANCE can be divided into four phases (see Sections IV-B–IV-E), namely, clustering, maintaining, hitchhiking, and rebuilding, among which the latter two can be categorized as subphases of the maintaining phase. We first introduce RANCE's protocol data unit (PDU) and configurations.

A. PDU and Configurations

RANCE is designed as an application protocol on top of UDP. Without losing generality, RANCE can also be implemented as a link-layer protocol in real-world implementations. It consists of several differently typed PDUs, including *cluster_build*, *cluster_join*, *cluster_role*, *cluster_confirm*, *cluster_ack*, etc. There are several different roles during the clustering process, namely, *peer*, *head*, *member*, *initiator*, etc. Explanations of PDUs and roles are shown in Table III.

The structure of PDUs is shown in Fig. 2. The *type* field indicates the PDU type among *cluster_build*, *cluster_join*,

TABLE IV
EXAMPLE OF FITNESS EVALUATION

	weight (wgt_i)	bench ($bench_i$)	unit	tendency ($sign_i$)	N1	N2	N3	N4
fitness, f					0.227	0.157	0.479	0.389
residual RAM, v_1	0.1	10000	MB	1	1000	51	5000	3000
occupied CPU, v_2	0.1	100	%	-1	80	80	50	60
residual battery, v_3	0.6	10000	J	1	5000	4000	8000	7000
residual bandwidth, v_4	0.1	10000000	Kbps	1	1000	999	2000	1000
delay, v_5	0.05	10	s	-1	0.005	0.005	0.005	0.006
packet loss, v_6	0.05	100	%	-1	5	5.1	0.025	0.05

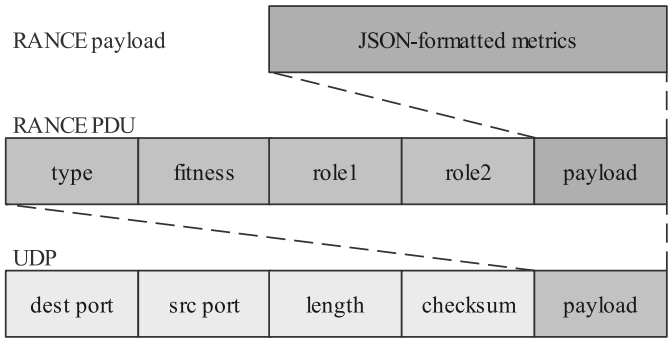


Fig. 2. RANCE clustering protocol PDU structure.

cluster_role, *cluster_confirm*, *cluster_ack*, etc. The *role1* indicates the self role of the PDU sender whereas *role2* indicates the role to be assigned to the PDU receiver in some cases. If no role is assigned, role2 is set as unspecified (i.e., unspec). The fitness in the PDU indicates how suitable a node is to be selected as CH. Therefore, the initiator must collect fitness values of nodes in advance before the CH selection. The acquisition of fitness values works in two ways: 1) the sender of the PDU can leave the fitness field blank, and encapsulate JSON-formatted metrics (if implemented as an application protocol) in the payload so that the initiator can calculate the fitness value in a centralized manner or 2) the sender can directly calculate the fitness value locally, fill it in the fitness field, and send to the initiator. Either way, the fitness value can be determined by aggregating important metrics (such as residual energy, CPU, RAM, bandwidth, etc.), together with their tendencies, i.e., positive (the larger values the better) or negative (the smaller values the better), as shown in 31, where i represents the ith metric of a node, sign_i represents its tendency, wgt_i represents its weight, v_i represents its real value, and bench_i represents the preconfigured benchmark value for the ith metric

$$\text{sign} = \begin{cases} 1, & \text{if positive} \\ -1, & \text{if negative} \end{cases}$$
$$f = \sum_{i=1}^{n} \text{sign}_i \times wgt_i \times \frac{v_i}{\text{bench}_i}, \quad \text{where} \quad \sum_{i=1}^{n} wgt_i = 1. \quad (31)$$

Table IV explains how fitness values can be evaluated considering metrics v_1–v_6, i.e., residual RAM, occupied CPU, residual battery, residual bandwidth, delay, and packet loss of several nodes (N1–N4). Obviously, occupied CPU and delay are metrics with negative tendencies while the others are positive ones. In the example shown in Table IV, residual battery (v_3) has the highest weight ($wgt_3 = 0.6$), which means it imposes the greatest impact for fitness evaluation. By applying (31), fitness values for nodes N1–N4 are 0.227, 0.157, 0.479, and 0.389, respectively, given metric values, weights, and bench values in Table IV. Therefore, when selecting CH, the initiator that collects fitness values is most likely to pick N3 as CH. It is reasonable to value residual battery the most in battery-critical missions (e.g., long-time periodical environmental monitoring), by assigning weight as high as 0.6 in Table IV. Other metrics and weights can also be considered for various applications, depending on their natures. For example, time-critical missions (e.g., battlefields monitoring) would favor delay by assigning higher weight to it, and bandwidth-critical missions (e.g., live video surveillance) would favor residual bandwidth instead. Indeed, weights assignments are quite application dependent.

B. Clustering

1) Initiation of Clustering: Every mobile node works in a peer-to-peer manner if no clustering process takes place, thus called *peers*, i.e., the initial role for every mobile node. Peers are configured to automatically join a multicast group (e.g., 224.0.0.1) and are all able to initiate the clustering process by multicasting the cluster_build PDU. In order to reduce the chance of conflicts with other cluster_build initiations, peers enter the WAIT_FOR_BUILD state to passively allow for clustering led by others. When the WAIT_FOR_BUILD state times out after B seconds (which means no other peer initiates clustering), a peer is free to send cluster_build, actively leading the clustering process, and enters the WAIT_FOR_JOIN state with a J-second timer. The reason why WAIT_FOR_JOIN state lasts for J seconds is to allow for enough PDU round-trip time. Meanwhile the role of this peer has now been changed to *initiator*. Note that an initiator is not a CH by default. A selection for CH will take place later. Other peers receiving cluster_build send back cluster_joins, and enter the WAIT_FOR_ROLE state whose timer lasts for R seconds. If no role is assigned within R seconds, a peer goes back to WAIT_FOR_BUILD state. The initiator adds the sender peer of cluster_join in the cand_list (candidate list), a key-value store whose keys are the IP addresses and values are corresponding node details.

2) CH Selection and Role Assignment: When the WAIT_FOR_JOIN state times out, the initiator starts the CH

selection among peers stored in the cand_list, based on the fitness either contained in cluster_join or calculated based on metrics encapsulated in the cluster_join payload. When the to-be CH is selected, the initiator notifies it with the cluster_role PDU wherein the role2 field is set as *head*, and then the peer sets its role as *head* accordingly upon PDU reception and becomes the CH. CH then multicasts the cluster_role PDU to all other peers with the role1 field as head (indicating itself as role the head) and role2 field as *member* (notifying participant peers of the role as members for themselves).

3) Cluster Formation: The CH enters the WAIT_FOR_CONFIRM state with a C-second timer. Peers reply cluster_confirm PDUs if they accept the role as members and enter the WAIT_FOR_ACK state with a A-second timer. If the A-second timer timeouts, it indicates that the CH does not reply the cluster_ack PDU in time, possibly because the participant peer and the CH mutually move out of radio range due to mobility, or node failures, etc., in which case, peers reenter the WAIT_FOR_BUILD state to start over the clustering process from scratch. If the CH replies cluster_ack in time before WAIT_FOR_ACK times out, the participant peer can safely assume the cluster is now constructed, enters the BUILT state, and becomes a controlled member of the cluster. When the WAIT_FOR_CONFIRM state times out, the CH stores those nodes who replied cluster_confirms (indicating willingness to become members) in the cluster_rel (cluster relationships), also a key-value store as cand_list, except for that cluster_rel stores real members after cluster formation while cand_list stores temporary candidate members during clustering, and finally enters the BUILT state where cluster-based collaboration can happen.

The clustering procedure is depicted in Algorithm 1. Fig. 3 depicts the finite-state machines (FSMs) of the whole workflow of RANCE (including the clustering phase described in this section, and maintenance, hitchhiking, and rebuilding phases to be described later), wherein black arrows are the state transitions of the clustering procedure. Fig. 4(a) gives an example on how a cluster is constructed: 1) N1 initiates cluster_build; 2) N2–N4 reply with cluster_joins; 3) N1 selects N3 as the CH by comparing fitness values and notifies N3 the result; 4) N3 notifies all other nodes its role as the CH, and their roles as the cluster members; 5) all other nodes confirm to join the cluster; and 6) N3 acknowledges the cluster formation. To summarize, during the clustering process, initiation of clustering is competed randomly, and CHs are selected in a centralized manner only in the local wireless network, which reduces energy consumption caused by long-range BS-involved communication.

Algorithm 1 Clustering Procedure

```
state WAIT_FOR_BUILD(timeout=B):
role = peer; head = null
cluster_rel = cand_list = {self}
if receive cluster_build then
    reply cluster_join
    enter WAIT_FOR_ROLE(timeout=R)
if timeout then
    role = initiator
    multicast cluster_build
    enter WAIT_FOR_JOIN(timeout=J)

state WAIT_FOR_JOIN(timeout=J):
if receive cluster_join then
    add to cand_list
if timeout then
    if cand_list.size == 1 then
        enter WAIT_FOR_BUILD(timeout=B)
    else if cand_list.size > 1 then
        send cluster_role(role2=head) to cand_list.best
        enter WAIT_FOR_ROLE(timeout=R)

state WAIT_FOR_ROLE(timeout=R):
if receives cluster_role then
    role = role2 in cluster_role
    if role == head then
        multicast cluster_role(role1=head,role2=member)
        enter WAIT_FOR_CONFIRM(timeout=C)
    else if role2 == member then
        head = sender of cluster_role
        reply cluster_confirm
        enter WAIT_FOR_ACK(timeout=A)
if timeout then
    enter WAIT_FOR_BUILD(timeout=B)

state WAIT_FOR_CONFIRM(timeout=C):
if receive cluster_confirm then
    reply cluster_ack
    add to cluster_rel and set its aliveness al=3
    enter WAIT_FOR_CONFIRM(timeout=C)
if timeout then
    if cluster_rel.size == 1 then
        enter WAIT_FOR_BUILD(timeout=B)
    else if cluster_rel.size > 1 then
        enter BUILT()

state WAIT_FOR_ACK(timeout=A):
if receive cluster_ack then
    enter BUILT()
else if timeout then
    enter WAIT_FOR_BUILD(timeout=B)
```

C. Maintenance

1) Bidirectional Heartbeat Interactions: The CH and all cluster members are mobile nodes. There might be changes of link-layer connectivities due to mobility, wireless communication range, signal strength, obstacles, terrains, etc., thus the possibility of inconsistency between logical cluster relationships and physical connectivities, if no maintenance is provided after cluster formation. Therefore, a fine-grained maintenance mechanism should be designed to distinguish short-term and long-term connectivity changes, to properly restore or dissolve cluster relationships stored in cluster_rel, respectively, so as to keep cluster relationships and physical connectivities as consistent as possible.

Cluster relationships are evaluated by the multilevel aliveness and maintained through bidirectional heartbeat packets interactions in RANCE. The CH has details of every participant member stored in the cluster_rel key-value store while members do not have to know about other members, thus having only the CH's details inside the cluster_rel of their own. The cluster_rel has the following key-value structure,

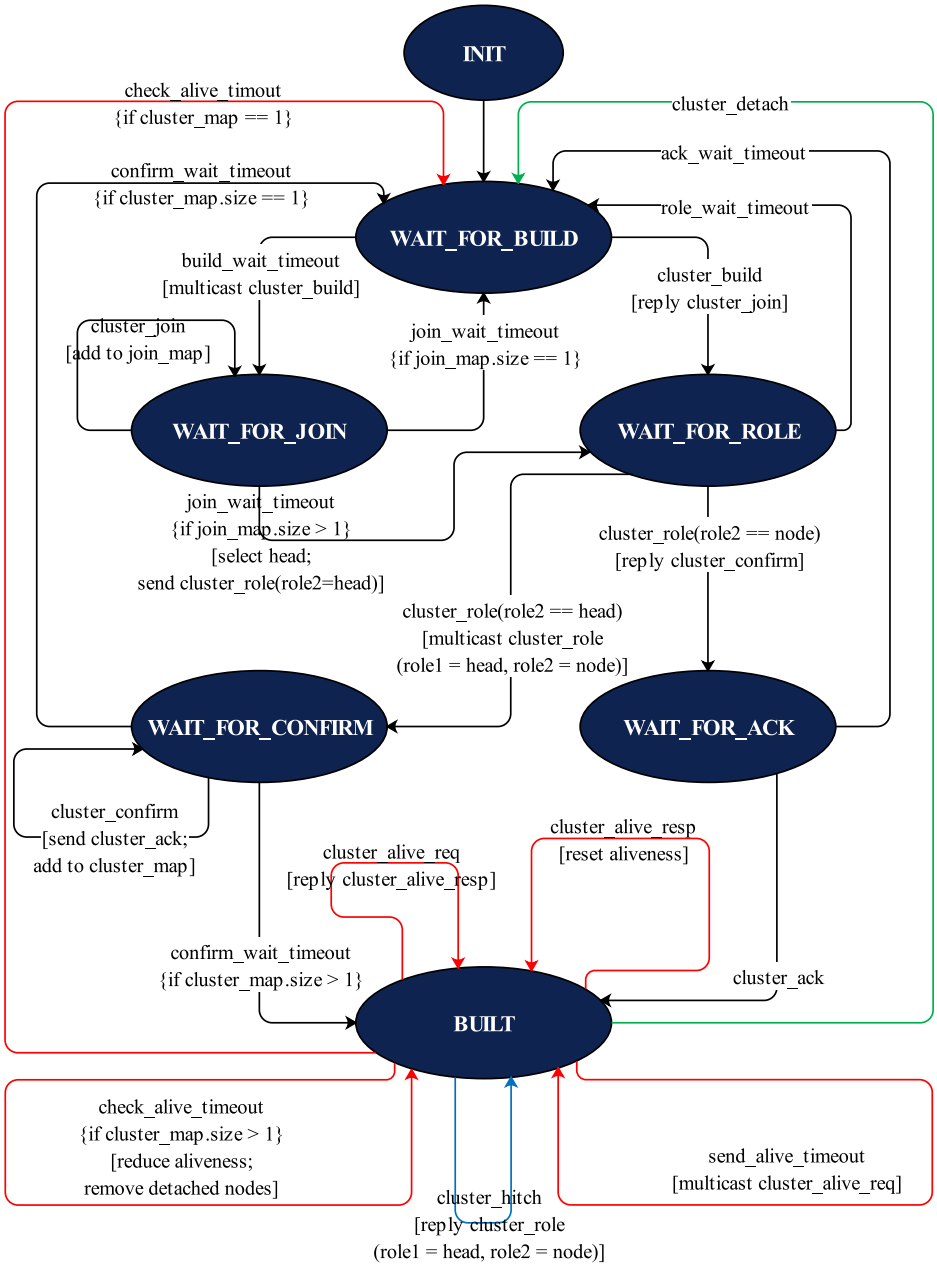


Fig. 3. FSMs of RANCE.

IP: {aliveness, fitness, x-coordinate, y-coordinate, ...}. Every node has an initial aliveness value of $al_{\text{ini}} = 3$. The CH initiates heartbeat interactions by multicasting a cluster_alive_req PDU every Δt seconds, i.e., the heart_beat_timer. Members in the same cluster reply a cluster_alive_resp PDU to the CH upon reception of cluster_alive_req. The successful reception of cluster_alive_req/resp resets the aliveness of the corresponding node k back to its initial value in the cluster_rel, i.e., $al_k = al_{\text{ini}}$, indicating the mutual communication is still alive. Every node also checks whether cluster_alive_req/resp is received within Δt seconds, i.e., the check_alive_timer. Node k's aliveness decreases by 1 for every failed reception of cluster_alive_req/resp within every Δt seconds, i.e., $al_k = al_k - 1$. If the aliveness reaches 0, i.e., $al_k = 0$, the corresponding node k is considered to be detached (i.e., out of radio range, node failure, etc.), and will be deleted from the cluster_rel.

The maintenance procedure is described in detail in Algorithm 2. Red arrows in Fig. 3 depict the state transitions of the maintenance procedure. Fig. 4(b) exhibits a possible maintenance situation, and Table V shows the aliveness values of cluster members known by CH N3 over time. The dashed circle indicates the radio range of CH N3. N3 repeatedly multicasts cluster_alive_req (①, ④, and ⑦ in the figure) every Δt seconds. Member N4 temporarily moves out of the radio range (step ③), which leads to the unsuccessful reception of cluster_alive_req from and cluster_alive_resp back to N3 within Δt seconds. At this point of time, N4's aliveness value stored in N3's cluster_rel is reduced by 1 and *vice versa*

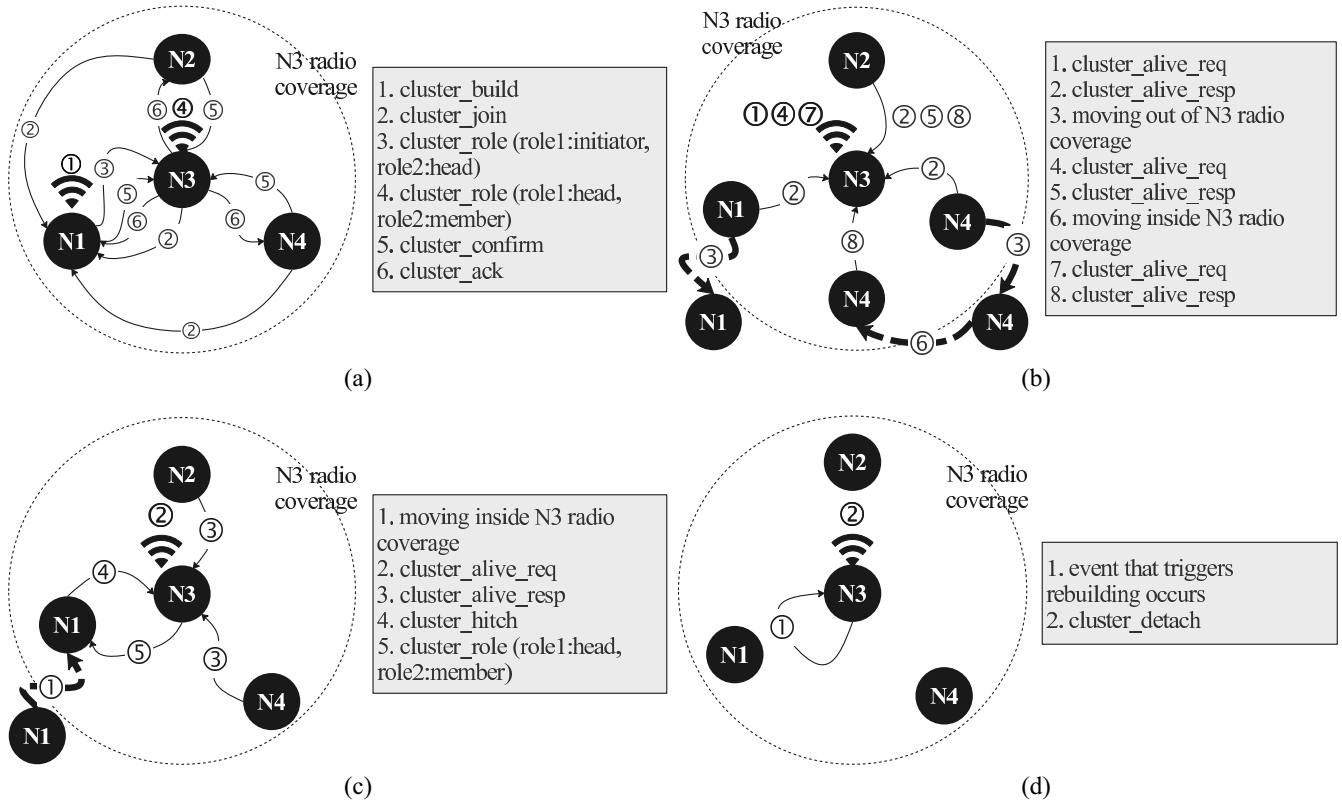


Fig. 4. Examples of the RANCE protocol. (a) Clustering. (b) Maintenance. (c) Hitchhiking. (d) Rebuilding.

Algorithm 2 Maintenance Procedure

```
state BUILT():
set heart_beat_timer(timeout=I) by Equation 32
set check_alive_timer(timeout=I) by Equation 32
if check_alive_timer times out then
    reduce all nodes' aliveness in cluster_rel by 1
    remove all nodes if aliveness al < 1 from cluster_rel
    if cluster_rel.size == 1 then
        multicast cluster_detach
        enter WAIT_FOR_BUILD(timeout=B)
    else if cluster_rel.size > 1 then
        set check_alive_timer(timeout=I) by Equation 32
if heart_beat_timer times out && role == head then
    multicast cluster_alive_req
    set heart_beat_timer(timeout=I) by Equation 32
if receives cluster_alive_req && role == member then
    reply cluster_alive_resp
    set al=3 in cluster_rel whose IP is the sender IP
if receives cluster_alive_resp && role == head then
    set al=3 in cluster_rel whose IP is the sender IP
if receives cluster_detach && role == member then
    enter WAIT_FOR_BUILD(timeout=B)
```

(see the "after ⑤" column in the table). However, it does not incur the immediate deletion of N4 from N3's cluster_rel (and *vice versa*) because mobile nodes do have some probability of moving back to the CH's radio range within Δt seconds, if the aliveness values does not reach to 0. The aliveness probing will attempt for at most al_{ini} times before the deletion. We can see that N4 moves back to the radio range (step ⑥) and sends back cluster_alive_resp upon N3's cluster_alive_req

TABLE V
ALIVENESS VALUES OF CLUSTER MEMBERS KNOWN BY CH N3

Node	before ②	after ②	after ⑤	after ⑧
N1	3	3	2	1
N2	3	3	3	3
N4	3	3	2	3

multicast (step ⑧), leading N3 and N4's aliveness values are both reset to al_{ini} (see the "after ⑧" column in the table). Nevertheless, N1 will be permanently removed from N3's cluster_rel since it never returns back to the N3's radio range before al_{ini} unsuccessful heartbeat interactions.

Indeed, the aliveness mechanism, instead of immediate removal of the out-of-range member from the cluster, introduces some tolerance to topology dynamics caused by mobility, etc., and prolongs the time of being clustered if the topology changes are just temporary, thus a more stable cluster structure and longer clustered time, which is critical for internodes cooperations in MANETs. It also keeps the cluster_rel consistent with physical topologies through repeated bidirectional interactions. However, these packet interactions incur extra energy consumption. We will try to balance the cluster relationship consistency and energy consumption by means of adaptive aliveness probing (Section IV-C2).

2) Adaptive Aliveness Probing: Note that the aliveness probing interval Δt can be set as a fixed value, meaning that the aliveness probing and checking occurs in a periodic manner. To better capture the mobility of nodes and the dynamics

of the cluster topology, we also designed a mechanism that can automatically calculate an adaptive aliveness probing interval based on the spacial distance between nodes. Intuitively, the larger the spacial distance, the more possible that a node moves out of the radio coverage of the CH, thus higher possibility of detaching. If the aliveness probing interval can be adaptively shortened based on this intuition, nodes more likely to get detached can be discovered as early as possible, so that topology alteration is known timely. On the contrary, if nodes stay close to each other, aliveness probing does not need to be executed as frequently. The adaptive aliveness probing interval is obtained by applying (32), a Euclidean distance-based decay function, at the CH. First, every spacial distance between member k in cluster_rel and the CH h itself is calculated, using x and y coordinates encapsulated in cluster_alive_resp sent by members. The maximum distance is selected and denoted as d. D is the bench spacial distance from the CH, usually the radio coverage. Since D is the furthest reach between the CH and members, any member beyond distance D will not be able to send cluster_alive_resp, thus $0 \leq d \leq D$, i.e., $-1 \leq (d - D/2/D/2) \leq 1 \rightarrow (1/e) \leq e^{-(d-D/2/D/2)} \leq e$. Actually, $D/2$ is the dividing line according to (32) in that if there is any member beyond this line, Δt_{adapt} (the adaptive interval) falls into a smaller range of $[(\Delta t_{rng}/e) + \Delta t_{\min}, \Delta t_{rng} + \Delta t_{\min})$ for more frequent aliveness probing, otherwise, into a larger range of $(\Delta t_{rng} + \Delta t_{\min}, e \cdot \Delta t_{rng} + \Delta t_{\min})$ for less frequent aliveness probing

$$d = \max_{k=1}^{N} \left(\sqrt{(x_k - x_h)^2 + (y_k - y_h)^2} \right)$$
$$\Delta t_{\text{adapt}} = \Delta t_{rng} \cdot e^{-\frac{d-D/2}{D/2}} + \Delta t_{\min}. \tag{32}$$

The reason why the interval is finally determined by decay function instead of just the Euclidean distance is that the nonlinear decay function varies much faster than the linear Euclidean distance, making the CH get ready for the worst scenario (i.e., possible detaching) in advance (i.e., short-term prediction). In this method, the aliveness probing interval automatically adapts to the topology. For more scattered topology, the interval becomes smaller to discovery possible member detaching, whereas the interval becomes bigger to lower energy consumption in a more dense topology where members are less likely to get detached from clusters.

Table VI gives an example of adaptive intervals, given the CH-member relationships in Fig. 4(b). The settings are as follows: $\Delta t_{rng} = 7.5$ s, $\Delta t_{\min} = 7.5$ s, and $D = 1500$ m. We assume that $\Delta t = \Delta t_{rng} + \Delta t_{\min} = 15$ s for nonadaptive intervals. For round 1 of aliveness probing [i.e., before ② in Fig. 4(b)], the CH N3 finds the furthest member N4 (554.6 m) and determines the aliveness probing interval should be set as $\Delta t_{\text{adapt}} = 17.2$ s, bigger than $\Delta t = 15$ s, thus less energy consumption. For round 2 (that is when the previous interval times out), N3 recalculates the interval as 14.7 s determined by the new furthest member N4 due to mobility. For round 4 ["after ⑤" in Fig. 4(b)], since N1 and N4 move out of the radio range of N3, their coordinates are not received through heartbeat interactions within an interval. However, because their aliveness values have not been reduced to 0 at this time, the CH N3 treats the distance to N1 or N3 as the furthest radio reach, i.e., $d = D = 1500$ m and $\Delta t_{\text{adapt}} = 10.3$ s, which results in more frequent aliveness probing to discover possible topology change. The same procedure applies for round 4 ("after ⑤").

TABLE VI
EXAMPLE OF ADAPTIVE INTERVALS

round	head	member	x_k	y_k	d	Δt
1	N3	N1	200.0	250.0	554.6	17.2
		N2	510.0	900.0		
			500.0	500.0		
		N4	1000.0	260.0		
2	N3	N1	100.0	200.0	779.0	14.7
		N2	515.0	895.0		
			520.0	530.0		
		N4	1200.0	150.0		
3	N3	N1			1500.0	10.3
		N2	505.0	910.0		
			520.0	530.0		
		N4				
4	N3	N1			1500.0	10.3
		N2	530.0	930.0		
			520.0	530.0		
		N4	600.0	300.0		

3) On-Demand Clustering: The fine-grained maintenance based on bidirectional heartbeat interactions brings another desired feature, the on-demand formation of clusters. Taking Fig. 4(b) as an example again, if N1 and N4 move out of CH N3's radio range and never return (i.e., detached), and happen to move inside each other's radio range afterward, they can start forming a cluster themselves automatically as per the clustering procedure described in Section IV-B, possibly with some other detached nodes within reach. Therefore, the formation of clusters works in an as-needed manner. The number of CHs is self-adaptive to the network topology, as many as needed, instead of a fixed percentage of CHs in LEACH-like protocols, i.e., p in (1), whose assignment is rather subjective and difficult. On-demand clustering further prolongs nodes clustered time to sustain continuous services or time-critical collaborations.

D. Hitchhiking

Note that the detached node N1 as depicted in Section IV-C will start its own clustering process from scratch by first entering the WAIT_FOR_BUILD state, as described in Algorithm 1. The start over of the clustering has some limitations in that it has to experience a whole clustering process requiring more time including some mandatory timeouts (*J*-timer, *C*-timer, etc.), and more energy due to several round-trip packet interactions. To further improve energy efficiency, we designed the hitchhiking mechanism. Any detached node that overhears the cluster_alive_req indicating an alive cluster can apply to join it. It is called hitchhiking since the node does not have to undergo the whole clustering process as described in Section IV-B. The hitchhiking has two constraints: 1) the node and the overheard head must both enable hitchhiking configuration and 2) the detached node must have the peer role (i.e., in the WAIT_FOR_BUILD state) other than the initiator role (i.e., in the WAIT_FOR_JOIN state) because initiator

Algorithm 3 Hitchhiking Procedure

```
state WAIT_FOR_BUILD(timeout=B):
if receives cluster_alive_req & role == peer & hitch == true then
    HITCH()

state BUILT():
if receives cluster_hitch & role == head & hitch == true then
    ACCEPT_HITCH()

function HITCH():
send cluster_hitch to sender of cluster_alive_req
if received cluster_role then
    role = role2 in cluster_role, i.e., "member"
    add to cluster_rel and set its aliveness al=3
    head = sender of cluster_role
    enter BUILT()

function ACCEPT_HITCH():
send cluster_role(role2=member) to sender of cluster_hitch
add to cluster_rel and set its aliveness al=3
```

might already have some correlation with other nodes, and must not "abandon" these nodes in the halfway, to avoid state chaos and unexpected exceptions.

The hitchhiking procedure is described in detail in Algorithm 3. Blue arrows in Fig. 3 depict the state transitions of the hitchhiking procedure. As we can see from Fig. 4(c), the detached node N1 moves inside N3's radio range and overhears the cluster_alive_req sent by N3, if N1 just happens to switch to WAIT_FOR_BUILD state. It sends the cluster_hitch PDU to N3 that later replies with a cluster_role with *role1* being set as *head* and *role2* as *member*. N1 accepts the role and becomes part of the cluster if hitchhiking is turned on. Then, N1 and N3 start normal cluster_alive_req/resp heartbeat interactions as described in Section IV-B for maintenance.

E. Rebuilding

Unlike LEACH-like protocols that hand off the role of CH by rounds, our protocol tends to keep the cluster as steady as possible, to prolong the clustered time and established collaboration; that is to keep the CH handovers as minimum as possible since it requires extra time and energy to rebuild the cluster that could compromise the continuity of CH-member collaboration or real-time services (e.g., video surveillance streaming). Therefore, unless it is necessary, CH of the cluster is not to be reselected (i.e., rebuilding of the cluster). The necessity of rebuilding is when the current CH is not capable of or no longer suitable to coordinate the cluster, for example, low fitness value caused by low battery, etc. To this regard, the CH examines the fitness values of all members stored in the cluster_rel once every z seconds. If its fitness value is lower than p% (e.g., 70%) of the current highest one, the CH multicasts a cluster_detach PDU to notify all controlled members that a rebuilding of the cluster is ready to be initiated. Members go to WAIT_FOR_BUILD upon reception so that a new clustering can be undergoing. Note that since the current CH does not have the highest fitness value, it will not be selected as the new CH to avoid the suboptimal CH selection.

F. Protocol Analysis

We now analyze how RANCE behaves under various effects.

1) Effect of Node Additions and Removals on CH Selection: The R-second timer for WAIT_FOR_ROLE state provides remarkable robustness for CH selection process in a topology with frequent node additions or removals. Due to the mobility of nodes, there might be some inconsistency between the peers stored in the initiator's cand_list and those in the physical topology. We first analyze the node removal scenario. If the initiator N_{ini} selects peer N_x in its cand_list as CH, it notifies N_x with cluster_role PDU and enters WAIT_FOR_ROLE with an R-second timer (see Section IV-B1). Nevertheless, during the role assignment, N_x might be out of N_{ini}'s radio range due to mobility, failure, etc., which N_{ini} is unaware of at that time. This might lead to some inconsistency. Fortunately, N_x is also in WAIT_FOR_ROLE as is N_{ini}. If N_x does not receive cluster_role within R seconds, it goes back to WAIT_FOR_BUILD, and competes for initiation of clustering from scratch after the B-second timer times out, as described in Section IV-B1, and so does N_{ini}. Therefore, random node removals do not cause critical impact on CH selection regardless of inconsistency. We then analyze a node addition scenario. A peer node N_y that randomly joins the physical topology is unknown by the initiator N_{ini} during CH selection, because N_y does not participate in the initiation of clustering led by N_{ini} in the first place (see Section IV-B1). N_y will not be selected as CH even if it has the highest fitness value. However, N_y is now in its WAIT_FOR_BUILD state, which means it can lead its own or follow another node's initiation of clustering, starting over the clustering process, or N_y can possibly hitchhike an overheard cluster during its WAIT_FOR_BUILD period (see Section IV-D). Therefore, random node additions do not compromise current CH selections and role assignments. Which cluster the newly added node will belong to is quite random. In a word, the proposed CH selection and role assignment procedure, together with the timeout mechanism, have basic tolerance to node removals and additions (see details in simulations in Section VI).

2) Effect on Routing After Clustering: In MANETs, nodes are peers, i.e., end systems as well as routing devices. It is required that every peer should maintain routing information in advance if table-driven routing protocols (e.g., DSDV, destination sequenced distance vector) are adopted, or find routes in an ad hoc manner if on-demand routing protocols (e.g., AODV, ad hoc on-demand distance vector routing) are adopted. Either way, peers have to conduct routing and data forwarding themselves. Once nodes are clustered with some CH, CH acts as the "gateway" of the cluster. All controlled members in a cluster rely on CH for both inner and intercluster data forwarding. Therefore, theoretically speaking, members store only one default route that takes CH as the gateway, and forward all traffic to it. That is to say, it is single-hop data routing inside a cluster with CH as the "hub," obviously. On the other hand, CHs across the physical topology must conduct multihop data forwarding, either to communicate with each other, or to aggregate data to BS for remote process. Therefore, CHs adopts multihop routing to address each other, much like ad

hoc routing schemes [e.g., DSDV, AODV, dynamic source routing (DSR)]. To summary, RANCE provides a hybrid routing scheme after clustering—single-hop routing inside a cluster and multihop routing across CHs. Routing scheme turns from a flattened structure before clustering to a hierarchical structure after clustering.

3) Scalability of RANCE: Suppose the radius of the radio coverage of a node k is D_k, and the theoretical radio coverage of a CH h can be acquired by

$$\text{cov}_h = \pi D_h^2. \tag{33}$$

The total area of the field for the network to be deployed is determined by (34), where X and Y denote the width and length, respectively

$$\text{area} = X \cdot Y. \tag{34}$$

Suppose, initially (i.e., at time t_0), nodes are evenly distributed across the network. Therefore, the number of CHs that are able to cluster all nodes is determined by

$$H = \frac{\text{area}}{\text{cov}_h} = \frac{X \cdot Y}{\pi D_h^2}. \tag{35}$$

The average size of a cluster can be acquired by (36), given that the total number of nodes is K

$$\|V_h(t_0)\| = \frac{K}{H} = \frac{\pi D_h^2}{X \cdot Y} \cdot K. \tag{36}$$

The nodes in a cluster are moving all the time from t_0 to t_1. We calculate the average distance between members and CH at time t_1. Note that $d_{k,h}(t_1)$ can be acquired by (4)

$$d_{av}(t_1) = \frac{\sum_{k=1}^{\|V_h\|} d_{k,h}(t_1)}{\|V_h\|}. \tag{37}$$

If $d_{k,h}(t_1) < D_h$ holds true, it stochastically indicates members tend to stay in the current cluster, and the size of the cluster remains unchanged, if there do not exist node additions or removals at this time. If $d_{k,h}(t_1) \geq D_h$ holds true, the size of the cluster is proportional to $\frac{D_h}{d_{av}}$, as revealed by

$$\|V_h(t_1)\| = \begin{cases} \frac{\pi D_h^2}{X \cdot Y} \cdot K = \|V_h(t_0)\|, & \text{if } d_{k,h}(t_1) < D_h \\ \frac{\pi D_h^3}{X \cdot Y \cdot d_{av}} \cdot K = \frac{D_h}{d_{av}} \cdot \|V_h(t_0)\|, & \text{if } d_{k,h}(t_1) \geq D_h. \end{cases} \tag{38}$$

For situation that $d_{k,h}(t_1) < D_h$, all nodes still belong to some cluster at time t_1, i.e., the following equation:

$$\|V_h(t_1)\| \times H = \frac{\pi D_h^2}{X \cdot Y} \cdot K \times \frac{X \cdot Y}{\pi D_h^2} = K. \tag{39}$$

Otherwise, some nodes are dynamically detached from CHs at time t_1, i.e., the following equation:

$$\|V_h(t_1)\| \times H = \frac{\pi D_h^3}{X \cdot Y \cdot d_{av}} \cdot K \times \frac{X \cdot Y}{\pi D_h^2} = K \cdot \frac{D_h}{d_{av}} < K. \tag{40}$$

However, detaching can be swiftly detected by RANCE's fine-grained maintenance mechanism, and the on-demand clustering or hitchhiking can be initiated as described in Section IV-B. Therefore, it is believed that RANCE clusters a very high percentage of all nodes, and provides good clustering scalability to the network size (see later in Section VI-D for simulation details).

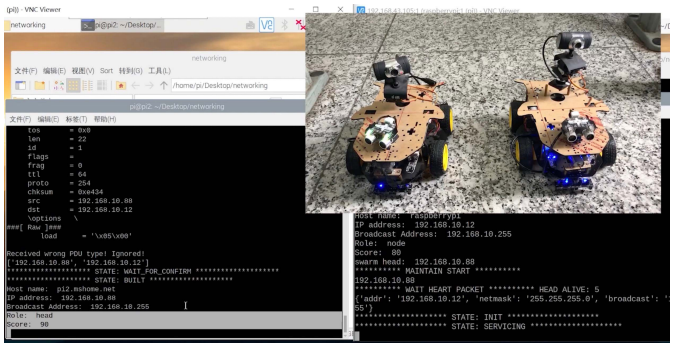
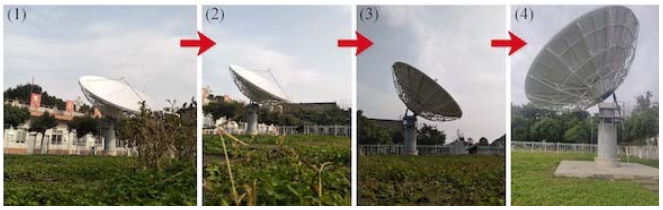


Fig. 5. Preliminary hardware prototype of RANCE in previous works.

(a) (b)



(c)

Fig. 6. Mobile surveillance of a meteorological radar. (a) Cluster and the radar. (b) Bird recognition. (c) Mobile surveillance around the radar.

V. Possible Applications

A. Mobile Surveillance of Unattended Assets

Some of RANCE's basic functions (such as clustering) have been preliminarily implemented on the Raspberry-based platform to prototype the clustering protocol in hardware environment in our previous works [33], [34], as shown in Fig. 5.

We believe these prototypes can be deployed for collaborative mobile surveillance over unattended facilities or assets. For example, in our previous work [33], several wheeled mobile nodes running early version of RANCE were deployed in a meteorological site to monitor harmful animals that would damage meteorological facilities. These nodes were clustered to provide a full-angle, mobile, and collaborative surveillance over meteorological facilities (e.g., radars) to reduce the chance of blind spots [see Fig. 6(c)]. Some nodes were equipped with image recognition functionalities, and others

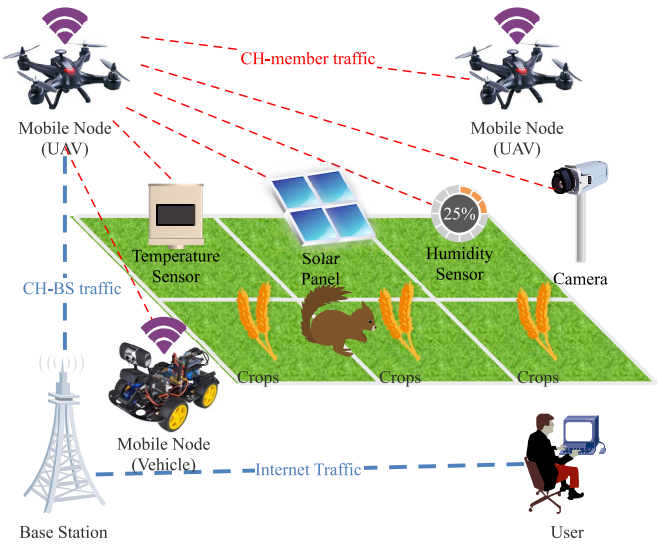


Fig. 7. Possible application of RANCE in IoT-based smart agriculture.

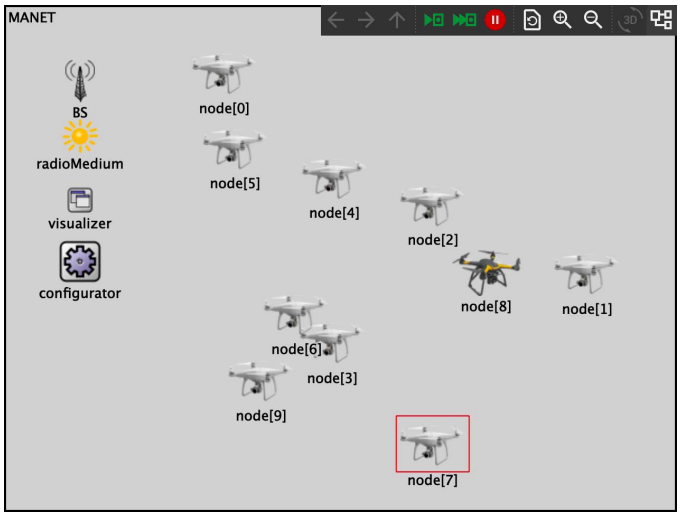


Fig. 8. OMNeT++ simulation.

were responsible for data forwarding of image and recognized results [see Fig. 6(b)].

B. Possible Applications in IoT

RANCE does not impose restrictions on link-layer specifications. Therefore, LPWAN [35] (low-power wide area networks, e.g., LoRa and NB-IoT) technologies that are widely adopted in IoT can also be used as the communication substrate for RANCE, in addition to IEEE 802.15.4 usually used in WSN/WPAN, 802.11-like technologies usually used in MANET, etc. This paves the way for RANCE's application on top of IoT, as long as RANCE PDUs are encapsulated in IoT packets, and RANCE agents are deployed on IoT devices. On the one hand, IoT settles the networking between smart objects. On the other hand, RANCE on top of IoT deals with task-oriented clustering and collaboration, enriching IoT's functionalities. One of IoT's remarkable applications is the smart agriculture that requires long-term mobile surveillance over unattended agricultural assets. We envision RANCE's application in IoT-based smart agriculture as shown in Fig. 7. In this example, mobile devices (e.g., UAVs and vehicles) and fixed IoT devices (e.g., various sensors and cameras) form a hybrid cluster using RANCE. Agricultural crops can be full-angle monitored with both fixed and mobile devices. Harmful animals can be identified by devices with recognition functions, and recognized results are sent to CH (i.e., the UAV on the upper left corner) for further process. CH can then command some node (e.g., the vehicle on the lower left corner) with alarm device to attempt to drive these animals away. Mobile nodes can even have fixed IoT devices under surveillance to check if they work properly. These collaborative functions can be derived from clustering by RANCE, and orchestrated as a service function chain. In addition, IoT devices can be clustered in an on-demand manner with other mobile devices using RANCE, if current mobile devices move away. In a word, RANCE is able to help build a mobile ad hoc IoT with heterogeneous devices.

VI. Evaluation

To evaluate the functionalities and performances, we implemented RANCE using OMNeT++ and INET framework, we conducted in this article a series of simulations with comparison against four LEACH-like protocols, namely, LEACH [4], DMH-LEACH [9], LEACH-C [4], and CEECR [30] (see Section II). These five protocols are representative in that RANCE is a randomly centralized protocol, and LEACH and DMH-LEACH are distributed ones, and LEACH-C and CEECR are centralized ones. Simulations were conducted in a wireless mobile environment where IEEE 802.15.4 was adopted for the data-link and physical layers. Nodes were randomly distributed in a constrained space initially, and moving continuously in a speed uniformly distributed between 10 and 20 m per second, with a direction change uniformly distributed between 0° and 30°. Several simulation sets were conducted, each of which lasts 5 h. For LEACH-like simulations, the duration of each round is 200 s, and CH percentage p is 0.2.

A. Simulation Set 1: Fixed Dimensions and Fixed Size

The purpose of simulation set 1 was to explain how RANCE works, and compare the working procedure of RANCE and LEACH-like protocols, through the runtime status of a specific node[0] in a small-scale 802.15.4 wireless network. The space dimensions of the environment are fixed (1600×1600 m^2) and the wireless network has a fixed number of nodes (ten mobile nodes), as shown in Fig. 8. Black nodes represent selected CHs while gray ones represent cluster members. The simulation results are shown in Fig. 9. Performances of these protocols were compared with regard to the following metrics.

1) Clustered Time: Results are shown in Fig. 9(a). RANCE works in a randomly centralized fashion in which every node has an even chance to initiate cluster formation. In addition, cluster relationships are maintained constantly through bidirectional heartbeats. Once any member gets detached, CH and the member itself quickly notices it and starts the clustering process on-demand. Therefore, we can see from the figure

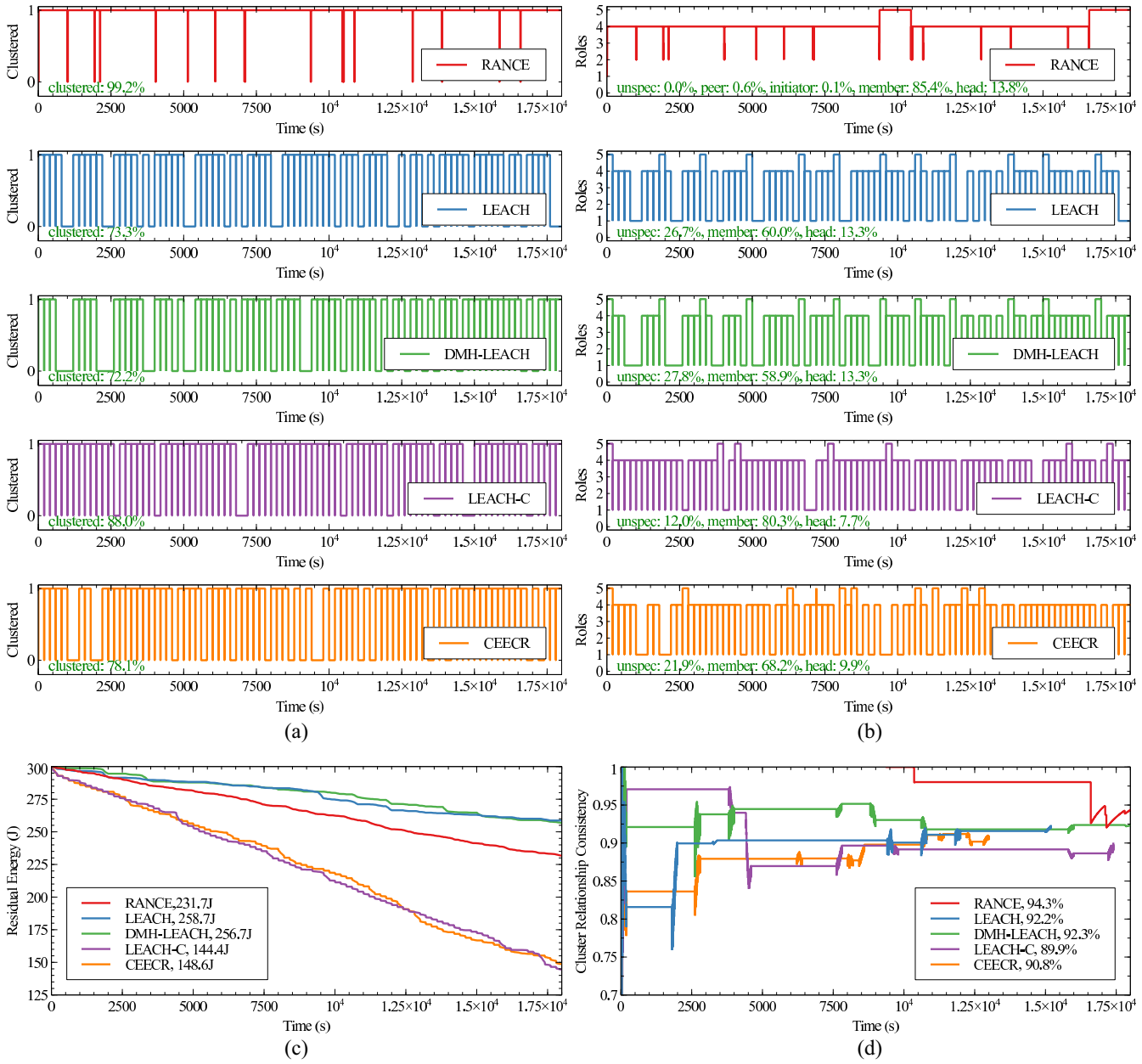


Fig. 9. Runtime status of node[0] using different protocols, fixed space dimension, and fixed network size. (a) Clustered time. (b) Role distribution. (c) Energy consumption. (d) Cluster relationship consistency.

that RANCE has a clustered percentage as high as 99.2% [where numeric value 1 stands for "clustered" in Fig. 9(a)]. Meanwhile, RANCE works in an event-driven manner in which the (re-)clustering occurs only when there happens something that indicates the necessity for clustering, such as moving out of the CH's radio coverage, weak signal strength, low battery, etc. So we can see from the figure that the RANCE's clustered time distribution was irregular and event triggered. Therefore, unnecessary CH handovers are reduced at large in RANCE.

On the contrary, LEACH-like protocols work in a time-triggered manner where (re-)clustering occurs repeatedly when the round duration times out, necessary or not. So we can see from Fig. 9(a) that the clustered time distributions were very regular. However, the clustered time percentages of LEACH-like protocols were lower than RANCE due to their lack of cluster relationships maintenance, causing unawareness of detached nodes. Among LEACH-like ones, centralized protocols (LEACH-C and CEECR) had better percentages, 88.0% and 78.1%, respectively, since nodes can always contact remote BS for clustering, though at the cost of higher energy consumption due to remote communication with BS. Among others, CEECR tends to select more stable (i.e., less mobile) nodes as CHs, which lowers the clustered time percentage in a mobile environment because dynamic radio coverage is smaller due to weaker mobility. Distributed protocols (LEACH and DMH-LEACH) have even lower clustered time percentages, 77.8% 77.1%, indicating that the stochastic approach for

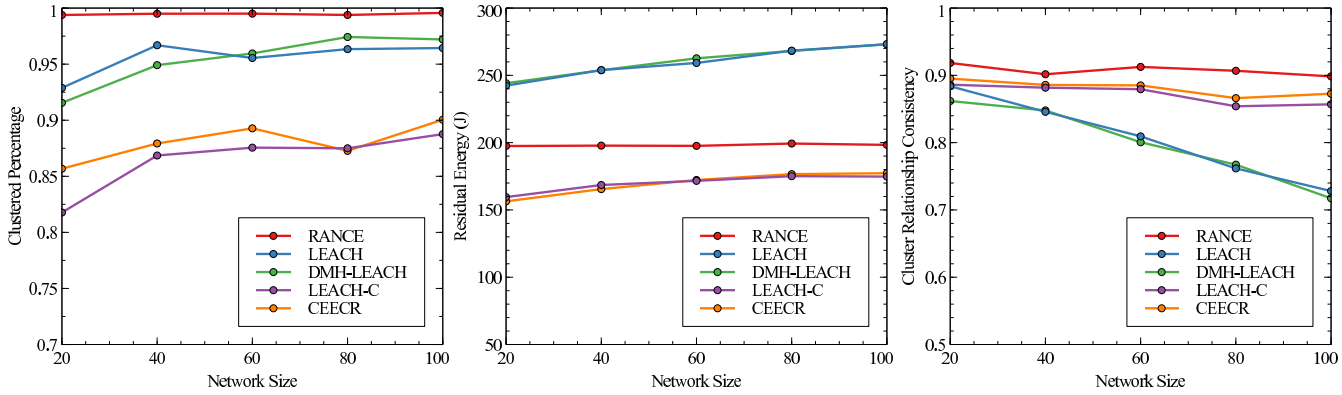


Fig. 10. Runtime network status using different protocols, fixed space dimension, and variable network size.

CH selection is not very applicable in a mobile environment. In a word, the key of RANCE's clustered time advantage over other counterparts in a mobile wireless environment is the cluster relationship maintenance (Section IV-C), which is absent in LEACH-like protocols.

2) Role Distribution: Role distribution represents how and when the lifetime of a node is divided into different roles, as shown in Fig. 9(b). Numeric values 1–5 represent roles of unspec, peer, initiator, member, and head, respectively, as specified in Table III. We can see from the figure that role distribution of LEACH-like protocols is more even, yet role switching is more frequent, causing cluster relationships to be re-established more often. This is beneficial to energy consumption balancing, but, otherwise, to continuous services, which are likely to require stable CH-member relationships. RANCE focuses on prolonging stable CH-member relationships, to better support continuous services. It conducts CH handovers in an on-demand manner when current CHs are no longer suitable.

3) Energy Consumption: Energy consumption is evaluated by the residual energy of a node, as shown in Fig. 9(c). Among LEACH-like ones, centralized protocols (LEACH-C and CEECR) have to report node status, such as energy, speed, etc., to the remote BS that selects CHs accordingly. Long-range communication is very costly, thus high energy consumption for LEACH-C (144.4 J) and CEECR (148.6 J). Distributed protocols adopt the stochastic self election approach without counseling the remote BS and advertise the result in the local radio coverage, thus lower energy consumption (LEACH 258.7.2 J and DMH-LEACH 256.7 J). RANCE (231.7 J) conducts cluster relationship maintenance constantly, thus higher consumption compared with distributed protocols, but still lower than centralized protocols because RANCE does not require long-range communication with BS.

4) Cluster Relationship Consistency: The results are shown in Fig. 9(d). The existence of inconsistency might be caused by delayed or unsuccessful arrivals of heartbeat packet interactions in RANCE, and the absence of maintenance during cluster lifetime in LEACH-like protocols, respectively. The cluster relationship maintenance procedure designed by RANCE enables it to distinguish unclustered members (i.e., false members) through heartbeat packets and eliminate them in a timely manner, providing consistency as high as 94.3%. For LEACH-like protocols, CHs have to wait for the the beginning of the next round to implicitly eliminate false members. During this period, CHs are not aware of false CH-member relationships, thus the inconsistency. Distributed protocols LEACH and DMH-LEACH provide consistency as much as 92.2% and 92.3%, respectively, whereas centralized protocols LEACH-C and CEECR provide consistency as much as 89.9% and 90.8%, respectively, lower than RANCE.

B. Simulation Set 2: Fixed Dimensions and Variable Size

The purpose of simulation set 2 was to compare performances of these protocols, in larger networks, where the space dimensions were fixed as 1600 m × 1600 m and the network size increased by 20 for every simulation run, starting from 20. The results are shown in Fig. 10, which exhibits the average of all nodes' corresponding metrics at the end of simulations.

With regard to clustered status (the left subfigure), we can see that along with the increment of nodes, the clustered percentages (i.e., how long a node stays attached to a cluster as opposed to its total lifetime) of LEACH-like protocols increased, and converged when the number of nodes reaches about 100. This indicated that LEACH-like protocols work better in denser networks. Distributed protocols (LEACH and DMH-LEACH) exhibited remarkable increment of clustered percentage (96.4% and 97.2% for 100 nodes), and surpassed centralized protocols (LEACH-C 88.8% and CEECR 90.0%). It also indicated that centralized protocols do not scale well to larger networks. RANCE exhibited excellent performance for clustered percentage (above 99% for all scenarios), almost irrelevant to the number of nodes due to the on-demand clustering feature enabled by fine-grained maintenance.

With regard to energy consumption (the middle subfigure), LEACH (273.2 J, when the number of nodes reached 100) and DMH-LEACH (273.0 J) outperformed other protocols since they do not have to undertake heartbeat interactions as does RANCE (198.3 J), nor the long-range communication with BS in centralized protocols (LEACH-C 174.8 J and CEECR 177.3 J). Meanwhile, RANCE offered very stable energy consumption that outperformed LEACH-C and CEECR. Notice that in Fig. 10, there was less energy consumption with the

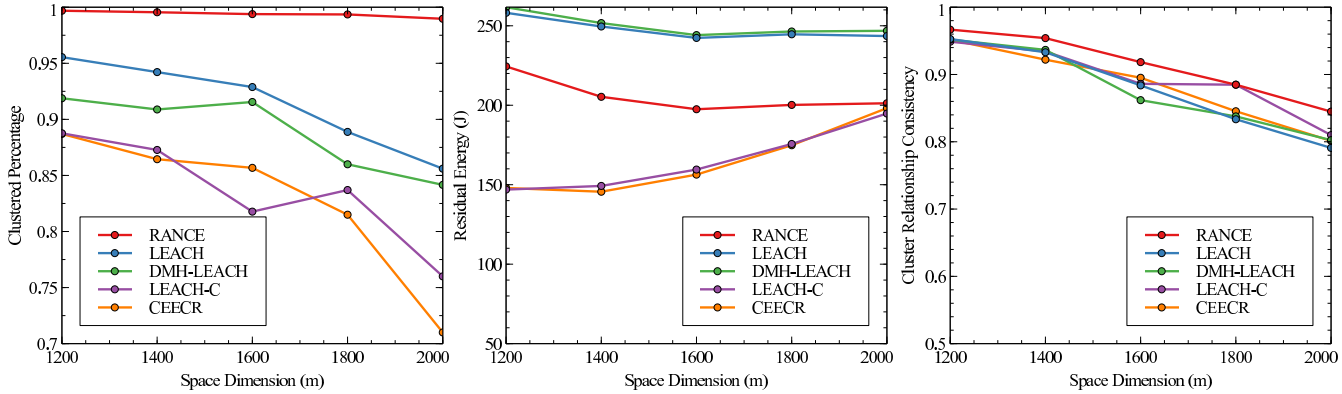


Fig. 11. Runtime network status using different protocols, variable space dimension, and fixed network size.

increment of node number for LEACH-like protocols. The reason for this phenomenon might be that for a denser network, CH advertisements are more likely to be received by other nodes (i.e., CHs competing for members). It results in higher percentage of CHs that hold a smaller cluster with fewer members. Smaller clusters require less energy for inner cluster communications. However, energy consumption for LEACH-like protocols eventually converged when the number of nodes reaches about 100.

With regard to cluster relationship consistency (the right subfigure), RANCE outperformed other protocols with the help of fine-grained maintenance that eliminates inconsistent CH-members relationships stored in CHs' cluster_rel in time. RANCE provides stable consistency even though the number of nodes reached 100 (about 90%) whereas other protocols had demonstrated drops of consistency, especially for distributed protocols (LEACH 72.8% and DMH-LEACH 71.7%).

C. Simulation Set 3: Variable Dimensions and Fixed Size

The purpose of simulation set 3 was to compare performances of these protocols, in larger networks, where the network size was fixed as 20, and the space dimensions increased by 200 m for each side in every simulation run, starting from 1200 m $\times$ 1200 m. The results are shown in Fig. 11, which exhibits the average of all nodes' corresponding metrics at the end of simulations.

With regard to clustered time (the left subfigure), we can see that along with the increment of the space dimensions, the clustered percentages of LEACH-like protocols had very obvious drops, especially for centralized protocols (LEACH-C and CEECR). This was consistent with the analysis as revealed in simulation set 2 that LEACH-like protocols work better in denser networks, but otherwise in more scattered ones. The reason for this phenomenon is that for distributed protocols (LEACH and DMH-LEACH), the distribution of CHs relies on stochastic approaches, in which CHs are not guaranteed to be evenly placed among nodes for every round. For example, CHs might gather tightly in a certain area in extreme cases. Nodes far away from these gathered CHs are likely to get detached. It also indicated that centralized protocols (LEACH-C and CEECR) are not very scalable to wider space dimensions where nodes are mobile. Meanwhile, the BS had its radio coverage limit, beyond which packets necessary for clustering cannot reach nodes, thus higher rate of detached nodes. RANCE exhibited good performance in clustered percentage due to that detached nodes initiate cluster formation autonomously in a timely fashion.

With regard to energy consumption (the middle subfigure), there was a noticeable phenomenon, which appeared contradictory to intuition in that it cost less energy for centralized protocols (LEACH-C and CEECR) in wider space, against the radio model as described in 16. Actually, there was a reasonable explanation for this. Since there were more detached nodes for LEACH-C and CEECR, CHs had fewer attached members in their own clusters, thus lower energy consumption. That is to say, the seemingly drop of energy consumption for centralized protocols, on the contrary, indicated again their inapplicability in clustering for wider space dimensions.

Meanwhile, for larger dimensions, nodes are more scattered. Due to limited radio coverages, nodes beyond these limits are hidden terminals to each other, thus no chance to be mutually clustered. That is to say, along with the increment of space, there might be a relatively "fixed" number of nodes in a cluster, reaching the balance between the cluster size and space dimensions. It further reaches the balance between the increase of energy consumption due to long transmission range, and the decrease of energy consumption due to fewer number members in a cluster. This analysis applies for RANCE and distributed protocols (LEACH and DMH-LEACH), neither of which require fixed BS.

With regard to cluster relationship consistency (the right subfigure), we can see from the results that all protocols had gradual drops in wider space, because nodes were more likely to be placed at the edge of each other's radio coverage, resulting in higher possibility to get detached from CHs. RANCE provided better consistency performance due to its heartbeats-based maintenance that timely discovered and removed falsely attached members from CHs' cluster_rel.

D. Simulation Set 4: Scalability Comparison in Even Larger Networks

The purpose of simulation set 4 was to compare clustering scalability of different protocols in even larger networks with 100–300 nodes. The space dimensions were 1600 m

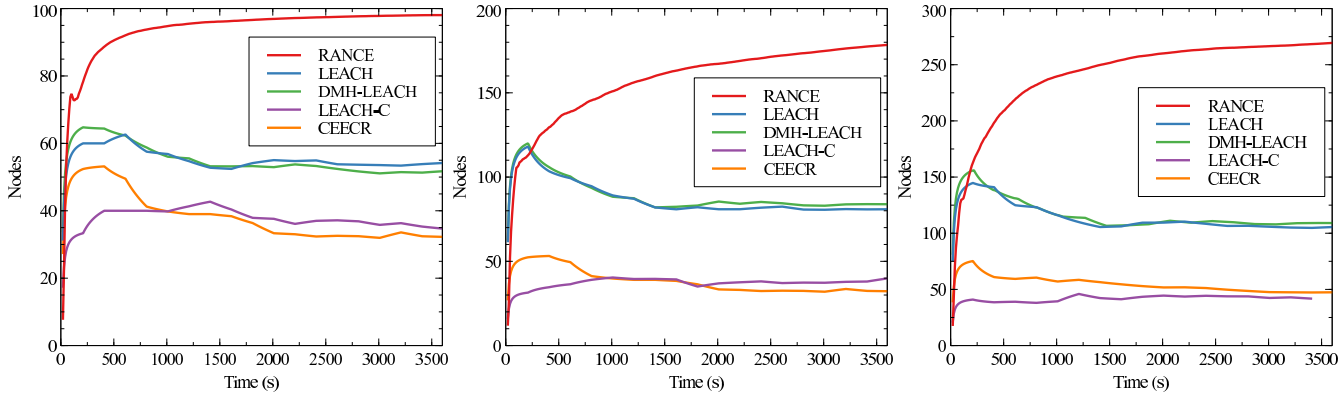


Fig. 12. Runtime network scalability using different protocols, fixed space dimension, and variable network size.

$\times$1600 m. The results are shown in Fig. 12, which exhibits the number of clustered nodes among all nodes to demonstrate how protocols scale to the network size. On average, RANCE clustered about 97 nodes in a 100-node network, 178 nodes in a 200-node network, and 270 nodes in a 300-node network. It can be seen that the clustering performance of RANCE scales well with the network size. The reason for its good scalability is that RANCE provides a fine-grained maintenance mechanism, and conducts on-demand clustering in a timely fashion once detached nodes are discovered. Also, existing clusters can be hitchhiked by detached nodes through the hitchhiking mechanism. LEACH clustered about 54 nodes in a 100-node network, 81 nodes in a 200-node network, and 106 nodes in a 300-node network, on average. The reason for this phenomenon is that LEACH (and possibly other similar distributed clustering protocols) selects CHs in a stochastic approach. CHs might not be chosen where they are most needed to cover as many unclustered nodes. DMH-LEACH provided similar clustering scalability. Meanwhile, centralized clustering protocols provided even lower clustering scalability. LEACH-C clustered about 35 nodes in a 100-node network, 40 nodes in a 200-node network, and 42 nodes in a 300-node network. It can be seen that LEACH-C did not scale well with network size. CEECR provided similar clustering scalability. One way to improve the scalability of LEACH-like protocols is to select more CHs [i.e., assigning larger p in (1)]. However, to define an optimal value for p a prior is not a trivial task in a dynamic and mobile wireless network.

To summary, RANCE outperforms LEACH-like protocols with regard to clustering scalability in larger networks. The major reason is that the number of CHs in RANCE is adaptive to the topology changes due to on-demand clustering, while that in LEACH-like protocols is determined a prior and usually stays unchanged during the network lifetime.

VII. Conclusion

In this article, we proposed RANCE, a randomly centralized and on-demand clustering protocol for MANETs. Compared with conventional LEACH-like protocols, RANCE provides several promising features, which we believe fit well in applications in MANETs, compared with LEACH-like protocols.

1) RANCE's random centralization guarantees the optimality when selecting CHs. In addition, the role to centrally and optimally select CHs is only competed in the local mobile wireless network without evolving remote BS to avoid high energy consumption.
2) RANCE's fine-grained maintenance guarantees on-demand clustering instead of round-based CH handovers seen in LEACH-like protocols. It offers longer clustered time thus better support for continuous services and collaborative tasks. It also provides good scalability in larger network. In addition, it also improve the consistency between logical cluster relationships and physical topologies.

The simulation results show that RANCE achieves these features at minimum energy cost. It has good potentials in mobile wireless environments that are infrastructureless, for continuous missions and internodes collaborations that involve intensive mobility, such as unmanned collaborative surveillance.

In our future works, we are going to study the following related issue. For multifunctional and time-sequenced tasks (such as collaboration between drones with different equipments in battlefields), a number of functionally heterogeneous nodes might be required to be orchestrated to execute a series of subtasks in a given order. This exhibits functionally constrained composition logics during clustering. In this article, functional heterogeneity and constraints are not studied for these complex tasks. Functionally heterogeneous orchestration of nodes and its optimization can be modeled as service function chaining problems [36], and will be studied in our future work.

Acknowledgment

The authors would like to thank the time and efforts by the editors and reviewers.

References

[1] D. S. Lakew, U. Sa'ad, N.-N. Dao, W. Na, and S. Cho, "Routing in flying ad hoc networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1071–1120, 2nd Quart., 2020.
[2] K. Streit, N. Rodday, F. Steuber, C. Schmitt, and G. D. Rodosek, "Wireless SDN for highly utilized MANETs," in *Proc. Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, 2019, pp. 226–234.

[3] P. Rawat and S. Chauhan, "Clustering protocols in wireless sensor network: A survey, classification, issues, and future directions," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100396.
[4] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.
[5] S. Umbreen, D. Shehzad, N. Shafi, B. Khan, and U. Habib, "An energy-efficient mobility-based cluster head selection for lifetime enhancement of wireless sensor networks," *IEEE Access*, vol. 8, pp. 207779–207793, 2020.
[6] A. Verma, S. Kumar, P. R. Gautam, T. Rashid, and A. Kumar, "Fuzzy logic based effective clustering of homogeneous wireless sensor networks for mobile sink," *IEEE Sensors J.*, vol. 20, no. 10, pp. 5615–5623, May 2020.
[7] Y. Chang, H. Tang, B. Li, and X. Yuan, "Distributed joint optimization routing algorithm based on the analytic hierarchy process for wireless sensor networks," *IEEE Commun. Lett.*, vol. 21, no. 12, pp. 2718–2721, Dec. 2017.
[8] Y. Zhang, T. T. Liu, H. G. Zhang, and Y. A. Liu, "LEACH-R: LEACH relay with cache strategy for mobile robot swarms," *IEEE Wireless Commun. Lett.*, vol. 10, no. 2, pp. 406–410, Feb. 2021.
[9] M. Elmonser, H. Ben Chikha, and R. Attia, "Mobile routing algorithm with dynamic clustering for energy large-scale wireless sensor networks," *IET Wireless Sens. Syst.*, vol. 10, no. 5, pp. 208–213, 2020.
[10] G. Tanganelli, A. Virdis, and E. Mingozzi, "Enabling multi-hop forwarding in 6LoWPANs through software-defined networking," in *Proc. IEEE 20th Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2019, pp. 1–9.
[11] J. Wang and X. Zhang, "Cooperative MIMO-OFDM-based exposure-path prevention over 3D clustered wireless camera sensor networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 4–18, Jan. 2020.
[12] C. Wang, Y. Zhang, X. Wang, and Z. Zhang, "Hybrid multihop partition-based clustering routing protocol for WSNs," *IEEE Sens. Lett.*, vol. 2, no. 1, Mar. 2018, Art. no. 7500504.
[13] I. Sohn, J.-H. Lee, and S. H. Lee, "Low-energy adaptive clustering hierarchy using affinity propagation for wireless sensor networks," *IEEE Commun. Lett.*, vol. 20, no. 3, pp. 558–561, Mar. 2016.
[14] V. Rajpoot *et al.*, "Analysis of machine learning based LEACH robust routing in the edge computing systems," *Comput. Elect. Eng.*, vol. 96, Dec. 2021, Art. no. 107574.
[15] X. Cai, S. Geng, D. Wu, L. Wang, and Q. Wu, "A unified heuristic bat algorithm to optimize the LEACH protocol," *Concurrency Comput. Pract. Exp.*, vol. 32, no. 9, p. e5619, 2020.
[16] A. Mehmood, Z. Lv, J. Lloret, and M. M. Umar, "ELDC: An artificial neural network based energy-efficient and robust routing scheme for pollution monitoring in WSNs," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 1, pp. 106–114, Jan.–Mar. 2020.
[17] Y. Jaradat, M. Masoud, and I. Jannoud, "A mathematical framework of optimal number of clusters in 3D noise-prone WSN environment," *IEEE Sensors J.*, vol. 19, no. 6, pp. 2378–2388, Mar. 2019.
[18] S. Tanwar, S. Tyagi, N. Kumar, and M. S. Obaidat, "LA-MHR: Learning automata based multilevel heterogeneous routing for opportunistic shared spectrum access to enhance lifetime of WSN," *IEEE Syst. J.*, vol. 13, no. 1, pp. 313–323, Mar. 2019.
[19] T. Kaur and D. Kumar, "Particle swarm optimization-based unequal and fault tolerant clustering protocol for wireless sensor networks," *IEEE Sensors J.*, vol. 18, no. 11, pp. 4614–4622, Jun. 2018.
[20] K. G. Omeke *et al.*, "DEKCS: A dynamic clustering protocol to prolong underwater sensor networks," *IEEE Sensors J.*, vol. 21, no. 7, pp. 9457–9464, Apr. 2021.
[21] C. Hao and C. Jiang, "Robust wireless sensor network against strong electromagnetic pulse," *IEEE Sensors J.*, vol. 21, no. 4, pp. 5572–5579, Feb. 2021.
[22] X. Lin, W. Mei, and R. Zhang, "A new store-then-amplify-and-forward protocol for UAV mobile relaying," *IEEE Wireless Commun. Lett.*, vol. 9, no. 5, pp. 591–595, May 2020.
[23] M. S. Bahbahani and E. Alsusa, "A cooperative clustering protocol with duty cycling for energy harvesting enabled wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 101–111, Jan. 2018.
[24] Y. Fathy and P. Barnaghi, "Quality-based and energy-efficient data communication for the Internet of Things networks," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10318–10331, Dec. 2019.
[25] J. Luo, Y. Chen, M. Wu, and Y. Yang, "A survey of routing protocols for underwater wireless sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 137–160, 1st Quart., 2021.
[26] R. Prajapat, R. N. Yadav, and R. Misra, "Energy-efficient k-hop clustering in cognitive radio sensor network for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13593–13607, Sep. 2021.
[27] P. Neamatollahi, M. Naghibzadeh, S. Abrishami, and M.-H. Yaghmaee, "Distributed clustering-task scheduling for wireless sensor networks using dynamic hyper round policy," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 334–347, Feb. 2018.
[28] M. Ahmad, A. Hameed, A. A. Ikram, and I. Wahid, "State-of-the-art clustering schemes in mobile ad hoc networks: Objectives, challenges, and future directions," *IEEE Access*, vol. 7, pp. 17067–17081, 2019.
[29] S. Varshney and R. Kuma, "Variants of LEACH routing protocol in WSN: A comparative analysis," in *Proc. 8th Int. Conf. Cloud Comput. Data Sci. Eng. (Confluence)*, 2018, pp. 199–204.
[30] J. Zhang and R. Yan, "Centralized energy-efficient clustering routing protocol for mobile nodes in wireless sensor networks," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1215–1218, Jul. 2019.
[31] U. Baroudi, "Robot-assisted maintenance of wireless sensor networks using wireless energy transfer," *IEEE Sensors J.*, vol. 17, no. 14, pp. 4661–4671, Jul. 2017.
[32] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2, 2000, p. 10.
[33] X. Chen, T. Wu, and Y. Tan, "The SDN-governed ad hoc swarm for mobile surveillance of meteorological facilities," in *Quality, Reliability, Security and Robustness in Heterogeneous Systems*. Cham, Switzerland: Springer Int., 2021, pp. 57–75.
[34] X. Chen, T. Wu, G. Sun, and H. Yu, "Software-defined MANET swarm for mobile monitoring in hydropower plants," *IEEE Access*, vol. 7, pp. 152243–152257, 2019.
[35] A. Ikpehai *et al.*, "Low-power wide area network technologies for Internet-of-Things: A comparative review," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2225–2240, Apr. 2019.
[36] G. Sun, Z. Xu, H. Yu, X. Chen, V. Chang, and A. V. Vasilakos, "Low-latency and resource-efficient service function chaining orchestration in network function virtualization," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5760–5772, Jul. 2020.

Xi Chen received the B.S. and Ph.D. degrees in computer science from Southwest Jiaotong University, Chengdu, China, in 2007 and 2013, respectively.

He is an Associate Professor with the School of Computer Science and Engineering, Southwest Minzu University, Chengdu, China. From 2011 to 2012, he was a visiting and joint Ph.D. student with the School of Computer Science and Engineering, The University of New South Wales, Kensington, NSW, Australia. He is also a Postdoctoral Fellow with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu. His research interests include software-defined networking, network function virtualization, service function chaining, cloud computing, IoT, wireless networks, and machine learning.

Gang Sun (Member, IEEE) received the Ph.D. degree in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2012.

He is a Professor with the University of Electronic Science and Technology of China. He has coauthored more than 90 technical publications including paper in refereed journals and conferences, invited papers and presentations, and book chapters. His research interests include network virtualization, cloud computing, high performance computing, parallel and distributed systems, ubiquitous/pervasive computing, and intelligence and cyber security.

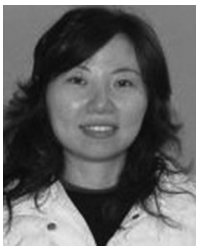
Prof. Sun has also edited special issues at top journals, such as *Future Generation Computer Systems* and *Multimedia Tool and Applications*. He has served as a Reviewer for IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, ACM/IEEE TRANSACTIONS ON NETWORKING, IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON NETWORK SERVICES MANAGEMENT, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, *IEEE Wireless Communications Magazine*, IEEE COMMUNICATIONS LETTERS, *Information Fusion*, *Future Generation Computer Systems*, *Journal of Network and Computer Applications*, *Journal of Supercomputing*, *Journal of Parallel and Distributed Computing*, and *Chinese Journal of Electronics.*

Tao Wu received the B.S. and Ph.D. degrees in computer science from Southwest Jiaotong University, Chengdu, China, in 2007 and 2014, respectively.

She is an Associate Professor with the School of Computer Science, Chengdu University of Information Technology, Chengdu. Her research interests include machine learning, wireless networks, and IoT.

Ling Liu received the Ph.D. degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2021.

She is currently a Lecturer with the College of Electronic and Information, Southwest Minzu University, Chengdu. Her research interests include distributed machine learning, network topology, and network scheduling and optimization.



Hongfang Yu (Member, IEEE) received the B.S. degree in electrical engineering from Xidian University, Xi'an, China, in 1996, and the M.S. and Ph.D. degrees in communication and information engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1999 and 2006, respectively.

She is a Professor and a Doctoral Supervisor with the School of Information and Communication Engineering, University of Electronic Science and Technology of China. From 2009 to 2010, she was a Visiting Scholar with the Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA. Her research interests include software-defined networking, network function virtualization, service function chaining, and cloud computing.

Mohsen Guizani (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from Syracuse University, Syracuse, NY, USA.

He is currently a Professor and an Associate Provost with the Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. His research interests include applied machine learning, smart city, wireless communications/networking, cloud computing, security, and its application to healthcare systems.

Prof. Guizani is currently serving on the editorial boards of many IEEE transactions and magazines.