Received November 11, 2020, accepted December 3, 2020, date of publication December 10, 2020, date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3043762

Fast Multicast With Adjusting Transmission Power and Active Slots in Software Define IoT

ANG LI[1], WEI LIU[2,3], SHAOBO ZHANG[4], AND MANDE XIE[5], (Member, IEEE)
[1]School of Computer Science and Technology, Hunan Institute of Technology, Hengyang 421002, China
[2]School of Informatics, Hunan University of Chinese Medicine, Changsha 410208, China
[3]School of Computer Science and Engineering, Central South University, Changsha 410083, China
[4]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[5]School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Corresponding author: Mande Xie (xiemd@zjgsu.edu.cn)

This work was supported in part by the Natural Science Foundation of Hunan Province, China, under Grant 2019JJ40064 and Grant 2019JJ40063, in part by the Scientific Research Project of Education Department of Hunan Province under Grant 19B142, in part by the Science-Technology of Hengyang City, China, under Grant 2012KS19, and in part by the National Natural Science Foundation of China, under Grant 619723525.

ABSTRACT Billions of Internet of Things (IoT) devices are deployed in a variety of scenarios to monitor events or objects of interest, thereby enhancing the quality of human living. Software define technology can expand the functions of IoT devices by updating the code so that it can evoke new life. However, how to disseminate the code to a given set of sensor nodes quickly and with long lifetime still faces challenges. In this paper, we are fully aware that most nodes in the sensor network have energy surplus, so this energy can be fully utilized to optimize code dissemination performance. So, we first propose an Adjusting Transmission Power based Code Multicast (ATP-CM) algorithm, which can effectively reduce the delivery delay of code dissemination without reducing the network lifetime. After that, an Augmenting Active Slots based Code Multicast (AAS-CM) algorithm is proposed, which can reduce broadcast waiting time and speed up code dissemination. Combining the above two improvements, an Adjusting Transmission Power and Augmenting Active Slots based Code Multicast (ATP-AAS-CM) scheme is proposed, which can greatly reduce delivery delay while maintaining a high lifetime. Extensive experimental results and analysis indicate that the scheme proposed in this paper can effectively increase the energy utilization by 13.43%-34.18%, reduce the delivery delay by 63.88%-77.38%. Meanwhile, its lifetime is the same as that of previous schemes, which fully illustrates the effectiveness of the ATP-AAS-CM scheme.

INDEX TERMS Internet of Thing, code dissemination, multicast, delivery delay, energy efficient.

I. INTRODUCTION

The development of microprocessor technology has greatly promoted the Internet of Things (IoT) [1]–[3]. According to the prediction from Gartner, by 2020, smart devices will reach 20.4 billion [4]. Most of these IoT devices are equipped with sensing devices that can sense the physical phenomena of surrounding environment, so that the system can deal with events accordingly [5]–[7]. Currently, these IoT devices are deployed in many application fields, such as transportation [8], [9], medical and health [10], [11], environmental monitoring [12], [13], industrial production, etc. [14]–[16]. Because these IoT devices are mainly deployed at the edge of network [17]–[19], and have powerful storage and processing capability [20], [21]. With the development of 5G [22]–[24], the utilization of IoT devices has been promoted more rapidly. Therefore, new network computing models such as Edge computing [25]–[27], Fog computing [17] have emerged, making the current network computing model shift from cloud to edge [28]. New computing models make full use of IoT devices deployed at the edge of the network to perceive data, and analyze and process data at the same time [29]–[31]. Since users accessing network services are all located at the edge of the network, users can get services nearby [3], [6], thus reducing the delay, jitter, and high energy consumption caused by sending service requests to the cloud through long routing [12], [14]. In particular, the combination of current edge computing and artificial intelligence has made

The associate editor coordinating the review of this manuscript and approving it for publication was Takuro Sato.

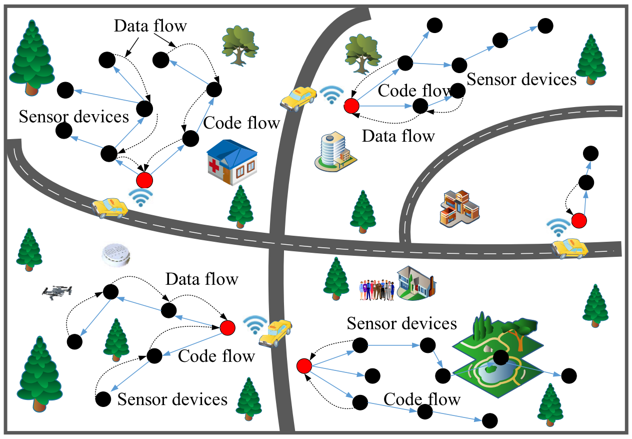

FIGURE 1. Network application scenarios.

distributed deep data analysis and processing more common [32], [33], which has further promoted the widespread application and deployment of IoT devices [34]–[36].

Wireless sensor networks (WSNs) are one of the earliest and most widely studied sensor-based networks [37]–[40]. In such a network, many sensor nodes are deployed in the areas to be monitored [41]–[43]. Among them, there is a special node called sink connected to the Internet. All other sensor nodes form a network through self-organization, and transmit the sensed data to the sink through multi-hop routing [44]–[46]. However, with the development of Internet of Things [1]–[3], senor-based network or called cloud sensor system exists in a more flexible and extensive way. Figure 1 is such an application scenario [4], [47], [48]. In the cloud sensor system, sensor nodes self-organize into a network on the one hand, and on the other hand, their data collection methods are more flexible and diversified. It is not necessary to connect to the Internet through the sink node like traditional WSNs, but to collect data through emerging tools such as mobile vehicles, Unmanned Aerial Vehicles (UAVs), which further expands the scope of their applications [49]–[51].

Another technology that promotes the further development of IoT devices is software define [41], [42]. Software define networks allow IoT devices to perform system updates or upgrades, thus invoking new functions. Due to the number of IoT devices is large [41], [42], [52], [53], redeployment is expensive, some devices are difficult to update, and other production activities need to be stopped when updating, which is costly. In the software define network, only by updating software codes, IoT devices can gain new life [54], and the application range of sensor nodes can be expanded, thus having good development prospects [41], [42].

The combination of software define technology and IoT devices has greatly expanded application prospects [41], [42]. Figure 1 shows such application scenarios. In such a network, IoT devices are randomly deployed in multiple application sites, such as bridges, buildings, roads, and other facilities [55], [56]. On the one hand, these IoT devices sense the data in the environment and send them to the application center [57], [58]. On the other hand, IoT devices can complete self-update by receiving codes. Therefore, in such applications, the network has two flows, one is data flow, and its flow direction is from sensor devices to data collection centers (such as sinks or other mobile vehicles, etc.) [57], [58]. In data collection, it can be directly connected to the Internet through the sink node like the traditional network, or it can be collected through mobile vehicles like the opportunistic routing method proposed by Bonola *et al.* [47]. The other is code flow [41], [42]. The direction of code flow is opposite to data flow, which is sent from sink to sensor nodes. Because the code update is only performed when sensor devices need to be upgraded, it is not a regular operation [41].

In code update, the frequently used operation is multicast, that is, the code is sent to a specified group of sensor nodes for system update [46]. Two important metrics in multicast are energy consumption and delivery delay [46]. Since sensor nodes are generally powered by battery, their energy is very tight, so code dissemination should save as much energy as possible [59], [60]. There have been many researches on energy saving for sensor nodes, including routing strategies, load balancing, adjusting transmission power, cooperative communication, and other technologies. Among them, the most important technology is the duty cycle mechanism. Since the event of the physical phenomenon monitored by sensor nodes is often happened in seconds, if the node is periodically sleep/awake in milliseconds, then it can not only not lose the monitoring phenomenon, but also can effectively save energy. Therefore, the duty cycle mechanism is widely used in sensor-based networks. The reason why the duty cycle mechanism saves energy is that the energy consumption of nodes in the sleep state is thousands of times less than the energy consumption in the awake state. Therefore, if possible, let nodes be in sleep state as much as possible. However, when nodes are in sleep, the delivery delay of code dissemination will increase. This is because when the node is spreading codes, if the receiver is in sleep state, it cannot perform data reception, and it needs to wait for the receiver to wake up before receiving data, which will increase the delivery delay.

There have been some studies on multicast for sensor-based network. These studies can be divided into two categories. One focuses on how to reduce the energy consumption of nodes, while the other focuses on how to reduce delivery delay. Like Ref [40], [41], [42], the main method used by these multicast strategies to reduce energy consumption is to construct a broadcast tree and adjust the broadcast slot, so that there are more sensor nodes in one broadcast able to receive code, thereby reducing the number of broadcasts and reducing energy consumption. In fact, this is like constructing a minimum spanning tree from the source node to all destination nodes. Each path in the minimum spanning tree represents a code broadcast, so in this case, the number of broadcasts is the least, that is, the energy is the most saved. But the delay for the code to be broadcast to each destination node is not the least. The research of optimizing delivery delay needs to make the code reach each destination node along the minimum route. Currently, the simplest method is to use the shortest path method. In the shortest path method, the node closer to the source node, the more serious its energy consumption, and energy consumption is much greater than the method of saving energy consumption mentioned above. Therefore, Cheng *et al.* [37] proposed a strategy that integrates the above methods. Their method is to make appropriate modifications to the minimum spanning tree based on the construction of the code diffusion tree, to reduce delivery delay while increasing a small number of paths (i.e. the number of code broadcasts), which is a compromise between the previous minimum energy research and the minimum delay research. The method proposed by Gu *et al.* [39] is to reduce the delivery delay by increasing the active slot of nodes. Obviously, if the active slot of the node can be increased, the time required for the sender to wait for the receiver to wake up will be reduced. Unfortunately, their method doesn't consider the additional energy consumed by increasing active slot, which affects the lifetime of the network.

It can be seen from the previous discussion that the existing code multicast strategies have the following situations: 1) The energy saving strategy has a large delivery delay, such as the code dissemination strategy for constructing a minimum spanning tree; 2) The strategy with a small delivery delay consumes more energy, which will damage the lifetime of the network. Such as the strategy of the shortest path, or the strategy of increasing the active slot. To our best knowledge, there is still a lack of a multicast method that can reduce delivery delay without affecting the lifetime of the network. So, in this paper, An Adjusting Transmission Power and Augmenting Active Slots based Code Multicast (ATP-AAS-CM) scheme is proposed to Fast Diffusion Code for Software Define IoT. The main innovations are as follows:

1) We first propose an Adjusting Transmission Power based Code Multicast (ATP-CM) algorithm, which can effectively reduce the delivery delay of code dissemination without reducing the lifetime of network. Reducing delivery delay while not reducing lifetime has not been achieved in previous strategies. However, we achieve it in this paper by making full use of remaining energy. Specifically, the sensor network collects data most of the time. Therefore, the sensor nodes near the source node have high energy consumption, while the nodes in the far source area have surplus energy. Therefore, in this article, we fully utilizing remaining energy to increase the transmission power of nodes, thereby expanding the broadcast range, making the routing hops for broadcasting the code from the source node to the destination node shorter, thereby greatly reducing the delivery delay without affecting the lifetime.

2) An Augmenting Active Slots based Code Multicast (AAS-CM) algorithm is proposed to reduce delivery delay while not affecting lifetime. In the AAS-CM, active slots are added by utilizing the remaining energy of nodes far away from the source node. Therefore, the delay can be reduced without reducing lifetime.

3) Combining the previous two proposals, an Adjusting Transmission Power and Augmenting Active Slots based Code Multicast (ATP-AAS-CM) scheme is proposed, which can further reduce delivery delay while maintaining a high lifetime. Experimental results and analysis confirm that the strategy proposed in this paper is superior to existing schemes, it can increase the energy utilization by 13.429%-34.175%, reduce the delivery delay by 63.88%-77.38%. At the same time, its lifetime is the same as that of previous strategy, which fully proves the effectiveness of the proposed scheme.

The rest of this paper is organized as follows: In Section II, we reviewed the related work. The system model and problem statement are presented in Section III. Then, the ATP-AAS-CM scheme is introduced in Section IV. Performance analysis is presented in Section V. Finally, Section VI. provides conclusion.

II. RELATED WORK

A. METRICS FOR EVALUATING CODE DISSEMINATION

The following metrics are most importantly considered in code dissemination.

1) DELIVERY DELAY

Delivery delay is the time it takes for the code to spread from the source node to destination nodes [41], [42]. Obviously, the smaller the delivery delay, the better. In many applications, especially in Industry Internet of Thing (IIoT), due to the very strict time requirements of industrial automatic production lines, inconsistent code versions in any collaborative node may cause serious accidents. Therefore, when IIoT nodes are required to update codes, the update cycle should be as short as possible, that is, the delivery delay is required to be as small as possible [40]. There are many factors that affect delivery delay. The duty cycle is an important factor affecting the delivery delay. In most cases, the duration t of the time slot is determined by the application and the hardware of the nodes, and is difficult to change. Therefore, what can be changed when the network is deployed is the number of time slots k in a cycle. The larger the k, the longer the time the nodes are in sleep, the more energy can be saved, but it will increase the delivery delay [39]. Delivery delay is also related to the route length of the code. Therefore, when designing code diffusion, the shortest path should be selected for routing. However, this will increase the number of transmissions. Because in multicast, multiple destination nodes should use the common path as much as possible, so the code on the public path is reused, which helps reduce the number of code transmissions to reduce energy consumption. However, this will make the path of some nodes not the shortest, thus increasing the delivery delay. And if the shortest path is used for each destination node, although the delivery delay can be effectively reduced, it increases energy consumption.

2) LIFEITME

There are many definitions about lifetime, such as the death time of the first node, the time when half of the nodes die, and the time when all nodes die [41], [42]. Among them, the time when the first node in the sensor network dies is the most important. Because in a monitoring network, the battery of most sensor nodes cannot be charged, when the first node dies, the network may not be able to cover the entire monitoring area, or if the critical node is dead, the data cannot be transmitted to the sink. The key to prolong lifetime is to reduce the energy consumption of code dissemination [8], [41], [42], [48], [51], [54]. The reduction of energy consumption can be reduced by the number of times the code is sent. Therefore, the more times the code is sent, the more its energy consumption. It is also worth pointing out that simply reducing energy consumption may not be able to effectively improve network lifetime. This is because the main work of sensor nodes is to sense and collect data [3], [6], [7], [49], [50], [58]. In the daily work of sensor nodes, the data flow is from the edge of the network to the sink [3], [6], [7]. This is a many-to-one data collection method, so its energy consumption is that the energy consumption of nodes in the near sink area is large, and the energy consumption of nodes in the far sink area is small [49], [50], [58]. Code multicast in this case can only save the energy consumption of nodes in the far sink area and cannot extend the lifetime. If the remaining energy can be used to improve the performance of the code multicast, it will not reduce the lifetime, but can increase the energy utilization.

3) CODE SENDING TIMES

The code sending times directly reflect the energy consumption. Therefore, the less sending times, the less energy consumption [8], [41], [42].

B. CODE DISSEMINATION IN IoT

The development of Internet of Thing (IoT) has brought about tremendous changes in the network [1]–[3]. In the past research, the code update of WSNs is often concentrated [8], [41], [42], [48], [51], [54]. But with the development of IoT, due to a huge number of IoT devices are deployed in a wide range of application fields, and their data acquisition and code diffusion methods have also undergone tremendous changes, which can be seen from Figure 1. Figure 1 shows the typical application scenarios of IoT devices [48], [51], [54]. In such applications, many IoT devices are deployed in areas that need to be monitored in the smart city. Bonola *et al.* pointed out an application scenario [47]. Some sensing nodes are deployed on both sides of the road and bridge to monitor the deformation of the road or bridge [47], and smart sensing devices are deployed in the street lights on both sides of the road to sense the status of the lights [47]. There are also smart sensing devices deployed in the smart trash can to sense the status of the trash can, and when the trash can is full or more than a certain value, it can remind sanitation workers to deal with it. However, these places and objects that need to be monitored are not fixed and require temporary deployment, so how to collect the data of these sensing devices at low cost is facing challenges. Bonola *et al.* [47] proposed a method for data collection using mobile vehicles with opportunity routing. Their method is that there are many mobile vehicles in the smart city that are constantly moving, when mobile vehicles pass the deployed sensing devices, they can receive the data of the sensing devices [47], so that the data can be collected and sent to the data center for processing. This is a low-cost deployment method of sensing devices, which can quickly deploy sensing devices to areas that need to be monitored on demand. The hardware cost of the deployment is very low [47]. And there is no requirement for network and other infrastructure. At the same time, these sensing devices can be self-organized into a network, which has strong scalability.

There are many studies on code update of IoT devices [8], [41], [42], [48], [51], [54]. Zhao *et al.* proposed an Update Code using Vehicles joint UAVs (UC- VU) scheme [48]. The main points of their work are as follows: 1) In the initial stage, first use Unmanned Aerial Vehicle (UAV) to fly to areas that cannot be reached by mobile vehicles for code dissemination, which can effectively reduce the total time required for code diffusion. 2) The second improvement aims at the problem exist in previous strategies that mobile vehicles need to pass through the code center to obtain the source code [48]. Their strategy is to select k smart devices as temporary code stations. These k code sations are distributed by UAV at the beginning of code proliferation. Therefore, this enables mobile vehicles to obtain codes from $k+1$ code stations. Thereby effectively speeding up code dissemination [48]. 3) Finally, they designed an optimized UAV trajectory strategy so that the UAV can fly to these k code stations as well as all mobile vehicles, to complete the code diffusion at minimum cost [48].

The code diffusion method proposed by Hu *et al.* [51] is somewhat like the above, mainly for scenarios where only mobile vehicles are used for code dissemination. In this scenario, there will be some IoT devices that are not within the communication range of their mobile vehicles, so codes cannot be obtained. Or IoT devices are within the communication range of mobile vehicles, but the mobile vehicles carrying the code have not passed through for a long time, resulting in the situation that the code is not actually obtained [51]. In this case, Hu *et al.* [51] proposed the use of emerging UAV technology for diffusion. First, mobile vehicles are used to spread the code, because mobile vehicles use the opportunistic routing method, so its cost is very low. After a period, send the UAV to traverse the IoT devices that have not received the code. When planning the path of the UAV, Hu *et al.* [51] adopted a clustering method to reduce the amount of calculation and the flight cost of the UAV. The specific method of the flight trajectory is to first cluster the IoT devices that have not obtained the code. In this way, UAV only needs to fly to these clusters, and then traverse the nodes in the cluster area to achieve communication with all IoT devices that require code [51].

Although the above discussion is aimed at the code proliferation of IoT devices, it implies this meaning: For IoT devices that self-organize into a network, if one node in the network receives the code, then this code can be used as the source node to spread the entire network. Therefore, in these studies, the focus is on how to spread code to an IoT device in the network through mobile vehicles or UAV, but there is no discussion on the spread of code in the wireless sensor network. There are also many studies on the code diffusion in WSNs. In such a network, many wireless sensor nodes are deployed to the areas that need to be monitored and self-organized to form a network, and the sink is the center of the entire network, responsible for data collection and code source. Code is sent from the sink to a specified set of destination nodes. In a wider range, as shown in Figure 1, it can be considered as a network composed of multiple WSNs.

C. RESEARCH ON ENERGY EFFICIENT FOR CODE DISSEMINATION

Energy consumption is one of the most important indicators in a sensor-based network. Therefore, it is also a factor to be considered for code dissemination. There are many such studies. In such studies, the most important thing is still how to reduce the times of sending code. According to whether the sensor nodes adopt the duty cycle mechanism, it can be divided into code dissemination in duty cycle based WSNs and code dissemination not in duty cycle based WSNs. Relatively speaking, the code dissemination scheme is relatively simple in non-duty cycle based WSNs. In such a network, after the sender sends the code, all nodes within its sending range can receive the code. Therefore, in order to reduce the number of codes sending, the general strategy adopted is to find a Minimum Connected Dominating Set (MCDS) in the network.

The code dissemination in the duty cycle-based network is much more complicated. In such a network, when the sender sends codes, some nodes within the broadcast range may be in sleep state, and therefore cannot receive the code. Therefore, some researchers have proposed code dissemination schemes adapted to the duty cycle-based network. One of them is the improvement of the code dissemination method adopted by the above non-duty cycle-based network. Specifically, after finding the minimum connected dominating set, the nodes on the minimum connected dominating set do not only perform one-time code transmission operation, but perform multiple code transmissions in the active slot of neighbor nodes. Therefore, in such a network, selecting the minimum connected dominating set and determining the slot in which the node sends the code will affect the number of code transmissions and the delivery delay. Therefore, the complexity of such a strategy is relatively large. The method using the minimum connected dominating set is a central algorithm. The advantage of this type of method is that the results obtained are more optimized, and the energy consumption required for code dissemination is small. However, the disadvantage of this method is that it needs to know the topology of the entire network and the information of the nodes in advance. Then the scheduling information of the code dissemination is sent to each node after the centralized calculation, and then the nodes operate according to the pre-agreed operation sequence. But in practice, the network topology is constantly changing, therefore, the centralized scheduling strategy will face difficulties. In this case, it is best to use the distributed code dissemination strategy.

The simplest method of distributed code diffusion strategy is flooding. In this method, after the source code broadcasts the code, every node that receives the code also broadcasts it. This ensures that every node in the network can receive code. However, in such a method, a serious broadcast storm will occur, which will seriously affect the lifetime of the network. The Counter-Based Broadcast (DCB) scheme [60] is a commonly used effective method to reduce broadcast storms. In the DCB scheme, the system sets two values for each node, one value is called the threshold of count and is usually represented by C_h [60]. Another value is called Random Access Delay (RAD). When the node receives the code, it starts timing and counts the number of received codes C_m. When the time length of the node count reaches RAD, it stops timing, and checks the number of received codes C_m during this period. If the number of codes received in the RAD time period is $C_m > C_h$. It means that the neighboring nodes of this node broadcast code more times, so this node is restricted from broadcasting code [60]. On the contrary, if $C_m < C_h$, then it means that the neighboring nodes of this node broadcast code less times, so this node needs to broadcast code. In this way, the RAD value of the node can be set to control the probability of node broadcasting [60]. Obviously, when the RAD is set for a longer time, the node counts for a longer time, and the node receives more codes. Therefore, The lower the probability of broadcasting. Conversely, if the RAD is set to be smaller, the probability of the node broadcasting is greater, for example, if the value of RAD is 0, the node must broadcast [60].

In fact, in the flooding code dissemination method, there are still many factors that affect the code dissemination [60]. The code spreads from the source node, so those nodes far away from the source node should have priority to be broadcasted, so that the code spreads to a long distance in a short time, thus accelerating the spread of the code. Based on the above ideas, some researchers have proposed to give different broadcast probabilities according to the different distances of the nodes from the source node. Some studies will compare the new broadcast coverage area of different nodes, and the probability of adding a new node with a large broadcast coverage area to broadcast is high. When selecting a broadcast node, the remaining energy of the node must also be considered. Those nodes with more remaining energy are more likely to be selected as the broadcast node.

This dissemination strategy is relatively simple compared to the duty cycle base network. Because in the non-duty cycle-based network, the node broadcasts once in its broadcast range, then all nodes inside can receive its code. In a duty cycle-based network, after a node broadcasts once, only nodes in the active slot among its neighbor nodes can receive it. In response to this situation, Shu *et al.* [61] proposed a scheme called proportion to duty cycle length-based broadcast (PDLB) scheme for duty cycle based wireless sensor networks. The main improvement of the PDLB scheme is that when the node broadcasts, it is based on the period length of the node to select multiple slots in proportion to broadcast. Since there are multiple nodes in the same area, each node selects a certain proportion of slots for broadcast. Therefore, after multiple nodes broadcast multiple times, each node can basically be broadcast, and can receive code [61]. In order to prevent the case where the active slot of a node is different from the broadcast slot in all neighbor nodes and the code cannot be received. The PDLB strategy provides a method of requesting code over time to solve this problem. When a node does not obtain the code after the code dissemination starts and the timeout period for obtaining the code is reached, the node selects the active slot of the neighbor node to initiate a code request, thereby obtaining the code [61].

D. RESEARCH ON DELIVERY DELAY OPTIMIZATION FOR CODE DISSEMINATION

Delivery delay is an important performance indicator in code dissemination. It is affected by the length of the duty cycle and the length of the time slot. But these two factors are fixed in a certain network. Therefore, currently, the code dissemination strategy has the greatest impact on code delivery delay. To make the delivery delay small, the routing path of code propagation should be made as short as possible. However, from the perspective of energy saving, the number of times the code to be sent should be minimized. The most effective method is to establish the Minimum Connected Dominating Set (MCDS) [41], [42]. Constructing MCDS can make the path to destination nodes the shortest, and the shortest paths corresponds to the number of times the code is sent, so the MCDS method can minimize the number of times the code is sent. However, the MCDS method may not necessarily minimize the delivery delay. Because MCDS reuses the routing path as much as possible to reduce the number of code transmissions, but the routing path is constructed so that the routing path from the source node to each target node is not the shortest path, so the delivery delay at this time is not the smallest. However, if the shortest path to each destination node is constructed, the number of routing paths, that is, the number of code transmissions, is often more than that of routes constructed by MCDS. Therefore, delivery delay and energy consumption are often contradictory [41], [42].

In many code disseminations studies [6], WSNs are abstracted into a tree-shaped network, where sink is the root of the tree and the starting point of code dissemination. Some researchers [37] studied the code dissemination strategy when the network has a packet loss rate. Chen *et al.* [37] studied this type of network. Due to the unreliability of the wireless network, when the code fails to be sent, it needs to wait for the next cycle to resend [37]. Obviously, in a network with packet loss, its energy consumption and code delivery delay will increase significantly. In order to reduce the delivery delay in the network with packet loss [37]. Qi *et al.* [62] proposed an effective method. The main measures of their method are: If the son node fails to receive the code in its active slot, then awake it in the active slot of its brother node, and when its parent node sends the code to its brother nodes, he can also receive the codes. Therefore, there is no need to wait until the next cycle to send the code, which greatly reduces the delay [62].

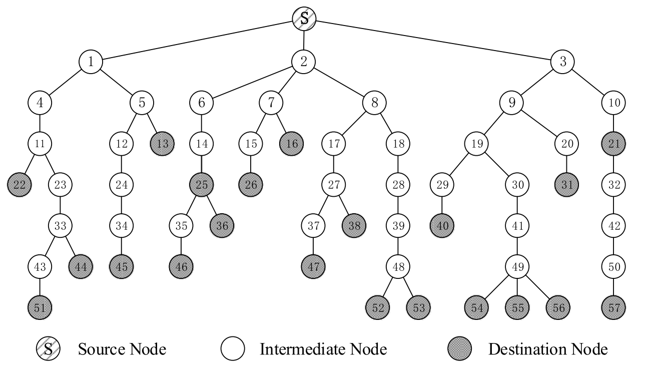

FIGURE 2. WSN characterized by multicast tree structure.

Adding active slots is also an effective way to reduce code delivery delay, especially in a duty cycle-based network. If add an active slot to the son nodes that is the same as the active slot of the parent node, it can effectively reduce the delivery delay.

III. SYSTEM MODEL AND PROBLEM STATEMENT

A. SYSTEM MODEL

We consider a network consisting of N sensor nodes and a source node, which is like Ref. [46]. Sensor nodes are randomly deployed in the network, their transmission radius is r, and each node periodically works in cycles. The research goal is to multicast the codes from the source node to a set of destination nodes, and make the code delivery delay small, energy efficiency high, and lifetime longer.

Like most studies, the wireless sensor network can be represented by a tree with code source node as the root. The transmission of data or broadcast packets from the source node to destination nodes can be described by the multicast tree shown in Figure 2. There are three types of nodes in the multicast tree, namely source node, intermediate node and destination node. Among them, the source node is the node that generates the data or broadcast packets, and packets are forwarded to destination nodes via intermediate nodes. Limited by energy and capability, the transmission power of sensor nodes is limited in traditional multicast schemes, and nodes can only communicate directly with nodes within one hop range (distance within r), nodes beyond communication range need to be relayed by other nodes.

Take the multicast tree shown in Figure 2 as an example to illustrate how to send data in this paper. Assume that

TABLE 1. System parameters.

Parameter	Value	Description
E_{ini}	0.5	Initial energy (J)
T_{com}	100	Communication duration (ms)
ε_t	0.0511	Power consumption in transmission (w)
ε_r	0.0588	Power consumption in receiving (w)
ε_s	$2.4*10^{-7}$	Power consumption in sleeping (w)
t_{pre}	0.26	Preamble duration (ms)
t_{ack}	0.26	Acknowledge window duration (ms)
t_d	0.93	Data packet duration (ms)
r_{ini}	60	Initial transmission radius (m)
M	8	Number of time slots in one cycle

TABLE 2. Parameters related to the calculation.

Notation	Description
D	End to end delay
d_{k-1}^{k}	Delay from the (k-1)-*th* hop to the k-*th* hop
E_{tot}	Total energy consumption
E_{tran}	Energy consumption for transmission
E_{rec}	Energy consumption for reception
E_{LPL}	Energy consumption for low power listening
E_{uti}	Energy utilization rate
E_{rest}	Residual energy
r_{new}	New transmission radius
L	Network lifetime

57 sensor nodes and one source node construct the multicast tree. If the source node S wants to broadcast packets to the destination node N_51, it has to go through a 7-hop route, and the route is S$\rightarrow N_1 \rightarrow N_4 \rightarrow N_{11} \rightarrow N_{23} \rightarrow N_{33} \rightarrow N_{43} \rightarrow N_{51}$.

B. SYSTEM PARAMETERS

In the duty-cycle based wireless sensor network, nodes adopt the asynchronous sleep/wake working mode. A working cycle is divided into slots of equal length. In some of the slots, the node wakes up to receive and send packets, and in the remaining slots, the node sleeps to save energy. The slot in which the node is in the awake state is called the active slot. In a working cycle, the ratio of the time the node is in the wake-up state to the time of the entire cycle is called the duty cycle, which is denoted as ϕ.

$$\phi = \frac{t_{wake}}{t_{total}} = \frac{t_{wake}}{t_{wake} + t_{sleep}} \tag{1}$$

Among them, t_{total} is the duration of a working cycle, t_{wake} is the time the node is in the wake-up state during a working cycle, and t_{sleep} is the time the node is in the sleep state during a working cycle.

System parameters are listed in Table 1.

Parameters in the calculation are shown in Table 2.

C. SYSTEM STATEMENT

1) MINIMIZE DELAY

Delay is defined as the time it takes for a data packet to be sent from the source node to the destination node [41], [42]. It is the sum of the delays of each hop. Assuming that a data packet is sent to the destination node through the multicast tree through a total of K hops, where the delay from the $(k-1)-th$ hop to the $k-$th hop is d_{k-1}^{k}, then the minimized delay can be expressed by formula 2.

$$Min\ D = Min \sum_{k=1}^{K} \mathrm{d}_{k-1}^{k} \tag{2}$$

2) MINIMIZE ENERGY CONSUMPTION

The initial energy of each sensor node is limited. When a node's energy is exhausted, it will die and cannot relay packets. Assuming that the energy consumed by the node for transmitting packets is E_{tran}, the energy consumed by receiving data is E_{rec}, and the energy consumed by low-power listening is E_{LPL}, we does not consider the energy consumption of event sensing in this paper, then the energy consumption of each node can be minimized by formula 3.

$$MinE_{tot} = Min\,(E_{tran} + E_{rec} + E_{LPL}) \tag{3}$$

3) MAXIMIZE ENERGY UTILIZATION

Energy utilization is the ratio of the energy consumed by the network to the initial energy [9], [37]. Assuming that the consumed energy of each node is E_{tot}, the initial energy is E_{ini}, and there are N nodes in the network, the maximum energy utilization can be expressed by formula 4.

$$MaxE_{uti} = Max \sum_{i=1}^{N} \frac{E_{tot}^{i}}{E_{ini}^{i}} \tag{4}$$

4) MAXIMIZE NETWORK LIFETIME

The network lifetime is defined as the time from the initialization of the network to the death of the first node. When the node's initial energy is exhausted, the node will die, and the network lifetime can be obtained at this time. Assuming that the network lifetime is denoted as L, E_{tot}^{i} is the energy consumption of N_i, and E_{ini}^{i} is the initial energy of the node, then maximum network lifetime is expressed as:

$$Max\ L = Max\left(Min \sum_{i=1}^{N} \frac{E_{ini}^{i}}{E_{tot}^{i}}\right) \tag{5}$$

In summary, the research objectives in this paper can be summarized as follows:

$$\begin{cases} Min\ D = Min \sum_{k=1}^{K} \mathrm{d}_{k-1}^{k} \\ Min\ E_{tot} = Min\,(E_{tran} + E_{rec} + E_{LPL}) \\ Max\ E_{uti} = Max \sum_{i=1}^{N} \frac{E_{tot}^{i}}{E_{ini}^{i}} \\ Max\ L = Max\left(Min \sum_{i=1}^{N} \frac{E_{ini}^{i}}{E_{tot}^{i}}\right) \end{cases} \tag{6}$$

IV. DESIGN OF THE ATP-AAS-CM SCHEME

A. RESEARCH MOTIVATION

In most studies on code multicast, the goal is to reduce the number of codes transmission to reduce energy consumption. But it is worth pointing out that although reducing energy consumption can help prolong lifetime. However, lifetime does not equal to energy consumption. Since lifetime is defined as the time when the first node dies. After a comprehensive analysis of the energy consumption of nodes in the wireless sensor network, we find the following two important characteristics. 1) The energy consumption characteristics in data collection is that the nodes near the source node undertake the data forwarding of other nodes far from source nodes, so their energy consumption is much greater than the energy consumption of nodes near the source node. 2) The energy consumption characteristic of code dissemination is that the energy consumption of the entire network has little difference.

If consider the energy consumption of the entire network together, it will find that in the actual network, the energy consumption of nodes far away from the source node is low. Therefore, when the entire network dies, nodes far away from the source node still have remaining energy. If the remaining energy can be used, the delay in the multicast process can be improved. Based on this, we proposes two improvements: One is to increase the transmission power of some nodes in the multicast tree (Adjusting Transmission Power based Code Multicast, ATP-CM algorithm), so as to send data packets to the next hop faster; the second is to increase the active slot (Augmenting Active Slots based Code Multicast, AAS-CM algorithm) of some nodes to reduce the sending waiting time.

Next, we compare the general multicast scheme with ATP-CM and AAS-CM to illustrate the contribution of the two innovations. The detailed comparative analysis is as follows: In the traditional multicast scheme shown in Figure 2, data packets are sent from the source node to all destination nodes along the multicast tree. Assume that a working cycle of nodes contains eight slots, and nodes only wakes up in one time slot. The active slot of the source node is $\{0\}$, which means that the source node is awake in the first slot of the working cycle. Other nodes randomly select a slot as the active slot during the work cycle, see TABLE 3 for the active slot of each node.

Then, based on the initial multicast tree in Figure 2, the slot delays from the source node to each destination node are calculated, as shown in TABLE 4. Note that the slot delay in this section refers to the number of slots waiting for broadcast packets from the source node to the destination node due to active slot differences, and the actual delay needs to be multiplied by the duration of a single slot (specific calculations please see formula 7).

As can be seen from Figure 2 and TABLE 4, the average number of hops is 5.38, and the average slot delay is 24.95. Among them, the destination nodes with the largest delays are N_{52}, N_{53}, N_{57}, delays are 40, 38, and 40 respectively, and the number of hops are all 7.

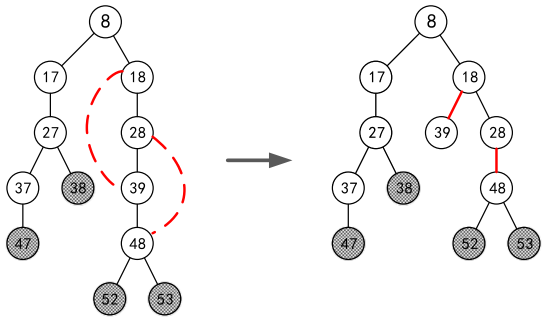

FIGURE 3. Demonstration of the ATP-CM algorithm.

TABLE 3. Active slot of nodes.

$N_1 - N_{10}$	2	4	1	4	1	1	0	3	7	0
$N_{11} - N_{20}$	4	3	1	4	3	7	7	2	7	4
$N_{21} - N_{30}$	5	3	0	0	5	0	7	2	0	4
$N_{31} - N_{40}$	3	3	3	6	3	6	7	5	6	2
$N_{41} - N_{50}$	6	3	0	3	3	7	2	3	2	3
$N_{51} - N_{57}$	2	0	6	4	2	2	0			

TABLE 4. Slot delay of nodes.

	N_{13}	N_{16}	N_{21}	N_{22}	N_{25}	N_{26}	N_{31}
Delay	17	15	13	19	13	16	19
	N_{36}	N_{38}	N_{40}	N_{44}	N_{45}	N_{46}	N_{47}
Delay	14	29	18	27	27	23	34
	N_{51}	N_{52}	N_{53}	N_{54}	N_{55}	N_{56}	N_{57}
Delay	26	40	38	28	34	34	40

First, this paper proposes the Adjusting Transmission Power based Code Multicast (ATP-CM) algorithm to reduce the delay. The core idea is: for nodes with more residual energy in the multicast tree, increase the transmission power of them, thereby changing the topology of the multicast tree and speeding up the broadcast. As shown in Figure 3, the left side is an original multicast tree. After increasing the transmission power of nodes, the new multicast tree on the right is obtained. The height of the tree has changed from 6 to 5, indicating that the number of routing hops has been reduced. The subtree whose root node is N_8 is used to illustrate the idea of this algorithm, that is, the multicast tree on the left. Assume that the child nodes of N_8 contain more residual energy. First, the node N_{18} increases its transmission power, and its multicast child node set is expanded from N_{28} to N_{28}, N_{39}, that is, N_{18} can directly send packets to N_{39} without passing through N_{28}. Similarly, the node N_{28} increases the transmission power, and its multicast child node set is expanded to N_{39}, N_{48}. At this time, N_{39} can receive data from N_{18} and N_{28}, that is, N_{39} has two multicast parent nodes. At this time, the node closer to the source node is selected as the parent node (N_{18}) to reduce the height of the multicast tree and reduce the delay.

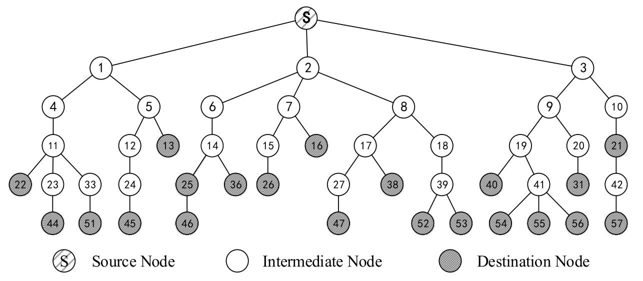

FIGURE 4. Multicast tree after running ATP-CM algorithm.

TABLE 5. Slot delay After ATP-CM algorithm.

	N_{13}	N_{16}	N_{21}	N_{22}	N_{25}	N_{26}	N_{31}
Delay	17	15	13	19	13	16	19
	N_{36}	N_{38}	N_{40}	N_{44}	N_{45}	N_{46}	N_{47}
Delay	14	21	18	19	19	15	26
	N_{51}	N_{52}	N_{53}	N_{54}	N_{55}	N_{56}	N_{57}
Delay	26	24	30	28	26	26	24

Similarly, the ATP-CM algorithm for increasing the transmission power shown in Figure 3 is applied to the multicast tree in Figure 2, and a new multicast tree is obtained, as shown in Figure 4. Note: The redundant intermediate nodes in the figure have been deleted.

Recalculate the slot delay of each destination node in the new multicast tree, and see TABLE 5 for the results. According to statistics, the average slot delay of destination nodes is 20.38, which is 18.3% less than the initial multicast solution. The average hops of destination nodes is 4.38, which is 18.6% less than the initial multicast scheme. The height of the multicast tree is reduced from 8 to 6. For the three most delayed destination nodes N_{52}, N_{53}, N_{57}, the delays are reduced by 40%, 21.1%, and 40% respectively. At the same time, we find that the farther the node is from the source node S, the greater the decrease in delay.

After adopting the algorithm of increasing the transmission power, we further proposed the Augmenting Active Slots based Code Multicast (AAS-CM) algorithm. Focus on the new multicast tree, according to the number of child nodes, the nodes in the tree can be divided into two categories: one is the node with the number of child nodes ≥2 (denoted as $N_{\text{ChildNum}\geq 2}$), and the other is the number of child nodes is 0 or 1 (denoted as $N_{\text{ChildNum}\leq 1}$). The number of child nodes affects the energy consumption of the node, leading to differences in the remaining energy of the two types of nodes. Nodes whose $N_{\text{ChildNum}\geq 2}$ has less remaining energy, and nodes whose $N_{\text{ChildNum}\leq 1}$ has more remaining energy. The idea of AAS-CM algorithm is to implement different active slot adding strategies for two kinds of nodes. For nodes $N_{\text{ChildNum}\geq 2}$, only add a time slot smaller than the active slot of its child nodes to complete data reception and forwarding as soon as possible. For nodes $N_{\text{ChildNum}\leq 1}$, add an active slot for every interval h slots according to the energy surplus.

TABLE 6. Slot delay after AAS-CM algorithm.

	N_{13}	N_{16}	N_{21}	N_{22}	N_{25}	N_{26}	N_{31}
Delay	12	10	11	14	13	11	11
	N_{36}	N_{38}	N_{40}	N_{44}	N_{45}	N_{46}	N_{47}
Delay	12	13	10	17	17	15	16
	N_{51}	N_{52}	N_{53}	N_{54}	N_{55}	N_{56}	N_{57}
Delay	16	16	17	15	16	16	16

Based on the multicast tree of Figure 4, the AAS-CM algorithm is added. Assume the interval $m=2$. The results are shown in TABLE 6.

From TABLE 6, the average slot delay of destination nodes is 14, and the average hops remains unchanged when the AAS-CM algorithm is run. Compared with the result after running ATP-CM algorithm, the delay after AAS-CM algorithm is reduced by 31.3%; compared with the initial multicast scheme, the delay is reduced by 43.9%. At the same time, the lifetime of the entire multicast network remains unchanged.

B. MULTICAST TREE CONSTRUCTION

This paper mainly studies the multicast issue in wireless sensor networks. In a structure where one source node broadcasts packets to multiple destination nodes, the remaining energy is exploited by increasing the node's transmission power and active slots. The content mainly includes three parts. First, build a multicast tree, then increase the node's transmission power, and finally increase the node's active slots.

Assuming that in the initial state of the network, there are m slots in a working cycle, and node N_i randomly selects a slot from the working cycle as the active slot, denoted by $\Gamma(i)$. The transmission radius of each node is r. When the distance between two nodes is less than r, packets can be transmitted directly. If node N_u wants to broadcast data to node N_v, the delay $d(u, v)$ is calculated as follows:

$$d\left(u,v\right)=\left[\left(\Gamma\left(v\right)-\Gamma\left(u\right)+m\right)\text{mode }m\right]*\left(\frac{T_{com}}{m}\right) \quad (7)$$

Among them, $(\Gamma(v)-\Gamma(u)+m)$ mode m is the number of slots that node N_u needs to wait for broadcasting packet to node N_v, and $\frac{T_{com}}{m}$ is the time of each slot.

All nodes and their edges in the network construct a graph $G = (V, E, d)$, where V is the set of nodes, E is the set of edges, and d is the distance measurement function between connected pair nodes, which is the delay calculation function above $d(u, v)$. We use Breadth-First-Search (BFS) method to construct a broadcast tree from the source node to all other nodes. The basic steps are as follows:

1) First, let the source node be the first traversed node, which is called the current node. Put it into the queue, and then traverse all other nodes to see if it is within the communication range of the current node;

2) For nodes within the communication range, judge whether the route established by the current node as a relay

Algorithm 1 Multicast Tree Construction Algorithm

```
Initial: $\{H_1, H_2, \ldots, H_n\} = \infty$, A[m+n+1][m+n+1]=0
1:  $\mathcal{Q}$.push(S)
2:  **While** $\mathcal{Q}$ is not empty **Do**
3:      $N_{tra} = \mathcal{Q}$.front()
4:      $\mathcal{Q}$.pop()
5:      **For** each $N_i$ in V **Do**
6:          Calculate the distance dist($N_{tra}, N_i$)
7:          **If** dist($N_{tra}, N_i$) $\leq r_{tra}$ and $H_{tra} + 1 < H_i$
8:              Update route hops $H_i \leftarrow H_{tra} + 1$
9:              $\mathcal{Q}$.push($N_i$)
10:             A[i][tra]=1
11:         **End if**
12:     **End for**
13: **End while**
```

can reduce the number of hops from the source node to the node, and if the number of hops can be reduced, update the route and hop number of the node. Then add the node whose hop count has just been updated into the queue;

3) When a round of traversal of a node is completed, remove it from the queue;

4) Update the first node in the queue as the current traverse node, and repeat the above process until the queue is empty.

Suppose that the source node is S, $\mathcal{Q}$ is the queue used to traverse the nodes in the process of constructing the multicast tree, $\mathcal{Q}$.push(N_i) indicates the operation of entering queue $\mathcal{Q}$ of node N_i, $\mathcal{Q}$.pop(N_i) indicates the operation of leaving queue $\mathcal{Q}$ of N_i, and $\mathcal{Q}$.front() means to get the first element from $\mathcal{Q}$, N_{tra} is the node currently traversed, H_i means the number of hops passed from the source node to N_i, r_{tra} is the transmission radius of N_{tra}, A[i][tra] means the connection between N_{tra} and N_i, A[i][tra]=1 means there is a route from node N_{tra} to N_i, otherwise A[i][tra]=0. Therefore, the multicast tree construction algorithm can be expressed by Algorithm 1.

C. ATP-CM ALGORITHM

In this section, an Adjusting Transmission Power based Code Multicast (ATP-CM) algorithm is proposed to use the remaining energy to increase the node's transmission power. According to the definition of network lifetime, the node with the largest energy consumption in the network is the node that determines the lifetime of the network. Therefore, in order to enhance energy utilization while ensuring the lifetime, we use the difference between the energy consumption of current node and the largest energy consumption in the network as the remaining energy. The ATP-CM algorithm is mainly implemented through the following three steps. (1) Calculate the consumed energy E_{tot}; (2) Calculate the remaining energy E_{rest}; (3) Based on the initial transmission power r_{ini}, use the remaining energy E_{rest} to increase the transmission power of nodes, and obtain the new transmission power r_{new}.

1) ENERGY CONSUMPTION CALCULATION MODEL

The energy consumption of a node includes the consumption of data transmission, data reception and low-power listening, which are represented by E_{tran}, E_{rec} and E_{LPL}. Due to the data broadcast structure of the network centered on the source node, the nodes closer to the source node have more energy consumption due to the greater amount of data forwarding. The consumption E^x_{tot} of N_i can be expressed as:

$$E^x_{tot} = E^x_{tran} + E^x_{rec} + E^x_{LPL} \tag{8}$$

E^x_{tran} indicates the energy consumption of the node in data transmission, and its calculation is as follows:

$$E^x_{tran} = E^{one}_t \Pi^x_t \tag{9}$$
$$E^{one}_t = \varepsilon_t t_d + \varepsilon_t t_{pre} + \varepsilon_r t_{ack} \tag{10}$$

where E^{one}_t is the energy consumption of sending a data packet, and its calculation is shown in formula (10), including transmission data consumption $\varepsilon_t t_d$, transmission preamble consumption $\varepsilon_t t_{pre}$, and reception confirmation message consumption $\varepsilon_r t_{ack}$. Π^x_t is the amount of data transmitted by a node in a cycle.

E^x_{rec} represents the energy consumption of the node in data reception, and its calculation is as follows:

$$E^x_{rec} = E^{one}_r \Pi^x_r \tag{11}$$
$$E^{one}_r = \varepsilon_r t_d + \varepsilon_r t_{pre} + \varepsilon_t t_{ack} \tag{12}$$

where E^{one}_r is the energy consumption of the node receiving a data packet, and its calculation is as shown in formula (12), including receiving data consumption $\varepsilon_r t_d$, receiving preamble consumption $\varepsilon_t t_{pre}$, transmission confirmation message consumption $\varepsilon_r t_{ack}$. Π^x_r is the amount of received data in a cycle.

E^x_{LPL} indicates the energy consumption of the node in low-power listening, which is calculated as follows:

$$E^x_{LPL} = [\varepsilon_r \phi_{com} + \varepsilon_s (1 - \phi_{com})] T_{com} \tag{13}$$

where $\varepsilon_r \phi_{com}$ is the receiving consumption of the node in the listening state, $\varepsilon_s (1 - \phi_{com})$ is the sleep consumption of the node in the listening state, ϕ_{com} is the communication duty cycle, and T_{com} is the time of a communication cycle.

2) RESIDUAL ENERGY CALCULATION MODEL

In a duty-cycle based wireless sensor network, the node that is close to the source node bears heavier data load and greater energy consumption. Assume that the energy consumption of the node with the largest energy consumption in the network is E_{max}. In order not to affect the lifetime of the network, the remaining energy in this paper refers to the energy consumption difference of other nodes compared to the node with the largest energy consumption. Assume that the residual energy of the node N_i is E^x_{rest}, its calculation is as follows:

$$E^x_{rest} = E^{one}_t \left(\Pi^{max}_t - \Pi^x_t\right) + E^{one}_r \left(\Pi^{max}_r - \Pi^x_r\right) \tag{14}$$

The proof of formula (14) is as follows. First, according to formula (8), the energy consumption of the node with the largest energy consumption is obtained: $E_{max} = E_t^{one}\Pi_t^{max} + E_r^{one}\Pi_r^{max} + [\varepsilon_r\phi_{com} + \varepsilon_s(1-\phi_{com})]T_{com}$. In the same way, the energy consumption of the node N_i can be obtained: $E_{tot}^x = E_t^{one}\Pi_t^x + E_r^{one}\Pi_r^x + [\varepsilon_r\phi_{com} + \varepsilon_s(1-\phi_{com})]T_{com}$. Therefore, the remaining energy of the node N_i can be obtained: $E_{rest}^x = E_{max} - E_{tot}^x = (E_t^{one}\Pi_t^{max} + E_r^{one}\Pi_r^{max} + [\varepsilon_r\phi_{com} + \varepsilon_s(1-\phi_{com})]T_{com}) - (E_t^{one}\Pi_t^x + E_r^{one}\Pi_r^x + [\varepsilon_r\phi_{com} + \varepsilon_s(1-\phi_{com})]T_{com}) = E_t^{one}(\Pi_t^{max} - \Pi_t^x) + E_r^{one}(\Pi_r^{max} - \Pi_r^x)$.

3) INCREASE THE TRANSMISSION POWER OF NODES

After the energy consumption and remaining energy of nodes are known, the new transmission power adjusted on the basis of the initial transmission power can be obtained. The calculation is as follows:

$$r_{new} = r_{ini} + \frac{E_{rest}}{E_\gamma} \tag{15}$$

where r_{ini} is the initial transmission power of nodes, and E_γ is the energy consumed by increase transmission power by adding γ length.

At regular intervals, nodes adjust the transmission power. After the adjustment, the multicast tree structure of the entire network changes. Obviously, when the node's transmission power increases, its broadcast range is wider, and the number of hops required to go through for the packet transmission is less, which can reduce the delay.

Assuming the set of all nodes in the network is V, the time interval for nodes perform transmission power adjustment is T_σ, and at each adjustment, nodes increase the transmission power γ. When the remaining energy of the node is lower than E_{rest}^Θ, no transmission power adjustment is performed. For the node N_i, its consumed energy is E_{tot}^i, the remaining energy relative to the maximum energy consumption E_{max} is E_{rest}^i, and the initial transmission power of the node is r_{ini}. Then the Adjusting Transmission Power based Code Multicast algorithm is shown as Algorithm 2.

D. AAS-CM AlGORITHM

For the multicast tree structure shown in Figure 2, increasing the transmission power of intermediate nodes close to the source node can speed up the broadcast, but for the destination nodes and intermediate nodes close to the destination node, the distance they need to broadcast is within one hop. Therefore, increasing the transmission power has little effect. Then, on the basis of increasing the transmission power, we propose an Augmenting Active Slots based Code Multicast (AAS-CM) algorithm to add the active slot of nodes. The detailed rules are as follows:

1) For nodes with $N_{\text{ChildNum}\geq 2}$, only one active slot is added, which is smaller than the active slot of the current node and child nodes. Its purpose is to let the node receive the data packet as early as possible, and at the same time can send it to the child node in the current working cycle.

Algorithm 2 Adjusting Transmission Power Based Code Multicast Algorithm

```
Initial: $E_{max} = 0$
1:  **For** each time interval $\mathcal{T}_\sigma$ **Do**
2:      **For** each $N_i$ in V **Do**
3:          Compute energy $E_{tot}^i$ consumed by $N_i$ with Eq. (8)
4:      **If** $E_{tot}^i > E_{max}$
5:          $E_{max} \leftarrow E_{tot}^i$
6:      **End if**
7:      **End for**
8:      **For** each $N_i$ in V **Do**
9:          Compute rest energy $E_{rest}^i$ of $N_i$ with Eq. (14)
10:         **While** $E_{rest}^i > E_{rest}^\Theta$ **Do**
11:             Update transmission power $r_{new}^i$ with Eq. (15)
12:             Update rest energy, $E_{rest}^i \leftarrow E_{rest}^i - E_\gamma$
13:         **End while**
14:     **End for**
15: **End for**
```

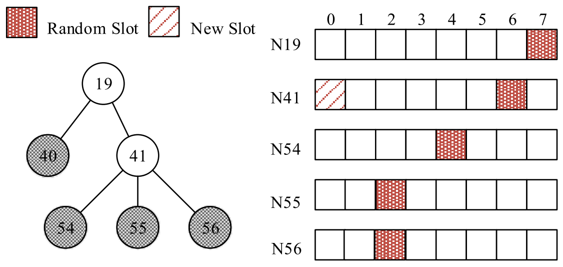

FIGURE 5. Adding active slot of nodes with $N_{\text{ChildNum}\geq 2}$.

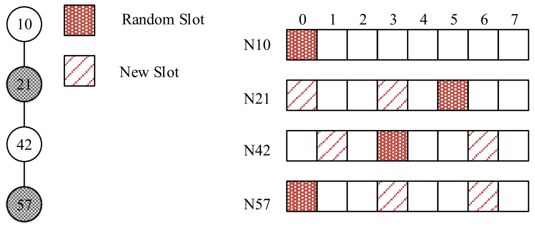

FIGURE 6. Adding active slot of nodes with $N_{\text{ChildNum}\geq 1}$.

2) For nodes with $N_{\text{ChildNum}\leq 1}$, take the randomly generated active slot as the starting point, and add an active slot every interval of h slots.

Next, Figure 5 and Figure 6 are taken as examples to illustrate how nodes with different numbers of child nodes add active slots.

As shown in Figure 5, for the node N_{41}, it has three child nodes, namely N_{54}, N_{55}, N_{56}, and the active time slot of them are 6, 4, 2, 2}. If we want to add an active slot for the node N_{41}, the new active slot should smaller than active slot of N_{54}, N_{55}, N_{56}, which can be the slot indexed by 0 or 1. Suppose the new active slot of N_{41} is 0, at this time, N_{41} can receive the packet sent by N_{19} earlier. This also enables the three child nodes N_{54}, N_{55} and N_{56} to receive packet in the same working cycle of N_{41}, reducing the delay. At this

Algorithm 3 Augmenting Active Slots Based Code Multicast Algorithm

```
Initial: $I_{slot} = $ m
1:  For each time period $\mathcal{T}_\sigma$ Do
2:    For each $N_i$ in V and $E^i_{rest} > E^{one}_{rec}$ Do
3:      If $|\mathbb{C}_i| \geq 2$
4:        For each $N_k$ in $\mathbb{C}_i$ Do
5:          If $I_{slot} > MinI^k_{slot}$ then
6:            $I_{slot} \leftarrow MinI^k_{slot}$
7:          End if
8:        End for
9:        $I^i_{new} \leftarrow (I_{slot} - 1 + m) \bmod m$
10:       Add $I^i_{new}$ in $\mathbb{I}^i_{slot}$
11:       Update rest energy, $E^i_{rest} \leftarrow E^i_{rest} - E^{one}_{rec}$
12:     End if
13:     Else
14:       While $E^i_{rest} > E_\Theta$ and $\left|\mathbb{I}^i_{slot}\right| < \left\lceil \frac{m}{h} \right\rceil$ Do
15:         $I^i_{new} \leftarrow \left(I^i_{pre} + h + 1\right) \bmod m$
16:         Add $I^i_{new}$ in $\mathbb{I}^i_{slot}$
17:         Update rest energy, $E^i_{rest} \leftarrow E^i_{rest} - E^{one}_{rec}$
18:         Update active slot, $I^i_{pre} \leftarrow I^i_{new}$
19:       End while
20:     End else
21:   End for
22: End for
```

time ($Delay\ (N_{19}) = 15$), the slot delays of $N_{41}, N_{54}, N_{55}, N_{56}$ are 16, 20, 18, and 18, respectively, and ruduce the delay by 27.3%, 28.6%, 30.8%, and 30.8% respectively.

Figure 6 shows the algorithm for increasing the active slot of nodes with $N_{\text{ChildNum}\leq 1}$. Assuming $h = 2$, we first adjust the active slot of the node N_{21}, starting from the initial active slot {5} and add active slot with the interval $h = 2$, and get the active slot set of node N_{21} as {5,0,3}. Similarly, N_{42} and N_{57} also use this method to add active slots. The result is shown in Figure 6. At this time ($Delay\ (N_{10}) = 8$), the delays of N_{21}, N_{42}, and N_{57} are 11, 14, and 16, respectively, and the delay is reduced by 15.4%, 26.3%, and 33.3%, respectively.

Since the energy consumption of nodes in the active state is several times that of the sleep state, the increase of the above active slots is based on the sufficient remaining energy. At the beginning of the network, there is only one active slot for each node. Assuming that the network data generation rate remains unchanged, based on the initial energy consumption of the node, the energy consumed by adding an active slot can be calculated E^{one}_{rec}. Assuming that the current node knows the active slots information of its child nodes, for the nodes with $N_{\text{ChildNum}\geq 2}$, when the remaining energy E^i_{rest} is greater than E^{one}_{rec}, an active slot is added for the current node. For the nodes with $N_{\text{ChildNum}\leq 1}$, the active slots are also added according to the above process until the number of active slots reaches the requirements, or there is not enough remaining energy.

Suppose the child nodes construct the set $\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \ldots\}$, where $\mathbb{C}_i$ is the set of child nodes of the node N_i, and the set of active slots of N_i is I^i_{slot}, the initial generated active slot of N_i is I^i_{pre}, the new added active slot is I^i_{new}. Meanwhile, a communication cycle is divided into m time slots. For each time interval $\mathcal{T}_\sigma$, nodes adjust their active slots. For nodes with $N_{\text{ChildNum}\geq 2}$, it increases an active slot samaller than its child nodes. For nodes with $N_{\text{ChildNum}\leq 1}$, starting from the initial active slot, a new active slot is added every m slots. When the remaining energy is below E_Θ, no more active slot is added. Then Augmenting Active Slots based Code Multicast algorithm is shown in Algorithm 3.

V. PERFORMANCE ANALYSIS

A. EXPERIMENTAL SETUP

The experimental scene consists of a source node, 500 intermediate nodes and 60 destination nodes. The network is centered on the source node, and other nodes are randomly distributed within a planar network 500 meters away from the source node. The initial transmission radius of each node is 60 meters, and a communication cycle of the node is equally divided into 8 time slots, m=8. Each node randomly selects one of the slots as its active slot, in which data can be received and forwarded, and sleeps in other slots, so the node's duty cycle is initially 0.125. In the experiment, the intermediate nodes are numbered from 1 to 500, and the destination nodes are numbered from 1 to 60 to compare the performance under different schemes. Meanwhile, we uses the breadth-first search algorithm to construct the multicast tree, and other experimental parameters are given in Table 1 and Table 2.

There are three comparison metrics: 1) energy consumption of intermediate nodes; 2) delay of destination nodes; and 3) network lifetime. At the same time, three schemes related to this work are selected for comparason. The first is the general scheme with Fixed Transmission radius and Active slots (FTA). In FTA, the transmission power and active slots of nodes are configured when the network is initialized, and remain fixed during network operation. The second is scheme with Adding Transmission radius and Fixed Active slots (ATFA). In ATFA, the awake slots of nodes are always fixed, but the transmission radius can be dynamically adjusted according to the remaining energy. Similar to ATFA, the third is scheme with Fixed Transmission radius and Adding Active slots (FTAA). In FTAA, the transmission power of nodes is always fixed, but the awake slots can be increased according to the remaining energy. Our scheme is called Adjusting Transmission Power and Augmenting Active Slots based Code Multicast (ATP-AAS-CM) scheme.

B. ENERGY

Figure 7 shows the energy consumption of 500 intermediate nodes under each scheme. The figure first makes a general comparison of the four schemes, and then compares the scheme proposed in this article with other schemes seperately.

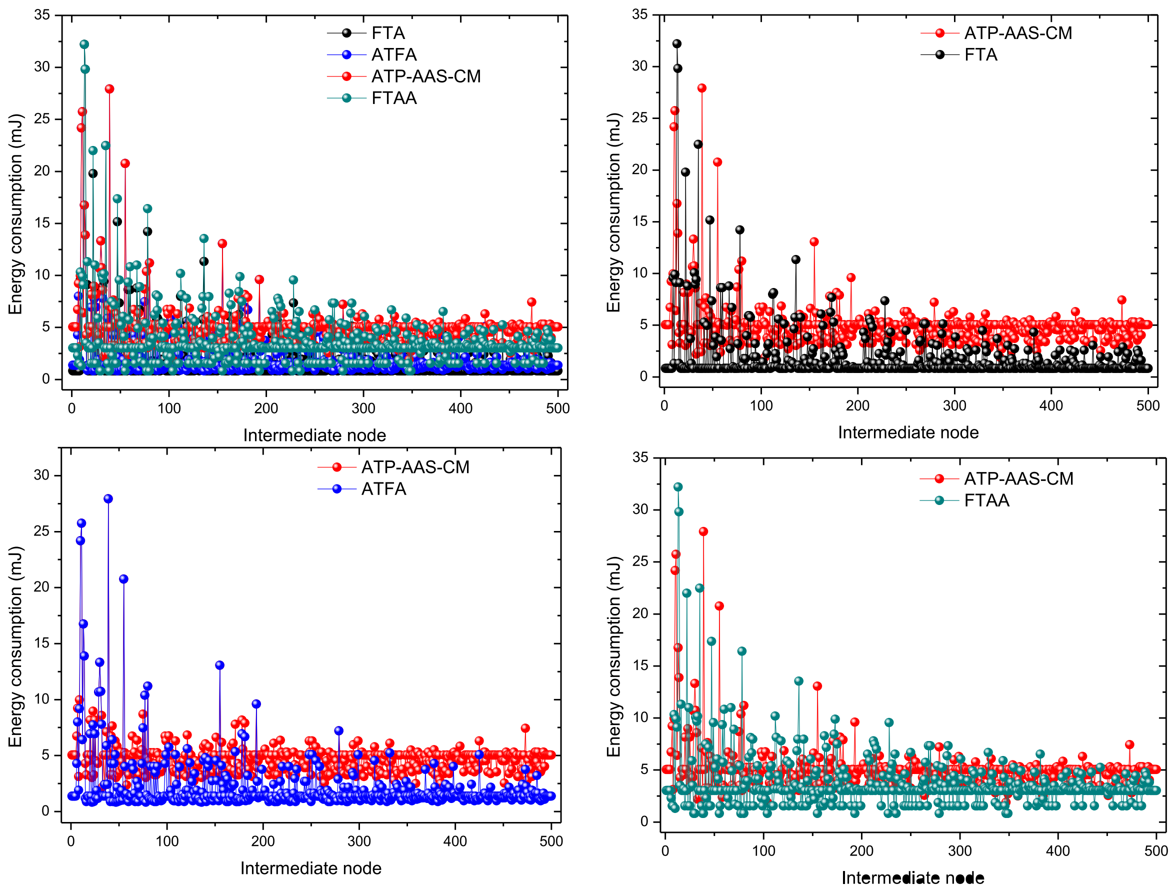

FIGURE 7. Energy consumption of intermediate nodes under the four schemes.

Overall, the average energy consumption is more under ATP-AAS-CM, followed by FTAA, and finally FTA and ATFA. Therefore, it can be concluded that increasing the active slots consumes more energy than increasing the transmission power. The ATP-AAS-CM further increases active slots on the basis of increasing transmission power, so nodes consume more energy.

According to the comparison between ATP-AAS-CM and FTA, the energy consumption of most nodes in ATP-AAS-CM is significantly higher than FTA. The energy consumption of nodes in FTA has a greater difference. Some nodes have small energy consumption, while the energy consumption of some nodes is very high. The energy consumption of nodes in ATP-AAS-CM is relatively balanced. According to the comparison between ATP-AAS-CM and ATFA, the energy consumption of nodes under the two schemes is relatively balanced, and the overall energy consumption in ATP-AAS-CM is higher than ATFA. According to the comparison between ATP-AAS-CM and FTAA, the energy consumption of nodes under FTAA is very different, and the largest consumption of nodes is much higher than ATP-AAS-CM. Therefore, it can be concluded that compared with the other three schemes, ATP-AAS-CM can make the energy consumption of nodes more balanced, and the maximum energy consumption is lower than other schemes.

Figure 8 is the remaining energy of intermediate nodes under each scheme. Overall, FTA and ATFA have the most remaining energy, and ATP-AAS-CM have the least remaining energy. Separately, there are very few nodes under FTA that have very little remaining energy, which is about 30% of the remaining energy of other nodes, while the energy of the least remaining energy node under ATP-AAS-CM is about 50% of other nodes. The same phenomenon can be seen from ATFA and FTAA. Therefore, we can conclude that the remaining energy of the nodes under ATP-AAS-CM is the least, and the difference in the remaining energy of the nodes is small, which shows that the energy of the nodes under ATP-AAS-CM is effectively used and consumed evenly.

The energy utilization rate refers to the ratio of the energy consumed by nodes to the initial energy when the network is dead. As shown in Figure 9, the energy utilization rates of FTA, ATFA, ATP-AAS-CM and FTAA are 13.429%, 15.399%, 34.175%, and 26.045%, respectively. Therefore, the energy utilization rate of ATP-AAS-CM is the highest, which is an increase of about 8%-21% compared to other schemes.

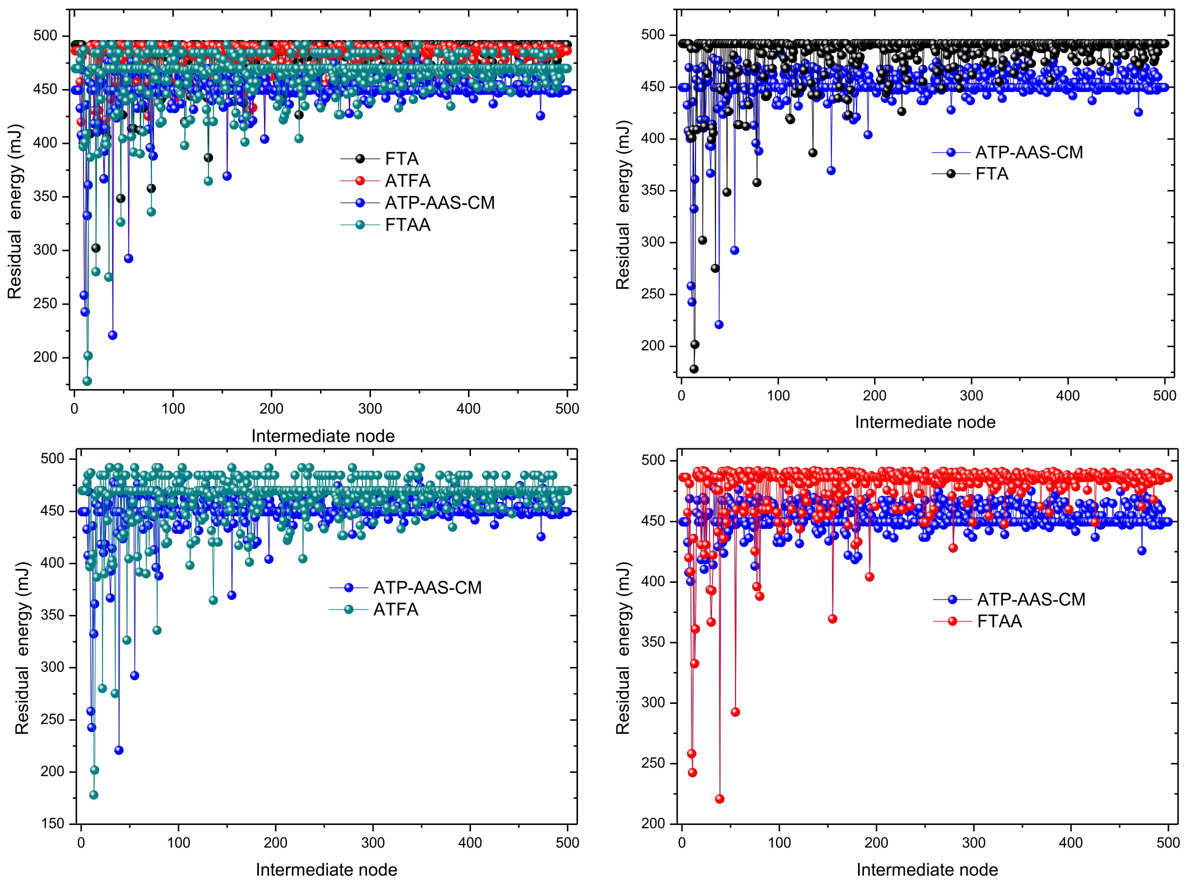

FIGURE 8. Remaining energy of intermediate nodes under the four schemes.

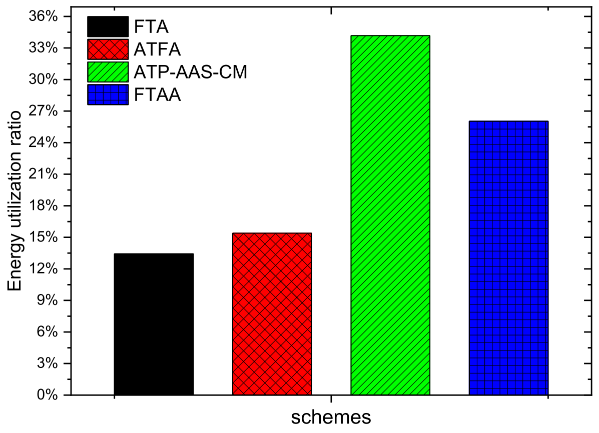

FIGURE 9. Energy utilization ratio under the four schemes.

C. DELAY

After the multicast tree is established, the number of hops to pass from the source node to the 60 destination nodes is shown in Figure 10. FTA and FTAA are obtained based on the initial transmission radius r=60. ATFA and ATP-AAS-CM are obtained by using the remaining energy to increase the transmission power. Obviously, increasing the transmission power of nodes can make the single-hop transmission distance longer, so there are fewer hops for data from the source node to the destination node after increasing the transmission power. The average number of hops under FTA and FTAA is 11.25, and the average number of hops under ATFA and ATP-AAS-CM is 6.82. ATP-AAS-CM reduces the number of hops by 39.38%.

Figure 11 shows the delay from the source node to the destination node under the four schemes. The delay is not only related to the number of hops, but also related to the active slots of the node. If the node is in a sleep state, it will not receive and forward data until it is active. It can be seen from the figure that the delay of FTAA is the smallest, because it uses all the remaining energy to increase the active slots, so the waiting time of the node at each hop is reduced, and the delay is also reduced. Secondly, the delay of ATP-AAS-CM is lower. The biggest delay is FTA. It can be concluded from Figure 11 that increasing the node transmission power and active slots can effectively reduce the delay, but increasing the active slots has a more obvious impact on the delay.

Figure 12 is the average delay of nodes. The delays of FTA, ATFA, ATP-AAS-CM and FTAA are 580.2083 ms, 291.875 ms, 131.25 ms, and 105.4167 ms respectively. Therefore, compared with FTA, ATP-AAS-CM reduces latency by 77.38%, and compared with ATFA, ATP-AAS-CM reduces latency by 63.88%.

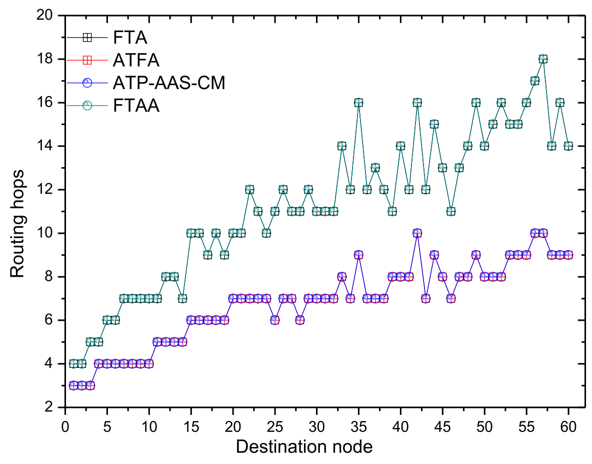

FIGURE 10. Number of hops from the source node to the destination node.

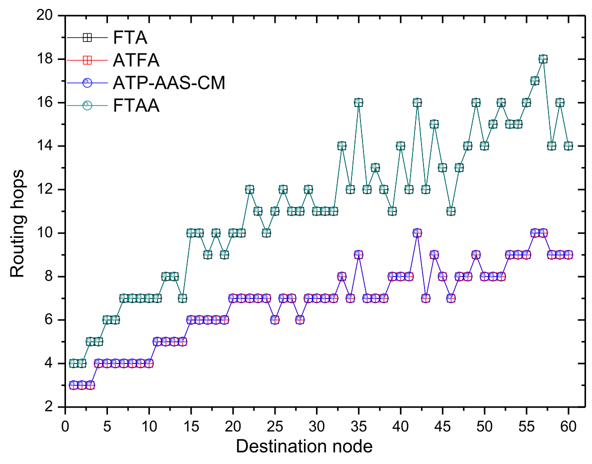

FIGURE 11. Delay from source node to the destination node.

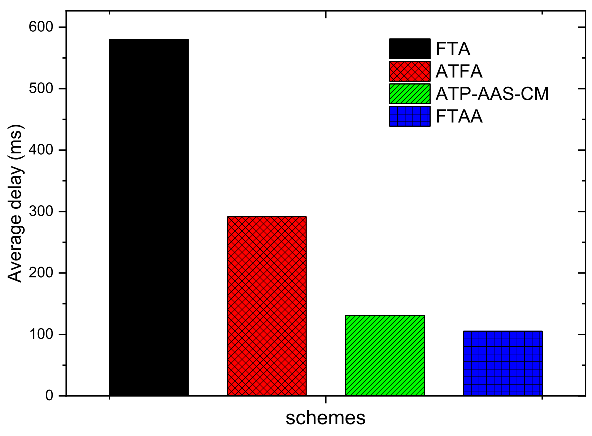

FIGURE 12. Average delay under the four schemes.

D. LIFETIME

Figure 13 is the network lifetime of the four schemes, which depends on the maximum energy consumption of the node. FTA and FTAA have a shorter network lifetime due to the large energy consumption of nodes. Energy consumption of nodes in ATFA and ATP-AAS-CM is balanced, and the maximum energy consumption is smaller, so the network lifetime is long.

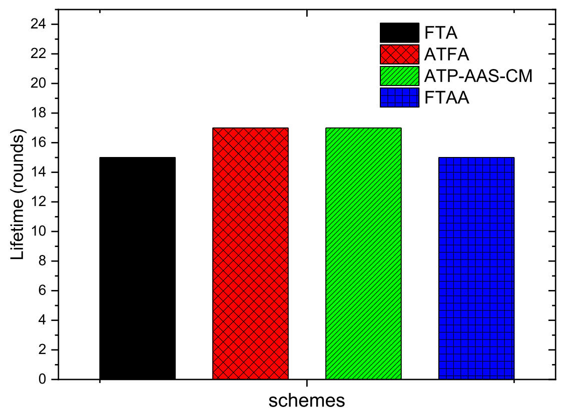

FIGURE 13. Network lifetime under the four schemes.

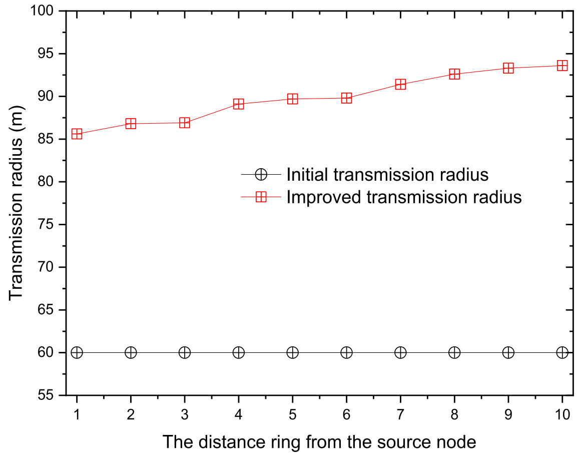

FIGURE 14. Transmission radius of nodes before and after improvement.

E. OTHERS

Figure 14 shows the transmission radius of nodes after the improved algorithm in this paper. Assuming that a circle is divided by 50 meters, the figure shows the average transmission radius of nodes from 1 to 10 rings from the source node. The initial transmission radius of all nodes is 60. After a period of time, nodes with remaining energy increase the transmission radius. Since the farther the node is from the source node, the more remaining energy it has, so the transmission radius that can be increased is larger. After the improvement, the transmission radius of the node is more than 1.42 times the initial radius.

The duty cycle of a node is the ratio of active slots to the entire communication cycle. After using the remaining energy to increase the active slot, the duty cycle of the node will also increase. Figure 15 is the improved duty cycle. At the beginning of the network, a communication cycle of a node is divided into 8 slots, and each node randomly selects a slot

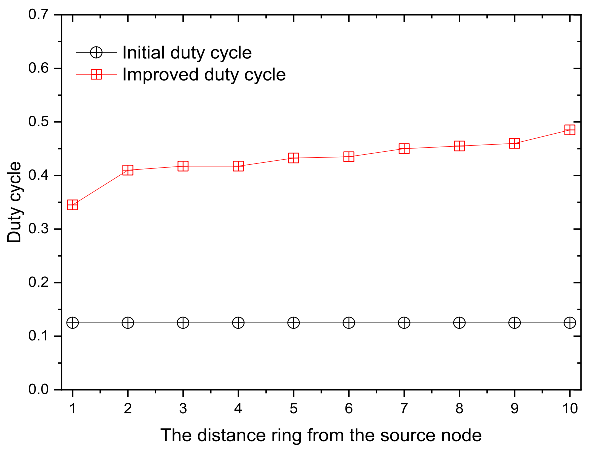

FIGURE 15. **Duty cycle of nodes before and after improvement.**

as the active slot, so the initial duty cycle is 0.125. After the active slot is increased by the remaining energy, the duty cycle gradually increases. Similarly, the farther the node is from the source node, the more remaining energy will be used to increase the active slot, so the larger the duty cycle. The improved duty cycle is at least 2.8 times higher than the initial duty cycle.

VI. CONCLUSION

A novel fast multicast solution is proposed in this article, including two improvements: one is to increase the transmission power of nodes in the multicast tree, so as to send data packets to the next hop faster; the second is to increase the active slot of some nodes to reduce the sending waiting time. The specific realization is to first construct a multicast tree from the source node to each destination node, and then according to the energy consumption of the node in the multicast process, make full use of the remaining energy to increase the transmission power of the node to reduce the number of routing hops. On this basis, the remaining energy is further used to increase active slots to reduce delay. Through the analysis of experimental results, it is found that our proposed scheme is better than others. These two improvements can effectively enhance energy utilization and reduce delay.

REFERENCES

[1] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, May 2019.

[2] S. Huang, Z. Zeng, K. Ota, M. Dong, T. Wang, and N. Xiong, "An intelligent collaboration trust interconnections system for mobile information control in ubiquitous 5G networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Nov. 17, 2020, doi: 10.1109/TNSE.2020.3038454.

[3] X. Zhu, Y. Luo, A. Liu, W. Tang, and M. Z. A. Bhuiyan, "A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 7, 2020, doi: 10.1109/TITS.2020.3023446.

[4] S. Huang, A. Liu, S. Zhang, T. Wang, and N. Xiong, "BD-VTE: A novel baseline data based verifiable trust evaluation scheme for smart network systems," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 7, 2020, doi: 10.1109/TNSE.2020.3014455.

[5] M. Huang, K. Zhang, Z. Zeng, T. Wang, and Y. Liu, "An AUV-assisted data gathering scheme based on clustering and matrix completion for smart ocean," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9904–9918, Oct. 2020.

[6] B. Jiang, G. Huang, T. Wang, J. Gui, and X. Zhu, "Trust based energy efficient data collection with unmanned aerial vehicle in edge network," *Trans. Emerg. Telecommun. Technol.*, Mar. 2020, Art. no. e3942, doi: 10.1002/ett.3942.

[7] Y. Ren, Z. Zeng, T. Wang, S. Zhang, and G. Zhi, "A trust-based minimum cost and quality aware data collection scheme in P2P network," *Peer Peer Netw. Appl.*, vol. 13, no. 6, pp. 2300–2323, Nov. 2020, doi: 10.1007/s12083-020-00898-2.

[8] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in IoT," *IEEE Internet Things J.*, early access, Nov. 26, 2020, doi: 10.1109/JIOT.2020.3040768.

[9] Y. Liu, M. Ma, X. Liu, N. N. Xiong, A. Liu, and Y. Zhu, "Design and analysis of probing route to defense sink-hole attacks for Internet of Things security," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 356–372, Jan. 2020.

[10] M. Yu, A. Liu, N. N. Xiong, and T. Wang, "An intelligent game based offloading scheme for maximizing benefits of IoT-Edge-Cloud ecosystems," *IEEE Internet Things J.*, early access, Nov. 23, 2020, doi: 10.1109/JIOT.2020.3039828.

[11] X. Liu, A. Liu, T. Wang, K. Ota, M. Dong, Y. Liu, and Z. Cai, "Adaptive data and verified message disjoint security routing for gathering big data in energy harvesting networks," *J. Parallel Distrib. Comput.*, vol. 135, pp. 140–155, Jan. 2020.

[12] T. Wang, Y. Liang, Y. Yang, G. Xu, H. Peng, A. Liu, and W. Jia, "An intelligent edge-computing-based method to counter coupling problems in cyber-physical systems," *IEEE Netw.*, vol. 34, no. 3, pp. 16–22, May 2020.

[13] X. Liu, J. Yin, S. Zhang, B. Xiao, and B. Ou, "Time-efficient target tags information collection in large-scale RFID systems," *IEEE Trans. Mobile Comput.*, early access, May 4, 2020, doi: 10.1109/TMC.2020.2992256.

[14] Y. Liu, Z. Zeng, X. Liu, X. Zhu, and M. Z. A. Bhuiyan, "A novel load balancing and low response delay framework for edge-cloud network based on SDN," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5922–5933, Jul. 2020.

[15] W. Mo, T. Wang, S. Zhang, and J. Zhang, "An active and verifiable trust evaluation approach for edge computing," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–19, Dec. 2020.

[16] T. Wang, H. Luo, X. Zeng, Z. Yu, A. Liu, and A. K. Sangaiah, "Mobility based trust evaluation for heterogeneous electric vehicles network in smart cities," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 19, 2020, doi: 10.1109/TITS.2020.2997377.

[17] H. Liao, Z. Zhou, W. Kong, Y. Chen, X. Wang, Z. Wang, and S. A. Otaibi, "Learning-based intent-aware task offloading for air-ground integrated vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 29, 2020, doi: 10.1109/TITS.2020.3027437.

[18] Z. Zhou, H. Yu, C. Xu, Z. Chang, S. Mumtaz, and J. Rodriguez, "BEGIN: Big data enabled energy-efficient vehicular edge computing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 82–89, Dec. 2018.

[19] O. Yan, A. Liu, N. Xiong, and T. Wang, "An effective early message ahead join adaptive data aggregation scheme for sustainable IoT," *IEEE Trans. Netw. Sci. Eng.*, early access, Oct. 29, 2020, doi: 10.1109/TNSE.2020.3033938.

[20] J. Luo, X. Deng, H. Zhang, and H. Qi, "QoE-driven computation offloading for edge computing," *J. Syst. Archit.*, vol. 97, pp. 34–39, Aug. 2019.

[21] J. Wang, F. Wang, Y. Wang, L. Wang, Z. Qiu, D. Zhang, B. Guo, and Q. Lv, "HyTasker: Hybrid task allocation in mobile crowd sensing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 3, pp. 598–611, Mar. 2020.

[22] H. Teng, Y. Liu, A. Liu, N. N. Xiong, Z. Cai, T. Wang, and X. Liu, "A novel code data dissemination scheme for Internet of Things through mobile vehicle of smart cities," *Future Gener. Comput. Syst.*, vol. 94, pp. 351–367, May 2019.

[23] J. Gui, L. Hui, N. N. Xiong, and J. Wu, "Improving spectrum efficiency of cell-edge devices by incentive architecture applications with dynamic charging," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 795–808, Feb. 2021, doi: 10.1109/TII.2020.2987089.

[24] N. Zhang, H. Liang, N. Cheng, Y. Tang, J. W. Mark, and X. Sherman Shen, "Dynamic spectrum access in multi-channel cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 11, pp. 2053–2064, Nov. 2014.

[25] X. Deng, J. Li, E. Liu, and H. Zhang, "Task allocation algorithm and optimization model on edge collaboration," *J. Syst. Archit.*, vol. 110, Nov. 2020, Art. no. 101778.

[26] Q. Xu, Z. Su, and Q. Yang, "Blockchain-based trustworthy edge caching scheme for mobile cyber-physical system," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1098–1110, Feb. 2020.

[27] W. Huang, K. Ota, M. Dong, T. Wang, S. Zhang, and J. Zhang, "Result return aware offloading scheme in vehicular edge networks for IoT," *Comput. Commun.*, vol. 164, pp. 201–214, Dec. 2020, doi: 10.1016/j.comcom.2020.10.019.

[28] Y. Liu, T. Wang, S. Zhang, X. Liu, and X. Liu, "Artificial intelligence aware and security-enhanced traceback technique in mobile edge computing," *Comput. Commun.*, vol. 161, pp. 375–386, Sep. 2020.

[29] H. Teng, K. Ota, A. Liu, T. Wang, and S. Zhang, "Vehicles joint UAVs to acquire and analyze data for topology discovery in large-scale IoT systems," *Peer Peer Netw. Appl.*, vol. 13, no. 5, pp. 1720–1743, Sep. 2020.

[30] T. Wang, Z. Cao, S. Wang, J. Wang, L. Qi, A. Liu, M. Xie, and X. Li, "Privacy-enhanced data collection based on deep learning for Internet of vehicles," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6663–6672, Oct. 2020.

[31] K. Xie, X. Ning, X. Wang, D. Xie, J. Cao, G. Xie, and J. Wen, "Recover corrupted data in sensor networks: A matrix completion solution," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1434–1448, May 2017.

[32] J. Gui, X. Dai, and X. Deng, "Stabilizing transmission capacity in millimeter wave links by Q-learning-based scheme," *Mobile Inf. Syst.*, vol. 2020, pp. 1–17, Feb. 2020.

[33] J. Ge, B. Liu, T. Wang, Q. Yang, A. Liu, and A. Li, "Q-learning based flexible task scheduling in a global view for the Internet of Things," *Trans. Emerg. Telecommun. Technol.*, Sep. 2020, Art. no. e4111, doi: 10.1002/ETT.4111.

[34] Q. Liu, Y. Tian, J. Wu, T. Peng, and G. Wang, "Enabling verifiable and dynamic ranked search over outsourced data," *IEEE Trans. Services Comput.*, early access, Jun. 11, 2019, doi: 10.1109/TSC.2019.2922177.

[35] X. Liu, P. Lin, T. Liu, T. Wang, A. Liu, and W. Xu, "Objective-variable tour planning for mobile data collection in partitioned sensor networks," *IEEE Trans. Mobile Comput.*, early access, Jun. 17, 2020, doi: 10.1109/TMC.2020.3003004.

[36] S. Liu, G. Huang, J. Gui, T. Wang, and X. Li, "Energy-aware MAC protocol for data differentiated services in sensor-cloud computing," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–33, Dec. 2020.

[37] L. Cheng, J. Niu, C. Luo, L. Shu, L. Kong, Z. Zhao, and Y. Gu, "Towards minimum-delay and energy-efficient flooding in low-duty-cycle wireless sensor networks," *Comput. Netw.*, vol. 134, pp. 66–77, Apr. 2018.

[38] H. Liao, Y. Mu, Z. Zhou, M. Sun, Z. Wang, and C. Pan, "Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing," *IEEE Trans. Intell. Transp. Syst.*, early access, Jul. 21, 2020, doi: 10.1109/TITS.2020.3007770.

[39] Y. Gu, T. He, M. Lin, and J. Xu, "Spatiotemporal delay control for low-duty-cycle sensor networks," in *Proc. 30th IEEE Real-Time Syst. Symp.*, Dec. 2009, pp. 127–137.

[40] J. Hong, J. Cao, W. Li, S. Lu, and D. Chen, "Minimum-transmission broadcast in uncoordinated duty-cycled wireless ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 307–318, Jan. 2010.

[41] X. Liu, G. Li, S. Zhang, and A. Liu, "Big program code dissemination scheme for emergency software-define wireless sensor networks," *Peer Peer Netw. Appl.*, vol. 11, no. 5, pp. 1038–1059, Sep. 2018.

[42] X. Liu, A. Liu, Q. Deng, and H. Liu, "Large-scale programing code dissemination for software-defined wireless networks," *Comput. J.*, vol. 60, no. 10, pp. 1417–1442, Oct. 2017.

[43] Y. Gu and T. He, "Dynamic switching-based data forwarding for low-duty-cycle wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 12, pp. 1741–1754, Dec. 2011.

[44] J. Gu, X.-D. Hu, and M.-H. Zhang, "Algorithms for multicast connection under multi-path routing model," *Inf. Process. Lett.*, vol. 84, no. 1, pp. 31–39, Oct. 2002.

[45] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta Inf.*, vol. 15, no. 2, pp. 141–145, 1981.

[46] Q. Chen, H. Gao, S. Cheng, X. Fang, Z. Cai, and J. Li, "Centralized and distributed delay-bounded scheduling algorithms for multicast in duty-cycled wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3573–3586, Dec. 2017.

[47] M. Bonola, L. Bracciale, P. Loreti, R. Amici, A. Rabuffi, and G. Bianchi, "Opportunistic communication in smart city: Experimental insight with small-scale taxi fleets as data carriers," *Ad Hoc Netw.*, vol. 43, pp. 43–55, Jun. 2016.

[48] Y. Zhao, T. Wang, S. Zhang, and Y. Wang, "Towards minimum code dissemination delay through UAV joint vehicles for smart city," *IET Commun.*, vol. 14, no. 15, pp. 2442–2452, Sep. 2020.

[49] S. Huang, J. Gui, T. Wang, and X. Li, "Joint mobile vehicle–UAV scheme for secure data collection in a smart city," *Ann. Telecommun.*, pp. 1–22, Aug. 2020, doi: 10.1007/s12243-020-00798-9.

[50] Y. Liu, A. Liu, S. Guo, Z. Li, Y.-J. Choi, and H. Sekiya, "Context-aware collect data with energy efficient in Cyber–physical cloud systems," *Future Gener. Comput. Syst.*, vol. 105, pp. 932–947, Apr. 2020.

[51] L. Hu, A. Liu, M. Xie, and T. Wang, "UAVs joint vehicles as data mules for fast codes dissemination for edge networking in smart city," *Peer Peer Netw. Appl.*, vol. 12, no. 6, pp. 1550–1574, Nov. 2019.

[52] X. Liu, A. Liu, T. Qiu, B. Dai, T. Wang, and L. Yang, "Restoring connectivity of damaged sensor networks for long-term survival in hostile environments," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1205–1215, Feb. 2020.

[53] C. Zhuo, S. Luo, H. Gan, J. Hu, and Z. Shi, "Noise-aware DVFS for efficient transitions on battery-powered IoT devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 7, pp. 1498–1510, Jul. 2020.

[54] T. Li, A. Liu, N. N. Xiong, S. Zhang, and T. Wang, "A trustworthiness-based vehicular recruitment scheme for information collections in distributed networked systems," *Inf. Sci.*, vol. 545, pp. 65–81, Feb. 2021, doi: 10.1016/j.ins.2020.07.052.

[55] K. Abdel Hafeez, L. Zhao, B. Ma, and J. W. Mark, "Performance analysis and enhancement of the DSRC for VANET's safety applications," *IEEE Trans. Veh. Technol.*, vol. 62, no. 7, pp. 3069–3083, Sep. 2013.

[56] Q. Liu, P. Hou, G. Wang, T. Peng, and S. Zhang, "Intelligent route planning on large road networks with efficiency and privacy," *J. Parallel Distrib. Comput.*, vol. 133, pp. 93–106, Nov. 2019.

[57] G. Li, F. Li, T. Wang, J. Gui, and S. Zhang, "Bi-adjusting duty cycle for green communications in wireless sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–55, Dec. 2020, doi: 10.1186/s13638-020-01767-5.

[58] X. Liu, H. Song, and A. Liu, "Intelligent UAVs trajectory optimization from space-time for data collection in social networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 19, 2020, doi: 10.1109/TNSE.2020.3017556.

[59] Y. Ren, T. Wang, S. Zhang, and J. Zhang, "An intelligent big data collection technology based on micro mobile data centers for crowdsensing vehicular sensor network," *Pers. Ubiquitous Comput.*, pp. 1–17, Aug. 2020, doi: 10.1007/s00779-020-01440-0.

[60] J.-Y. Jung, D.-Y. Seo, and J.-R. Lee, "Counter-based broadcast scheme considering reachability, network density, and energy efficiency for wireless sensor networks," *Sensors*, vol. 18, no. 2, p. 120, Jan. 2018.

[61] T. Shu, W. Liu, T. Wang, Q. Deng, M. Zhao, N. N. Xiong, X. Li, and A. Liu, "Broadcast based code dissemination scheme for duty cycle based wireless sensor networks," *IEEE Access*, vol. 7, pp. 105258–105286, 2019.

[62] W. Qi, W. Liu, X. Liu, A. Liu, T. Wang, N. N. Xiong, and Z. Cai, "Minimizing delay and transmission times with long lifetime in code dissemination scheme for high loss ratio and low duty cycle wireless sensor networks," *Sensors*, vol. 18, no. 10, p. 3516, Oct. 2018.

ANG LI received the master's degree in computer application technology from Central South University, China. He is currently an Associate Professor with the School of Computer Science and Technology, Hunan Institute of Technology, China. His research interests include data mining, crowd sensing networks, wireless sensor networks, and bioinformatics.

WEI LIU received the Ph.D. degree in computer application technology from Central South University, China, in 2014. He is currently an Associate Professor and a Senior Engineer with the School of Informatics, Hunan University of Chinese Medicine, China. His research interests include software engineering, data mining, and medical informatics. He has published over 20 articles in the related fields.

SHAOBO ZHANG received the B.Sc. and M.Sc. degrees in computer science from the Hunan University of Science and Technology, Xiangtan, China, in 2003 and 2009, respectively, and the Ph.D. degree in computer science from Central South University, Changsha, China, in 2017. He is currently a Lecturer with the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interests include privacy and security issues in social networks and cloud computing.

MANDE XIE (Member, IEEE) was born in 1977. He received the Ph.D. degree in circuit and system from Zhejiang University, in 2006. He is currently a Professor with Zhejiang Gongshang University. His research interests include wireless sensor networks (WSNs), social networks, and privacy preservation.