SM-UAN: A Software-Defined Underwater Acoustic Network of Multi-Controllers for Inland Waterway Systems

WANG JIANPING1,2, LV YINGYING1, WEI CHEN2, (Senior Member, IEEE), GAO GUOHONG1,2, QU PEIXIN1,2, AND ZHOU YU1

1School of Information Engineering, Henan Institute of Science and Technology, Xinxiang 453003, China
2School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Wang Jianping (xunjii2002@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61701356, in part by the Young Scholar Training Program of Higher Education in Henan Province under Grant 2019GGJS172, and in part by the Key Scientific Research Projects of Colleges and Universities in Henan Province under Grant 20A520002.

ABSTRACT

Underwater acoustic network (UAN) is a lightweight system used in inland waterway systems. Software-defined networking (SDN) is a promising approach to improve the supervision abilities and integration services. This paper introduces SM-UAN, an SDN-based UAN with multi-controllers. Firstly, a hierarchical framework of SM-UAN is offered. Secondly, a multi-flow table structure and a multi-flow processing method is built. Followed by, a distributed deployment model of controllers and a load balancing mechanism are established. Finally, we construct an experimental scenario on Mininet with WOSS-NS3. For the multi-flow table processing method, the simulation demonstrates that the compression ratio is promoted to 24.1% and 29.4% separately, the packet forwarding rate 14.7% and the processing delay 31.2%, when comparing SM-UAN with the item-by-item matching (IM) system. For the load balancing mechanism, it shows that the maximum load for a master controller is balanced to 0.55, while the minimum load for a slave controller 0.46, which proves that a fair load distribution achieves. In addition, SM-UAN is compared with the traditional single-sink-based UAN (TS-UAN), the traditional multi-sink-based UAN (TM-UAN), and the SDN-based UAN with a single controller (SS-UAN). The results reveal that the survival time of SM-UAN is extended to 32.7%, 24.9%, and 21.6%, respectively. Also, the bit error rate (BER) of that is less than $10^{-4}$. In conclusion, advantages of SM-UAN have been highlighted, which provides theoretical support for inland waterway systems.

INDEX TERMS

Inland waterway sensing, multi-controller load balancing, multiple flow table, software-defined networking, underwater acoustic networks.

VOLUME 8, 2020 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
underwater nodes as virtual switches and sink nodes controllers. Virtualization is utilized to make UANs reusable [3].

Based on SDN, it realizes a lightweight UAN with cost-effective advantages and provides a new way to build inland waterway systems, whereas there are no whole technical means. On the one hand, the architecture of multi-flow tables is an important support for multi-service multiplexing. Nevertheless, there are complexities in designing processing methods for efficient and aggregated flow tables, due to the limited energy and functional requirements of underwater nodes. On the other hand, data flow is unevenness and uncertainty in time and space distribution, which poses a considerable obstacle to the load balancing of controllers.

In this paper, we approach SM-UAN, a software-defined UAN of multi-controllers for inland waterway systems, in which SM means the SDN-based architecture of multi-controllers. We present a hierarchical framework of SM-UAN and focus on the multi-flow processing method and the load balancing mechanism of controllers.

The main contributions of the paper are summarized as follows. First, we propose a hierarchical architecture of SM-UAN; Second, we build a storage structure of multi-flow tables by the ternary content addressable memory (TCAM), and present a multi-flow processing method based on the balanced iterative reducing and clustering using the hierarchy (BIRCH) algorithm combined with an Aho-Corasick (AC) automaton; Third, we achieve a deployment model of controllers and develop a load balancing mechanism according to the consistent hashing algorithm; Finally, we construct an experimental scenario which combines Mininet with WOSS-NS3 to analyze and verify the performance of SM-UAN.

To our knowledge, it is the first time that SDN-based UAN has been introduced in inland waterway systems. Based on this work, we hope to provide a hierarchical SDN deployment theory and method for digital river systems.

The remainder of the paper is organized as follows. Section II summarizes the related work. Section III describes the framework of SM-UAN. Section IV presents a storage structure of multi-flow tables and the multi-flow processing method. Section V introduces the multi-controller deployment model and the load balancing mechanism. The simulation experiment is discussed in section VI. Finally, section VII concludes the paper.

II. RELATED WORK

A. THE PROGRESS OF INLAND WATERWAY SENSING TECHNOLOGY

A sensing network is mainly used to collect and analyze various inland data, such as the channel dredging, geomorphic mapping of riverbeds, and underwater obstacle detection. It provides technical assistance for vessel traffic.

The European Union settled a unified inland navigation network named river information system (RIS) [4] for integrating and exchanging vessel data in the Danube and Rhine River. The United States developed a waterway system for locating and ensuring vessel safety [5]. Khalfallah et al. [6] introduced a solution of the wireless sensor network (WSN) for submarine obstacle detection. Ch et al. [7] designed a network for vessel tracking and underwater vibration detection. Zunair et al. [8] presented an observation network applied to vessel inspection in Bangladesh. Ellouze et al. [9] modeled a sensing network for detecting, identifying, and settling the specific location in rivers. Fan [10] developed an overload detection system that combined sensing, wireless communication, and statistical analysis to record and sample the data of vessels. Yang et al. [11] suggested a smart inland waterway sensing system based on the Internet of things (IoT) and security warning technology. Zhang et al. [12] employed a digital waterway model that combined the global positioning system (GPS) with observation equipment for navigation and early-warning services. Bian et al. [13] set up a low-power wide-area network (LPWAN) in the Yangtze River for the tele-control of navigation marks. Feng et al. [14] devised a channel prediction system for data sampling, such as channel hydrology, video surveillance, bridge height, and location. Chen and Chen [15] realized an inland sensing network named GeoSensor, which recorded and integrated real-time data, such as rainfall, navigation marks, and water level.

He [16] (2016) drew an evaluation model for the deployment of waterway sensing equipment, which guided the optimal implementation of devices. Song [17] put forward the deployment principle of sensing facilities and clarified the technical need in rivers, such as site planning and equipment selection. Wu et al. [18] proposed a deployment model for inland waterways and built a management system based on IoT. Wu and Zhong [19] designed a sensing network based on multi-functional navigation marks and placed the system in the Wuhan section of the Yangtze River. Song et al. [20] presented a navigation mark system, which used an advanced RISC machine (ARM) host to record the hydrological data. Li et al. [21] developed a vessel observation program based on the global navigation satellite system (GNSS) for data sampling, such as water depth, hydrology, location, etc.

In summary, studies on inland waterway systems and related facilities were plentiful. However, most of these works relied on hardware infrastructure, which were hard to reprogram, reconfigure, and update after being deployed. Networking and collaborative communication among nodes were not emphasized. Various systems were offline patterns, which had to salvage manually for data recovery. Besides, traditional facilities were bulky and expensive. Therefore, it is of profound significance to develop a systematic waterway system based on UAN.

B. ADVANCES IN SOFTWARE-DEFINED UANS

1) ADVANCEMENT IN THE ARCHITECTURE DESIGN OF SOFTWARE-DEFINED UANS

Akylidiz et al. [22] introduced a software-defined UAN called SoftWater. It was a solution based on the network
function virtualization (NFV) for underwater communication, which offered distinguished services with SDN. Wang et al. [23] fulfilled an SDN-based UAN which separated the data plane from the control plane. Ghahoor and Koo [24] defined the natural system and artificial acoustic system as two distinct types of UANs and designed the network based on SDN. The experimental results revealed that it advanced in the end-to-end delay, data delivery rate, and overhead. Lal et al. [25] suggested a network architecture that combined SDN and cognitive radio (CR), which provided support for underwater networks of high performance. Wang et al. [26] proposed a software-defined UAN architecture with multi-controllers. The simulation results revealed that the network was significantly improved. Petroccia et al. [27] presented an underwater cognitive communication architecture (CCA), which defined the protocol stack with an adaptive physical layer, a media access control (MAC) layer, and a network layer. Luo et al. [28] conducted technical analyses on the software-defined modem, network virtualization, IoT, and cloud computing. They indicated that SDN was the driving force for UANs of the next generation. Wang et al. [29] defined a UAN of the three-dimensional layered grid based on SDN. The experiment revealed that it had advantages in the end-to-end delay, the surviving time, and coverage. Wang et al. [30] made a comprehensive investigation of SDN and analyzed the progress of software-defined underwater networks.

All in all, there were a lot of software-defined UANs. Many of them were layered protocol structures based on OpenFlow. Corresponding devices, like virtual switches and controllers were defined. Unfortunately, there was a lack of application scenarios, such as how to plan and deploy them in inland waterways. In addition, most of the work only assumed a simple model with a single controller, which was a considerable gap in applications. Therefore, the implementation of UANs based on SDN in actual application scenarios becomes a hot spot.

2) THE PROGRESS OF THE EXPERIMENTAL PLATFORMS FOR SOFTWARE-DEFINED UANS
Torres et al. [31] proposed a software-defined experimental platform for UAN, which adjusted communication features through software configuration without relying on dedicated hardware. The experiment showed that it improved the development process based on SDN. Demirors et al. [32] presented a software-defined UAN platform named SEANet G2. It had spectrum agility and the flexibility of hardware or/and software support. Wei et al. [33] developed a UAN testbed based on SDN, which integrated Mininet with NS-3 through the TapBridge module for multi-mode communication. Demirors et al. [34] developed an underwater IoT platform called SEANet based on SDN, which allowed the flexible definition, addition, updating, and exchanging of hardware and software items. Wang et al. [35] suggested a UAN based on software-defined radio (SDR). It used a board of field-programmable gate array (FPGA) and a microprocessor of ARM for data processing. Luo et al. [36] designed an underwater IoT platform based on SDN. The experimental results demonstrated that the protocols developed in the platform can be migrated to the actual hardware platform accurately. Luo et al. [37] conducted a UAN platform based on SDN. The results indicated that it improved the efficiency, flexibility, and adaptability.

For all, there was some work focused on designing experimental platforms of software-defined UANs. Nevertheless, they were mostly in the preliminary stage instead of complete solutions. Some so-called software-defined platforms were based on CR or SDR. These studies were different from SDN based on OpenFlow. Some work claimed to be SDN type, but lacked the modeling of actual underwater scenes. They only combined traditional SDN platforms and did not consider the unique characteristics of UAN, such as energy consumption, the signal rate, the acoustic channel modeling and other constraints. As a result, the credibility was low. Therefore, developing experimental platforms of software-defined UANs are still meaningful and innovative tasks.

3) ADVANCEMENT IN THE FACILITY DESIGN OF SOFTWARE-DEFINED UANS
Restuccia et al. [38] proposed a software-defined UAN named iSonar. It used a lightweight protocol stack to transmit and receive data. Dol et al. [39] surveyed the development progress of software-defined acoustic modems (SDAM) and assessed the performance of the modem. Demirors et al. [40] designed a high-speed underwater modem that adjusted channel parameters based on SDN. Galioto et al. [41] developed a modem for UANs, in which the modulation and demodulation modules were carried out based on SDN. The experiment indicated that it adjusted the communication features flexibly. Pelekanakis et al. [42] explored a software-defined modem for underwater communication. The experimental results revealed that it had a higher data transmission rate and lower energy consumption.

In a word, the development of UAN devices had strong pertinence. Currently, most studies focused on the development of SDN-based modems. From the perspective of SDN, modems were considered as data layer devices. As management hubs, controllers were more critical. However, studies on the facility design of controllers were relatively rare. The development of controllers under constraints of limited resources is also a challenging task.

4) ADVANCEMENT IN THE HIGH-PERFORMANCE COMMUNICATION OF SOFTWARE-DEFINED UANS
Skilvanitis et al. [43] introduced a distributed underwater MIMO system based on SDN. The experimental results showed that it achieved adaptive data compression and enhanced the transmission performance. Sakaushi et al. [44] presented a software-defined underwater MIMO-OFDM system. The simulation displayed that it improved the transmission quality and efficiency of UANs. Ghanadrezai et al. [45] planned a cross-layer UAN based on SDN and suggested a
multi-hop relay method. The experimental results revealed that it preserved a higher data transmission rate and a lower delay. Lin et al. [46] complemented a software-defined routing scheme for UAN and realized an accurate estimation of the underwater acoustic channel. Wang et al. [47] set up a large-scale UAN based on SDN. The simulation results indicated that it had superior flexibility and improved the management performance.

In contrast, studies on high-performance communications of SDN-based UANS were more extensive. For example, software-defined clustering, time synchronization, routing, and others were all hot issues. These studies indeed improved the performance of UANs in a certain way. Unfortunately, there were no systematic frameworks and technical routes for deploying UANs in inland waterways.

III. THE SDN-BASED NETWORK ARCHITECTURE OF SM-UAN
A. THE DEPLOYMENT MODEL OF SM-UAN

A UAN consists of data nodes (DNs), sink nodes (Controllers), and base stations (BSs). A DN is made up of a buoy tie to an anchor through an anchor chain, in which various sensors are sealed to the buoy suspend in the water. The anchor chain is a lightweight cable with strong toughness and tension. When the DN is placed, the anchor sinks to the bottom of water. The anchor chain expands to a fixed length depending on the application needs. All buoys suspend at different depths of the water to form a three-dimensional space. Occasionally, DNs encapsulate in anchors for marking the riverbed. In some cases, an anchor chain may conduct the electromagnetic or optical communication from the anchor to the buoy. Therefore, the material selection and design are critical for the anchor chain. Besides, buoys float on the surface under certain circumstances and move randomly with the water flow. However, these systems are not appropriate for building inland networks.

Controllers are placed on vessels, autonomous underwater vehicles (AUVs), unmanned underwater vehicles (UUVs), remotely operated vehicles (ROVs), and other facilities. Sensing data are sent to matching controllers from DNs. Controllers transfer data to designated BSs after checking and aggregating. Application programs are installed on BSs. The deployment model of a UAN is illustrated in Fig. 1.

In this paper, we present SM-UAN for inland waterways, which consists of a hierarchical architecture with the application plane, the control plane, and the data plane. These planes are separated from each other. Fig. 2 shows the hierarchical architecture of SM-UAN.

The application plane disposes of BSs, which uses a northbound interface [48] to interact with the control plane. It splits the hardware resource into multiple slices. Each one forms a virtual facility and is allocated to a specific application for service virtualization. The upper-layer applications and the lower-layer facilities are abstracted into logical entities. Customers configure the network through programmable applications to share the same physical infrastructure.

Controllers equip with ONOS for management virtualization [49], which process the orchestration of the data plane. Controllers perform unified listening and statistics for link discovery and topology management. They fulfill policy formulation and flow delivery through a southbound interface.

The data plane composes of multiple DNs, which equip with Open vSwitch [50] for node virtualization. DNs perform clustering based on preset or issued instructions to form...
management units. The data plane is utilized for user-defined parsing, content matching, forwarding, and scheduling based on flow tables.

**B. THE COMMUNICATION PROCESS OF SM-UAN**

When SM-UAN is deployed, the communication is planned as well. In the control communication, BSs issue instructions to related controllers. Controllers forward these instructions to DNs in one-hop mode after being received. DNs respond to suiting controllers in the same mode. Finally, controllers gain the data in real-time, such as node ID, location, residual energy, resource occupancy rate, and the membership cluster ID. Therefore, controllers acquire the network topology comprehensively.

The data communication deals with data sensing and transmission. The data communication consists of three stages, i.e., intra-cluster communication, inter-cluster communication, and the communication between head nodes (HNs) and controllers. The clustering is triggered by the control communication before the intra-cluster communication. DNs perform the clustering algorithm which is preset or issued by controllers. After a while, multiple independent clusters are formed.

In a round, a DN belongs to only one appointed cluster. Each cluster elects an HN, and other nodes send the sensing data to it in one-hop mode. The HN verifies and aggregates data and then forwards them to a neighbor node, i.e., the HN of an adjacency cluster. Data are delivered to controllers based on the relay communication of HNs. The communication procedure is illustrated in Fig. 3.

To facilitate management, we implement the out-of-band communication in SM-UANs, i.e., we separate the data communication from the control communication. For data communication, it uses the multi-hop relay method. Considering the energy saving, the communication distance is set to be 100m. The control communication is mainly used for the control and management of SDN. Taking into account the immediacy and efficiency, the control communication is designed as a one-hop mode with 1km.

**IV. THE STORAGE STRUCTURE AND A PROCESSING METHOD OF MULTIPLE FLOW TABLES FOR SM-UAN**

As services expanding, matching fields of a flow table increase in an SDN-based UAN. Therefore, the space of flow table grows exponentially. It means that the storage repetition rate is high, and the matching efficiency is poor. Multiple flow tables are supported in OpenFlow v1.3. Flow entries and storage space are reduced through multi-flow table conversion. Nevertheless, considering the energy consumption and resource occupancy, design the storage structure and the processing method of multi-flow tables are challenging.

**A. THE STORAGE STRUCTURE OF SM-UAN**

In SM-UAN, DNs compose the data plane, which forward data based on OpenFlow to ensure the configuration, communication compatibility, and interoperability. Aiming at the flexibility, redundancy, and overhead of a single flow table, we design a storage structure of multiple flow tables based on OpenFlow v1.3, as shown in Fig. 4.

Flow tables are stored in TCAM, which is attached to a microcontroller unit (MCU). All fields are filtered through the mask. When matching OpenFlow entries, the MCU sends keywords to search the TCAM and returns the address index after finding a matched entry. Finally, the MCU reads the required data in the RAM. Considering the TCAM space, the flow table of each level is decomposed into smaller representation items [51]. We resolve flow tables into entries based on the management domain, service ID, virtual node ID, and the controller domain.

**B. A PROCESSING METHOD OF MULTIPLE FLOW TABLES FOR SM-UAN**

Flow table features are matched in order of the priority stored in the TCAM. The rapid feature extraction is the basis of data matching. Generally, fixed fields are established for feature extraction. In theory, the more the feature field, the higher the extraction accuracy. However, due to the energy and capability of DNs, the predetermined storage cannot be achieved for feature extraction. In addition, the complexity of creating and preserving flow for feature extraction is high.

The BIRCH algorithm [52] fulfills data aggregation through a single-pass scan of the dataset and minimizes the
We implement the feature extraction of multiple flow tables based on the BIRCH algorithm, which includes the definition of the cluster feature (CF) vector and the construction of a CF tree. The feature extraction method is shown as follows.

First, the features of flow tables are modeled as clusters \( \{F_i\} \) \((i = 1, 2, \ldots, n)\) with \(n\)-dimensional data points, and a clustering feature vector is defined as:

\[
\text{CF} = (N, \overline{LS}, \overline{SS})
\]

\[
\overline{LS} = \left( \sum_{i=1}^{N} F_i \right)
\]

\[
\overline{SS} = \left( \sum_{i=1}^{n} F_i^2 \right)
\]

(1)

where \(N\) represents the decomposition number of entries, i.e., the number of data objects in the cluster, \(\overline{LS}\) the centroid of the cluster, and \(\overline{SS}\) the diameter of the cluster.

Let \(R(\mu_1, \mu_2, \ldots, \mu_n)\) represent a reference point, for an \(n\)-dimensional feature \(F(X_1, X_2, \ldots, X_n)\) of multiple flow tables, aggregate \(R\) into an \(n\)-dimensional data point \(D(X_1, X_2, \ldots, X_n)\), as shown in:

\[
D_i = F_i \left( \left[ \frac{X_i}{\mu_i} \right] + \frac{1}{2} \right), \quad i = 1, 2, \ldots, n
\]

(2)

To unify the reference point \(\mu\), we add a parameter \(\sigma\), which is shown as:

\[
D_i = \frac{F_i}{\sigma_i} \left( \left[ \frac{X_i}{\mu_i} \right] + \frac{1}{2} \right), \quad i = 1, 2, \ldots, n
\]

(3)

Features are decomposed into reference sets, which form a CF tree. Multiple entries are used as root nodes, and reference sets non-leaf nodes. Each DN stores the clustering information of its child nodes. Leaf nodes extract the features. The process for building a CF tree is presented in Fig. 5.

In SM-UAN, a packet is matched by the entry of table 0 at first. If the matching fails, it will be forwarded to the next table. If all tables are not matched, a table missing event will occur. In this case, it will perform a drop operation or transmit the packet to the controller. The matching process of multiple flow tables is illustrated in Fig. 6.

As the extraction items are increasing, the matching efficiency will reduce sharply, and the matching complexity will increase exponentially. This paper proposes a feature matching method based on the AC automaton, which settles a keyword set and employs the automaton for matching [53]. The matching process is listed as follows:

First, a matching pattern string is set up based on the feature items that are extracted by the BIRCH algorithm. And it creates a dictionary tree, i.e., a trie. Then, the pattern string is inserted into the trie in sequence, and each mapping node is used for the state transition of the trie. If no comparable state transition occurs, a new trie will create. Followed by, it traverses the trie based on the breadth-first method and adds a fail pointer to the trie. After that, it decides the matching domain of each node by setting up a suffix substring list and expands the trie into an AC automaton. Finally, it adopts the mapping node as the transition basis to perform the state transition on the automaton and achieves the fast matching features of multiple flow tables.

V. A DEPLOYMENT MODEL AND A LOAD BALANCING MECHANISM OF MULTI-CONTROLLERS FOR SM-UAN

In an SDN-based UAN, DNs send massive Packet-in messages to interact with controllers. If there are enormous nodes placed in the network, a single controller cannot respond timely, which will cause the network bottlenecks. Besides, when the controller fails, the network will paralyze. Often, a deployment model of multi-controllers is employed in large-scale UANs. However, there are great differences in the cooperative processing of controllers.

Although roles settle in discrepant controllers such as ONOS and OpenDaylight [54], no unified standards support the east-west interface. It poses a notable obstacle to the collaborative communication of controllers. Due to the random deployment of underwater nodes, the management domain of
each controller is different, and the number of DNs it manages is distinctive. Therefore, the load of controllers will cause uneven distribution. Furthermore, with the dynamic changing of the network, the load of controllers is varying. In short, the load balancing mechanism is a major challenge, which relates to the stability, robustness, and security of the system.

A. THE DEPLOYMENT MODEL OF MULTI-CONTROLLERS FOR SM-UAN

The vertical and horizontal schemes are two deployment strategies in SM-UAN. In the vertical scheme, sub-controllers are directly connected to a general root controller, the flexibility of which is poor. In the horizontal method, controllers are placed in different domains through the east-west interface. Considering the flexibility, in this paper we adopt the horizontal method. We develop five modules based on ONOS, which are the device manager, topology manager, link load monitor, data forwarder, and the routing manager. The relationship of these modules is shown in Fig. 7. Besides, multiple controllers settle the mapping mechanism. The horizontal deployment model of multi-controllers is shown in Fig. 8.

The master, slave, and equal are three roles defined in an ONOS system. For collaborative processing, we rebuild the roles as master, pri-slave, and sec-slave. During initialization, master controllers are allocated to fixed clusters. Moreover, each cluster allocates a pri-slave controller and a sec-slave one. Meanwhile, the pri-slave controller in a cluster is mapped as the sec-slave of another cluster. Data of the master controller and the pri-slave controller back up mutually. According to the real-time load, the pri-slave controller will migrate partial load to a sec-slave one. Taking into account the data consistency, the role mapping of controllers is designed. The mapping mechanism of controllers is presented in Fig. 9.

It is clearly that the IDs of 1, 3, 5, and 7 are four master controllers, and the IDs of 2, 4, 6, and 8 are four slave controllers that have dual roles of pri-slave and sec-slave.
Therefore, the load of a slave controller is the sum of that pr-i slave controller and sec-slave one.

**B. THE LOAD DETECTION MECHANISM OF MULTI-CONTROLLERS**

As Packet-in messages sent by DNs increase, the load of master controllers rises sharply. When the load reaches to the preset threshold, it will trigger the load balancing algorithm, and partial load will migrate to corresponding slave controllers.

1) THE COST OF CONTROLLERS

We divide the UAN into different domains, i.e., $G = \{D_1, D_2, \ldots, D_k, \ldots, D_n\}$. Supposing that master controllers are deployed in each domain, and there is no overlap between any two domains, i.e., a DN only belongs to one cluster at any time. Given a controller set as $V = \{v_1, v_2, \ldots, v_j, \ldots, v_u\}$, a domain controller as $v_k$, and DNs set as $\{n_{k1}, n_{k2}, \ldots, n_{km}\}$ in domain $D_k$. For load detection, the cost of a controller is defined as follows:

(1) The interaction cost between DNs and residual controllers

When the controller $v_k$ in the domain $D_k$ fails, the set of residual controllers is $V^* = V - \{v_k\}$. Let $d(n_{ki}, v_j)$ denote the shortest distance between a DN $n_{ki}$ and the controller $v_j$. Given $\chi_{n_{ki}v_j}$ as the link factor, and $|D_k|$ the number of residual DNs in domain $D_k$. If $v_k$ fails, the interaction cost $C_{interaction}$ among DNs and the residual controllers is expressed as:

$$C_{interaction} = \frac{1}{|D_k|} \sum_{n_{ki} \in D_k} \sum_{v_j \in V^*} d(n_{ki}, v_j) \cdot \chi_{n_{ki}v_j}$$ (4)

(2) The average communication failure cost of DNs

Assuming that the failure probability of the link between a HN and a controller is $l$, the average communication failure cost $C_{disconnection}$ of a DN in the domain $D_k$ is shown in:

$$C_{disconnection} = \frac{1}{|D_k|} \sum_{n_{ki} \in D_k} \prod_{v_j \in V^*} \left(1 - l^{d(n_{ki}, v_j)}\right) \cdot \chi_{n_{ki}v_j}$$ (5)

(3) The migration cost of DNs

In SM-UAN, the random deployment produces the non-uniform clustering, which causes the number of DNs in each cluster to be quite different. If the controller related to a cluster is overloaded or fails, flow of part or all affiliated DNs are migrated to other controllers. That is, to release the connection with the current controller and transfer the flow to other controllers. We adopt the equal probability distribution (EPD) method [55] to quantify the migration cost. When the controller $v_j$ does not fail, the load $L_{v_j}$ will be quantified according to the number of Packet-in messages it handles.

Let $R_{n_{ki}v_j}$ represent the average number of Packet-in messages, which are sent from the DN $n_{ki}$ to a master controller in domain $D_k$. Due to the migration of DNs, the mean number of Packet-in $R_{v_j}$ that the controller $v_j$ needs to handle is expressed as:

$$R_{v_j} = L_{v_j} + \sum_{n_{ki} \in D_k} \frac{1}{|V^*|} \cdot R_{n_{ki}v_j}$$ (6)

If the maximum number of Packet-in messages can be handled by the controller $v_j$ is max ($R_{v_j}$), the migration cost is expressed as:

$$C_{migration} = \text{var}_{v_j \in V^*} \left(\frac{R_{v_j}}{\text{max} \left(R_{v_j}\right)}\right)$$ (7)

where the vector variance of max ($R_{v_j}$) is represented by $M_{R_v}$, as shown in:

$$M_{R_v} = \left(\frac{R_{v_1}}{\text{max} \left(R_{v_1}\right)}, \frac{R_{v_2}}{\text{max} \left(R_{v_2}\right)}, \ldots, \frac{R_{v_{|V^*|}}}{\text{max} \left(R_{v_{|V^*|}}\right)}\right)$$ (8)

In (8), $|V^*|$ represents the number of controllers in $R_v$.

(4) The communication cost of controllers

Data of each cluster transfer to a specified master controller. To impose the load balancing mechanism, a mapping method of controllers is designed. The master controllers and the slave ones are required to interact in real-time for gaining the load conditions. Therefore, there will be communication overhead $C_{traverse}$ among controllers, which is shown as:

$$C_{traverse} = \frac{1}{|D_k|} \sum_{n_{ki} \in D_k} \sum_{v_j \in V^*} R_{n_{ki}v_j} \times S_{n_{ki}v_j} \times \sum_{v_j \in V^*} d(n_{ki}, v_j)$$ (9)

where $S_{n_{ki}v_j}$ means the controller set in the shortest path from a DN $n_{ki}$ to the controller $v_j$. Therefore, the total cost $Cost_K$ is shown as:

$$Cost_K = \alpha C_{interaction} + \beta C_{disconnection} \nonumber + \gamma C_{migration} + \mu C_{traverse}$$ (10)

where $\alpha, \beta, \gamma, \mu$ represent weight factors of the above costs.

For the DN node $n_{ki} \in \{n_{k1}, n_{k2}, \ldots, n_{km}\}$ and the controller $v_j \in V^*$, the constraint of $Cost_K$ is shown as:

$$\sum_{j} \chi_{n_{ki}v_j} \geq 2$$ (11)

In (11), it ensures the backup membership matrix is a binary variable, and preserves two slave controllers for a DN. In this way, the controller $v_j$ in domain $D_k$ gains the backup information $[\chi_{n_{ki}v_j}]_{D_k \times |V^*|}$. By traversing the network, the backup membership matrix is achieved, and all controllers obtain the global view of the network.

2) THE LOAD DETECTION OF CONTROLLERS

According to the costs defined above, the load of a controller can be identified. We divide the cost into three intervals, i.e., $(0, 0.3]$, $(0.3, 0.6]$, and $(0.6, 1.0]$. The cost matches to three
load conditions, which are the light load, the suitable load, and the heavy load.

If a master controller is in the light-load state, it will broadcast an accept-node-request message. The signal strength threshold $Th_{\text{Accept}}$ is set in this message. A DN whose status is uncontrolled receives the message and compares its signal strength with $Th_{\text{Accept}}$. If its signal strength is less than $Th_{\text{Accept}}$, the DN will wait for the next accept-node-request message. If the signal strength is greater than $Th_{\text{Accept}}$, the DN sends a register-request message to the controller and adjusts the node state to the managed status simultaneously.

If the controller is in the suitable load, it will broadcast a lock-node message. A field of the cluster ID is included in this message. DNs belong to the cluster that the controller manages will receive this message, and keep remaining in the original cluster. Nodes in other clusters will discard the message because the cluster ID does not match.

If the controller is in the heavy-load state, it will broadcast a release-node-request message. This message contains a signal strength threshold $Th_{\text{Release}}$ and the cluster ID, which will be received by DNs in the cluster. A DN receives the release-node-request message and compares its signal strength with $Th_{\text{Release}}$. If the signal strength is greater than $Th_{\text{Release}}$, it will not respond to the controller. Otherwise, the DN sets its node status to uncontrolled and replies to a release-node-response message. The controller deletes the management information of DN from its domain after receiving this message. DNs in other clusters will discard this message because the cluster IDs do not match. The pseudocode of the load detection procedure is shown in Table 1.

| Algorithm 1. Load detection of controllers |
|------------------------------------------------|
| 1. INPUT $V_{\text{St}} = \{v_1, v_2, \ldots, v_n\}$; |
| // Initialize the controller set |
| 2. INPUT $N_{\text{St}} = \{n_1, n_2, \ldots, n_m\}$; |
| // Initial underwater node set |
| 3. set $\alpha, \beta, \gamma, \mu$; |
| // Set the weight factor of the cost |
| 4. void Load-detection($N_{\text{St}}, V_{\text{St}}$) |
| 5. int i=1, j=1; |
| 6. for each ($n_i$, in $N_{\text{St}}$) |
| 7. { |
| 8. for each ($v_j$, in $V_{\text{St}}$) |
| 9. { |
| 10. compute $d(n_i, v_j)$; |
| // The shortest path from $n_i$ to $v_j$ |
| 11. $C_{\text{interaction}} \leftarrow \frac{1}{|D_i|} \sum_{n \neq \text{uncontrolled}} d(n, v_j) Z_{n, v_j}$; |
| // Information interaction overhead |
| 12. $C_{\text{disconnection}} \leftarrow \frac{1}{|D_i|} \sum_{n \neq \text{uncontrolled}} \prod_{v \neq v_j} \left(1 - e^{-f(n, v)}\right) Z_{n, v}$; |
| // Node disconnection overhead |
| 13. $C_{\text{migration}} \leftarrow \frac{\text{var} \left( \frac{R_v}{\max(R)} \right)}{|D_i|}$; |
| // Node migration overhead |
| 14. $C_{\text{reverse}} \leftarrow \frac{1}{|D_i|} \sum_{n \neq \text{uncontrolled}} R_{n, v_j} \times Z_{n, v_j} \times \sum_{v \neq v_j} d(n, v) |D_i|$; |
| // Communication overhead between multi-controllers |
| 15. $Cost_t \leftarrow \alpha C_{\text{interaction}} + \beta C_{\text{disconnection}} + \gamma C_{\text{migration}} + \mu C_{\text{reverse}}$; |
| // Total cost |
| 16. /* Detection process under light load conditions */ |
| 17. if ($Cost_t < 0.3$) |
| 18. { |
| 19. $v_j$, sends Accept-Node-Request(); |
| 20. if ($\text{Strength}(\text{Accept \_Node \_Request}) > Th_{\text{Accept}}$) |
| 21. { |
| 22. $n_i$, sends Register-Request() to $v_j$; |
| 23. } |
| 24. else |
| 25. { listen $v_{j+1}$; // Listen to the next controller |
| 26. } |
| 27. } |
| 28. /* The detection process under suitable load conditions */ |
| 29. else if ($Cost_t < 0.75$) |
| 30. { |
| 31. $v_j$, sends Lock-Node(); // Lock node state |
| 32. } |
| 33. else |
| 34. { |
| 35. /* Detection process under heavy load conditions */ |
| 36. sends Release-Node-Request(); |
| 37. if ($\text{Strength}(\text{Release \_Node \_Request}) < Th_{\text{Release}}$) |
| 38. { |
| 39. set $n_i$, status uncontrolled; |

3) A LOAD BALANCING MECHANISM BASED ON THE CONSISTENT HASHING ALGORITHM

In the load distribution stage, a master controller performs the initial allocation of flow. If the controller is overloaded, the excess load is migrated to corresponding slave controllers. For a fair distribution, we design a load balancing mechanism based on the consistent hashing algorithm [56].

First, we employ a hash function to map flow and controllers into a 32-bit value, i.e., the space range is from 0 to $2^{32} - 1$. Second, the flow traverses the space clockwise to find a closest controller. If the controller is in a light-load state, it will receive the assigned flow. Otherwise, it will continue to find the next one nearby. The controller takes the node ID to calculate its hash address and uses the flow ID to calculate the hash address of the flow.

We map controllers and flow into a closed ring separately. Given the maximum number of flow that a controller handles as $\delta_{\text{max}}$, we set an idle flag $IS_{\text{idle}}$ and a flow number $Flow_{\text{count}}$ to a controller. The flow traverses the closed ring clockwise to find the nearest controller. If the status of a controller satisfies $IS_{\text{idle}} = 1$, i.e., it is in a light-load state, the flow allocates to the controller, and the flow number is marked as $Flow_{\text{count}} = Flow_{\text{count}} + 1$. If that satisfies $IS_{\text{idle}} = 0$, the flow continues...
TABLE 1. (Continued.) The pseudocode of the load detection procedure.

| Line | Description |
|------|-------------|
| 40   | \( n_n \) sends Release-Node-response(); |
| 41   | \( n_j \) removes \( n_n \); |
| 42   | start algorithm 2; |
| 43   | } |
| 44   | else{ |
| 45   | listen to \( n_{n+1} \); |
| 46   | } |
| 47   | } |
| 48   | \( i++, j++ \); |
| 49   | } |
| 50   | } |

FIGURE 10. The initial distribution of flow.

FIGURE 11. The unfairness of load distribution.

FIGURE 12. The fair redistribution of flow.

to traverse the hash ring until the next closest controller is found.

For example, there are four controllers \((C_1, C_2, C_3, C_4)\) and 12 flow \((F_1, F_2, \ldots, F_{12})\) in SM-UAN. If we set the initial state of a controller as \(\delta_{\text{max}} = 4\), \(\text{Flow}_{\text{count}}(i) = 0\), and \(\text{IS}_{\text{idle}}(i) = 0\). Then, \(F_1\) and \(F_2\) are assigned to \(C_1\), \(F_3\), \(F_4\), \(F_5\), and \(F_6\) to \(C_2\). Similarly, \(F_7\), \(F_8\), and \(F_9\) are allocated to \(C_3\), \(F_{10}\), \(F_{11}\), and \(F_{12}\) are assigned to \(C_4\). The initial distribution of flow is presented in Fig. 10.

In fact, there are uneven distributions. If \(C_4\) fails, \(F_{10}\), \(F_{11}\), and \(F_{12}\) cannot be allocated to other controllers. According to the initial distribution, \(F_{10}\) and \(F_{11}\) can be allocated to \(C_1\) sequentially, and the hash ring stops running. Therefore, \(F_{12}\) cannot be assigned to any controller, as shown in Fig. 11.

If another controller fails at the same time, the flow that cannot be allocated to any controller will increase substantially. Therefore, we use the consistent hash algorithm to map a physical controller into multiple virtual ones. When a new controller adds, there will be numerous logical controllers, which are inserted into the hash ring. It realizes the fair redistribution of flow for load balancing.

We assume that a physical controller maps as two virtual ones. Therefore, \(F_1\) and \(F_2\) are assigned to \(\text{VC}_{1-1}\), while \(F_3\), \(F_4\), \(F_5\), and \(F_6\) to \(\text{VC}_{2-1}\) and \(\text{VC}_{2-2}\) automatically. Followed by, \(F_7\) and \(F_8\) to \(\text{VC}_{3-1}\), and \(F_9\) is allocated to \(\text{VC}_{3-2}\). Here, the flow distribution completes in the initial state. For \(F_{10}\) and \(F_{11}\), the hash ring is scanned in order, and then the two flow is allocated to \(\text{VC}_{1-2}\). Finally, \(F_{12}\) is assigned to \(\text{VC}_{3-2}\). The fair redistribution of flow is shown in Fig. 12.

The pseudocode of the load balancing process based on the consistent hashing algorithm is shown in Table 2.

VI. SIMULATION AND EXPERIMENTS
A. EXPERIMENTAL SETTINGS
In this paper, we set up the experimental environment which equips with 10 controllers and 200 DNs. Controllers are
TABLE 2. The pseudocode of the load balancing process.

| Algorithm 2: The load balancing based on the consistent hashing algorithm |
|---------------------------------------------------------------|
| 1 INPUT \( v_m \leftarrow \{v_1, v_2, \ldots, v_m\} \); //Define slave controller set |
| 2 INPUT \( F_m \leftarrow \{F_1, F_2, \ldots, F_m\} \); //Define flow set |
| 3 INPUT \( N_m \leftarrow \{N_1, N_2, \ldots, N_m\} \); //Define node set |
| 4 \#DEFINE Virtual-Nodes-Per-Controller 4; //Define the maximum number of virtualized nodes of the controller |
| 5 Void Load-distribution(\( v_m \), \( F_m \), \( N_m \)) { |
| 6 map(\( F_m \), \( N_m \)) ; |
| 7 Hash-Flow= mHash(\( F_m \)); //Map underwater nodes and flow |
| 8 Hash-Controller= mHash(\( v_m \)); //The flow is mapped to the hash ring |
| 9 Virtual_controllers = n x Virtual_Nodes_Per_Controller //Number of virtual controllers |
| 10 for(i=1; i<virtual-Controllers; i++) |
| 11 { |
| 12 if( \( IS_{\text{add}} = 1 \) & Flow_count < \( \delta_{\text{max}} \) ) |
| 13 { |
| 14 Virtual_controller(\( C_i \)) = Hash_controller.add([hash(\( F_i \))]); //Flow distribution |
| 15 Flow_count = Flow_count + 1 ; //Point to the next flow |
| 16 return isadd=true; //Add successfully |
| 17 } |
| 18 else |
| 19 { |
| 20 virtual(\( C_i \)) = ControllerList.remove([hash(\( F_i \))]); //Remove the virtual node |
| 21 return isadd=false; //Unallocated flow |
| 22 } |
| 23 Virtual_controller(\( C_i \)) & Virtual_controller(\( C_i+1 \)); |
| 24 } |
| 25 } |

The deployment model of DNs is considered as a Poisson distribution. The deployment depth of DNs is in 3m to 30m. We use Cbench to simulate Packet-in messages and wait controllers for delivering Flow-Mod packets. We build a blade system with 10 IBM servers. Each one configures with two Intel E5-2698 CPUs and 256GB of memory. A master controller is allocated to a 20-core CPU, 64GB memory, and 100GB disk space. A slave controller (including pri-slave and sec-slave) is configured with a 10-core CPU, 32GB memory, and 50GB disk space. A DN is equipped with a 2-core CPU, 4GB of memory, and 10GB of disk space. The main experimental parameters are listed in Table 3.

B. PERFORMANCE ANALYSIS OF THE MULTI-FLOW PROCESSING METHOD

In the experiment, we use CBench to generate flow tables with \( 10^2 \) and \( 10^3 \) entries, respectively. We mainly compare the storage efficiency and performance with SM-UAN and the item-by-item matching (IM) system [59].

Multi-flow tables are supported in OpenFlow 1.3, which adopts the IM system. The IM is a commonly used multi-flow processing technology, which implements pipeline processing according to defined matching domain and rules. For the multi-flow processing, the flow table will be matched sequentially from table 0. It uses the matching result of the previous table as the input of the next one.

1) THE COMPARISON OF STORAGE EFFICIENCY

In SM-UAN, the storage efficiency is constrained by flow table levels and the repetition proportion of matching fields. We compare the compression ratios under different repetition proportions and different flow table levels. At a lower repetition proportion, the compression ratio of SM-UAN is comparable to that of the IM system. The main reason is that when the repetition proportion is low, there are fewer compressible matching fields. However, when the repetition proportion is high, the compression ratio of SM-UAN is superior to that of the IM system.

As shown in Fig. 13 (a), when the repetition proportion is less than 40%, the compression ratios of SM-UAN and IM system are basically the same. However, when the repetition proportion is greater than 40%, the compression ratios of the two networks appear to be quite different. If the repetition proportion gets up to 45%, the compression ratio of SM-UAN is 0.32, and that of IM system is 0.27. When the repetition proportion reaches to 50%, the compression ratio

installed with ONOS system and embedded the load balancing mechanism. DNs are set up with OvS and burned into the multi-flow processing method. We construct an experimental scenario that combines Mininet [57] with WOSS-NS3 [58] for the performance analysis of SM-UAN. Mininet is an emulator which realizes the topology definition and builds an SDN simulation environment. WOSS-NS3 is an integrated framework of the world ocean simulation system (WOSS) and NS-3. It is a multi-threaded framework that permits the integration of existing underwater channel simulators, and expects environmental data as input and channel realization as output. In the experiment, Tap-Bridge is used to connect Mininet and WOSS-NS3. Tap Bridge is designed to integrate hosts into NS-3 simulations.

We simulate the status of SM-UAN, including the position, depth, residual energy, and communication capabilities of DNs. In the experiment, DNs, controllers, and the applications are running on the Mininet host. The multi-flow processing method is designed on DNs, in which equip with Open vSwitch for node virtualization. The load balancing mechanism is set on controllers, in which equip with ONOS for management virtualization. The applications dispose on BSs, which use a northbound interface to communicate with controllers. Packets are sent from specific ports of DNs and can be relayed to the network device and finally transmitted to the UWA channel. The received data packets are sent through Tap-Bridge and processed by the network program in Mininet.
TABLE 3. Experimental parameters.

| Parameter                      | Value            | Parameter                      | Value            |
|--------------------------------|------------------|--------------------------------|------------------|
| Controller type                | ONOS             | Node type                      | Open vSwitch     |
| Control channel type           | Rayleigh Fading  | Data channel type              | Rice             |
| Noise type                     | Complex Gaussian | Spread Code                    | Walsh-Hadamard   |
| Channel bandwidth              | 6 kHz            | Carrier frequency              | 15 kHz           |
| Range of data communication    | 100m             | Range of control communication | 1km              |
| Modulation type                | QPSK             | Statistical method             | Monte-Carlo      |
| Underwater acoustic velocity   | 1430-1500 m/s    | Node mobility                  | 1-5 m/s          |
| Doppler spread factor          | 0.3-1.4          | Maximum multipath delay        | 40 ms            |
| Guard interval                 | 120 ms           | Code length                    | 8                |
| Threshold of residual energy   | 0.35             | Threshold of BER               | 0.005            |
| Threshold of delay             | 0.01             | Threshold of load balancing    | 0.85             |

![Graph](image)

FIGURE 13. The comparison of compression ratios.

The compression ratio of SM-UAN is 0.36 and that of the IM system is 0.29. The compression ratio of SM-UAN is promoted to 24.1%.

Like the status of the repetition proportion, compression ratios of the two networks are basically the same when the level of a flow table is low. Nevertheless, if the level is greater than four, there will be a significant difference in the compression ratio. As shown in Fig. 13 (b), if the level is five, the compression ratio of SM-UAN is 0.44 and that of the IM system is 0.42. Additionally, when the level is seven, the compression ratio of SM-UAN is 0.48, and that of the IM system is only 0.42. In this case, the compression ratio of SM-UAN is approximately a 29.4% improvement.

For IM, repetition fields are compressed item by item. It increases the complexity of matching queries and does not save the storage space. In SM-UAN, the BIRCH algorithm is used to form a feature tree, and feature items are used as matching strings to build a pattern tree (i.e., trie). The trie is expanded into a tree-shaped AC automaton for matching. Therefore, a tradeoff between storage efficiency and the matching performance is achieved.

2) THE PERFORMANCE COMPARISON OF MULTI-FLOW PROCESSING METHOD

As data increase, the matching efficiency declines, which causes the transmission delay. We compare the packet forwarding rate and the handling delay with SM-UAN and the IM system.

As shown in Fig. 14(a), the packet forwarding rate of SM-UAN is higher than that of IM system. When networks run at 400 seconds, the data packets that IM forwards is generally stable, and it only handles 6130 packets. By contrast, forwarding packets keep increasing in SM-UAN. In 400 seconds, 7032 packets are forwarded, and at 600 seconds, 8827 ones are forwarded. The result announces that the packet forwarding rate of SM-UAN is approximately a 14.7% improvement than that of IM system.

We also compare the handling delay under different number of processing packets. We see that as the processing packets increase, the handling delay of two systems rises, as shown in Fig. 14(b). By comparison, the handling delay of IM system is longer than that of SM-UAN. When the number of the processing packets is 100, the delay of the two networks are 25.3ms and 44.1ms, respectively. When the number increases to 450, the delay of IM system is 141ms, and that of SM-UAN is 108ms. The results show that the processing delay of SM-UAN is decreased by approximately 31.2%.

It is worth mentioning that when the processing packets exceeds 350, the handling delay of the two systems will
change slightly. It is because that levels of multi-flow tables are not restricted in IM system. It needs the flow of each level to make a matching action. With the combination entries increasing, the matching efficiency drops sharply, and the handling complexity increases exponentially.

For SM-UAN, the BIRCH algorithm is used to perform effective clustering through a single-pass scan of flow tables. And an AC automaton is built for the flow feature set. Under the constraints of storage space, energy consumption, and cost, feature extraction items are used as matching pattern strings to build an AC pattern tree. A mismatch pointer is set on the tree for the fast matching. Therefore, the performance of multi-flow processing method is preferable in SM-UAN than that of IM system.

### C. PERFORMANCE ANALYSIS OF THE LOAD BALANCING MECHANISM

To verify the load balancing mechanism, we conduct a simulation of nine rounds continuously based on the Monte Carlo method. We count node status under each round, as shown in Table 4. In SM-UAN, we divide the management domain based on the clustering mechanism proposed in [47]. We set a mapping rule in which five controllers undertake master roles and five more slaves are divided into pri-slave and sec-slave roles. To map DNs and controllers in each round, which is shown in Table 5.

In the load balancing mechanism of SM-UAN, the load thresholds assigned to master controllers are predetermined. Master controllers correspond to designated management domains, pri-slave controllers and sec-slave controllers construct the mutual mapping of corresponding domains with each other. In this paper, we define that 80% of the load in a domain is first allocated to a master controller, and the remaining 20% is assigned to the slave controller. Among them, 16% of the load is allocated to the pri-slave controller, and the last 4% is allocated to the sec-slave controller. When the load of a master controller is in the heavy load, it will start the load distribution based on the consistent hash algorithm and transfer part load to corresponding slave controllers.

In a round, massive Packet-in messages produced by CBench are sent to relative controllers. We divide the load of controllers into three stages. Due to the randomness of

---

**TABLE 4. Node status statistics.**

| Node state | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 | Round 6 | Round 7 | Round 8 | Round 9 |
|------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| active     | 189     | 186     | 178     | 171     | 160     | 156     | 151     | 142     | 125     |
| sleep      | 11      | 14      | 22      | 29      | 27      | 33      | 30      | 36      | 51      |
| dead       | 0       | 0       | 0       | 0       | 13      | 11      | 19      | 22      | 24      |

**TABLE 5. The mapping relationship between DNs and controllers.**

| Master ID | Pri-slave ID | Sec-slave ID | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 | Round 6 | Round 7 | Round 8 | Round 9 |
|-----------|--------------|--------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| C1        | C6           | C10          | 36      | 20      | 36      | 31      | 34      | 33      | 33      | 34      | 15      |
| C2        | C7           | C6           | 51      | 39      | 30      | 41      | 21      | 30      | 29      | 21      | 34      |
| C3        | C8           | C9           | 28      | 47      | 27      | 34      | 32      | 36      | 27      | 31      | 24      |
| C4        | C9           | C8           | 26      | 41      | 37      | 38      | 34      | 31      | 27      | 24      | 30      |
| C5        | C10          | C7           | 47      | 39      | 48      | 27      | 40      | 27      | 35      | 31      | 23      |

**FIGURE 14. The Performance comparison of multi-flow processing.**

(a) Comparison of the packet forwarding rate.

(b) Comparison of the handling delay.
services, load of master controllers is fluctuating in the first stage. As shown in Fig.15, the maximum load of a master controller reaches to 0.85, and the minimum load is 0.7. However, load on slave controllers is minimal at this stage, and as the network runs, the load will rise steadily. The minimum load is 0.1 for C8, and the maximum load is 0.25 for C10.

In the second stage, load of master controllers rises to the preset threshold, and the load balancing mechanism is triggered. Therefore, excessive load of master controllers is allocated to the matching slave ones. We see that load of master controllers falls, and that of slave controllers rises. As shown in Fig. 16, load of C2 and C3 is fallen to 0.6, and that of C8 and C9 is raised up to 0.48.

In the third stage, the process of load adjustment and distribution is completed. At this time, the excessive load of master controllers has migrated to corresponding slave ones. Finally, load of all controllers tends to be in a balanced state. As shown in Fig. 17, the load of C4 is the highest, and its load is 0.55, while that of C5 is balanced to 0.51. By contrast, the maximum load is 0.50 for C9, and the minimum load of that is 0.46 for C7. It reveals that a fair load distribution has achieved.

D. THE COMPARISON OF THE NETWORK SURVIVAL TIME AND BER

We compare the network survival time and BER with the other three networks, which are the traditional single-sink-based UAN (TS-UAN), the traditional multi-sink-based UAN (TM-UAN), and the SDN-based UAN with a single controller (SS-UAN). It is worth mentioning that SS-UAN and SM-UAN are SDN type. However, TM-UAN and TS-UAN are based on traditional UAN architecture. The difference is that multiple sink nodes are deployed in TM-UAN, and there is only one sink node in TS-UAN.

In the experiment, we use the clustering algorithm which was proposed in [47]. For TS-UAN and TM-UAN, fixed instructions are settled on DNs for clustering. While for SS-UAN and SM-UAN, the clustering algorithm is implemented based on SDN, i.e., instructions which encapsulate in OpenFlow messages are issued by controllers to DNs. The main parameters of the four networks are shown in Table 6.

1) THE COMPARISON OF THE NETWORK SURVIVAL TIME

Energy consumption is the key to SM-UAN. We use the average expectation of residual energy to quantify the network survival time, as shown in Fig. 18. As data increases, the expectation of residual energy for the four systems decreases. In comparison, energy consumption of TS-UAN has a larger decline, while that of SM-UAN has a relatively flat downward trend. When the data is 10,000, the expectation of TS-UAN is 0.7, while that of SM-UAN is 0.9. When data increases to 50,000, the expectation drops to 0.1, while that of SM-UAN is still maintained at 0.69.

In TS-UAN, it uses a single sink node, and all HNs send messages to the sink node at the same time. Due to traditional communication instructions, flexible management cannot be achieved, and the energy consumption is rapid. For SS-UAN, it utilizes a single controller in the control plane and cannot impose the load balancing mechanism. When data is 10,000, the expectation of SS-UAN is 0.84, while data increases to 50,000, the expectation drops to 0.38. However,
for TM-UAN, it adopts multiple sink nodes and a mapping rule of sink nodes is comparable to that of SM-UAN. When data is 10,000, the expectation is 0.8. If data increases to 50,000, the expectation drops to 0.29.

Based on data analysis, it indicates that the survival time of SM-UAN is extended to 32.7%, 24.9%, and 21.6%, respectively, when compared with TS-UAN, TM-UAN, and SS-UAN. Therefore, it prolongs the network survival time in SM-UAN.

2) THE COMPARISON OF BER

The comparison of BER is illustrated in Fig. 19. We observe that BER of TS-UAN is relatively largest, while that of SM-UAN is proportionately smallest. With the running of the system, BER of the four networks increases simultaneously. In the initial state, the BER of TM-UAN is basically the same as that of SM-UAN. Nevertheless, when the network runs for 800 seconds, that of TM-UAN increases sharply. All in all, BER of SM-UAN is less than $10^{-4}$, while that of the other three systems is higher than $10^{-4}$. Particularly for TS-UAN, its BER gets closer to $10^{-3}$.

For TS-UAN, it uses fixed instructions for clustering and communication. With the gradual exhaustion of energy, some nodes will die. It means that re-clustering will happen and be more frequent. In this case, efficiency of TM-UAN is lower than that of TS-UAN, not to speak that of SM-UAN. The BER of TS-UAN is the highest, due to the fixed communication instructions and a single sink node with inadequate performance. It just proves the advantage of SM-UAN.

VII. CONCLUSION

This paper presented SM-UAN, a software-defined UAN with multi-controllers for inland waterway systems. First, we proposed a hierarchical framework for SM-UAN with a separate data plane, control plane, and application plane. Second, we designed a multi-flow table structure based on TCAM connection storage and put forward a processing method of multi-flow tables with the BIRCH algorithm and an AC automaton. Followed by, we introduced a distributed deployment model of controllers and presented a load balancing mechanism based on the consistent hashing algorithm. Finally, we constructed a scenario on Mininet with WOSS-NS3 to analyze and verify the performance of SM-UAN.

For the multi-flow processing method, we quantified the storage efficiency based on the compression ratios under different repetition proportions and different flow table levels. The results showed that, if the repetition proportion was less than 40%, and the levels were not more than 4, the compression ratios of SM-UAN and IM system were approximately the same. However, when the repetition proportion reached to 50%, the compression ratio of SM-UAN was promoted to 24.1%. If levels of a flow table were greater than four, the compression ratio of SM-UAN was approximately a 29.4% improvement. And then, we compared the forwarding data rate of the two systems. The results illustrated that the packet forwarding rate of SM-UAN was approximately a 14.7% improvement. Next, we compared the handling delay under different number of processing packets. The results revealed that the processing delay of SM-UAN decreased by approximately 31.2%. The experiments revealed that the compression ratio was higher for the considerable amount of data. Therefore, the processing performance was better in SM-UAN.
For the load balancing mechanism, it indicated that load of all controllers tended to be in a balanced state, the maximum load for a master controller was 0.55, while that for a slave one was 0.46. It proved that a fair load distribution of multi-controllers based on the consistent hashing algorithm had been achieved.

For analyzing the network survival time, we compared SM-UAN with TS-UAN, TM-UAN, and SS-UAN. It showed that the survival time of SM-UAN was extended to 32.7%, compared with that of TS-UAN, TM-UAN, and SS-UAN. It showed that the survival time of SM-UAN was extended to 32.7%, while that for others was higher than 10−3. Particularly for TS-UAN, the BER was closer to 10−3. It indicated that SM-UAN had advantages in the survival time and BER.

Currently, we are evaluating the deployment of SM-UAN in rivers. We hope to provide support for building inland waterway systems based on this work.

REFERENCES

[1] L. Xue-xiang, “Summary and prospect of the technology of inland digital waterway,” in Proc. 4th Int. Conf. Transp. Inf. Saf. (ICTIS), Aug. 2017, pp. 163–171.
[2] S. Jiang, “On reliable data transfer in underwater acoustic networks: A survey from networking perspective,” IEEE Commun. Survys Tuts., vol. 20, no. 2, pp. 1036–1055, 2nd Quart., 2018.
[3] P. V. Venkateswara Rao, N. Mohan Krishna Varma, and R. Sudhakar, “A systematic survey on software-defined networks, routing protocols and security infrastructure for Underwater Wireless Sensor Networks (UWSNs),” in Proc. CODEC, Feb. 2020, pp. 551–559.
[4] H. Liu, H. Zeng, J. Zhou, and Y. Lv, “Present situation and prospect of intelligent navigation system technology for inland waterway,” J. Hydraulic Eng., vol. 6, pp. 82–87, Dec. 2015.
[5] R. G. Wright, “4-Dimensional virtual aids to navigation for arctic maritime transits,” in OCEANS, Sep., 2017, pp. 1–5.
[6] Z. Khalifah, I. Fajjari, N. Aitsaadi, P. Rubin, and G. Pujolle, “A novel 3D underwater WSN deployment strategy for full-coverage and connectivity in rivers,” in Proc. IEEE Int. Conf. Comput. (ICCC), May 2016, pp. 1–7.
[7] S. Ch, M. Kumar, N. K. Yadav, N. Kumar, and S. Shiva, “Tracking and emergency detection of inland vessel using GPS-GSM system,” in Proc. 2nd Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT), May 2017, pp. 2039–2042.
[8] H. Zunair, W. U. Hasan, K. T. Zaman, M. I. Haque, and S. S. Aoyon, “Design and implementation of an IoT based monitoring system for inland vessels using multiple sensors network,” in Proc. 2nd Int. Conf. Smart Sensors Appl. (ICSSA), Jul. 2018, pp. 38–43.
[9] N. Ellouze, S. Rekhis, and N. Boudriga, “A WSN-based solution for pollution detection and localization in waterways,” Arabian J. Sci. Eng., vol. 44, no. 4, pp. 3213–3233, Apr. 2019.
[10] Q. Fan, “Overload monitoring system for inland river ferries,” in Proc. 5th Int. Conf. Transp. Inf. Saf. (ICTIS), Jul. 2019, pp. 1487–1490.
[11] Y. Yang and C. Huang, “Preliminary idea for intelligent waterway of the Yangtze Estuary,” Port Waterway Eng., vol. 11, pp. 7–11, Nov. 2017.
[12] S. Zhang, T. Zhang, and Y. Wu, “Three-dimensional waterway system for ship navigation based on integrated virtual waterway and flow simulation,” J. Waterway, Port, Coastal, Ocean Eng., vol. 143, no. 1, Oct. 2017, Art. no. 04016011.
[13] N. Bian, Z. Zhang, and D. Wang, “Reflections on the application of low-power wide-area networks in the Yangtze River digital waterway,” China Water Transp., vol. 18, no. 7, pp. 43–48, Mar. 2018.
[14] B. Feng, J. Liu, and Z. Yan, “Design and implementation of the Pearl River inland waterway perception platform,” China Water Transp., vol. 19, no. 7, pp. 70–74, Oct. 2019.
[15] Z. Chen and N. Chen, “A real-time and open geographic information system and its application for smart rivers: A case study of the yangtze river,” ISPRS Int. J. Geo-Inf., vol. 8, no. 3, p. 114, Feb. 2019.
[40] E. Demirors, G. Sklivanitis, G. E. Santagati, T. Melodia, and S. N. Batalama, “A high-rate software-defined underwater acoustic modem with real-time adaptation capabilities,” *IEEE Access*, vol. 6, pp. 18602–18615, 2018.

[41] G. E. Galioto, D. Garlisi, D. Croce, L. Mistruttta, R. Badalamenti, I. Tinnirello, C. G. Giaconia, P. Petroilli, and P. Ganci, “FLUMO: FFlexible underwater MOdern,” in *Proc. OCEANS-Marseille*, Jun. 2019, pp. 1–12.

[42] K. Pelekanakis, R. Petroccia, Y. Fountoulakis, D. Green, S. Fioravanti, J. Alves, S. Blouin, and S. Pecknold, “A simulation study for long-range underwater acoustic networks in the high north,” *IEEE J. Ocean. Eng.*, vol. 44, no. 4, pp. 850–864, Oct. 2019.

[43] G. Sklivanitis, Y. Cao, S. N. Batalama, and W. Su, “Distributed MIMO underwater systems: Receiver design and software-defined testbed implementation,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–8.

[44] A. Sakaushi, M. Okano, K. Kanai, and J. Katto, “Performance evaluations of software-defined acoustic MIMO-OFDM transmission,” in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[45] H. Ghannadrezaei, J.-F. Bousquet, and I. Haque, “Cross-layer design for software-defined underwater acoustic networking,” in *Proc. OCEANS-Marseille*, Jun. 2019, pp. 1–7.

[46] C. Lin, G. Han, M. Guizani, Y. Bi, and J. Du, “A scheme for delay-sensitive spatiotemporal routing in SDN-enabled underwater acoustic sensor networks,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9280–9292, Sep. 2019.

[47] J. Wang, G. Gao, P. Qu, W. Chen, S. Zhang, X. Zuo, and Z. Yu, “A software-defined clustering mechanism for underwater acoustic sensor networks,” *IEEE Access*, vol. 7, pp. 121742–121754, 2019.

[48] M. Pham and D. B. Hoang, “SDN applications—The intent-based north-bound interface realisation for extended applications,” in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 372–377.

[49] Y. Han, J. Li, D. Hoang, J.-H. Yoo, and J. W.-K. Hong, “An intent-based network virtualization platform for SDN,” in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2016, pp. 353–358.

[50] *Linux Foundation, Open vSwitch*. Accessed: Aug. 18, 2019. [Online]. Available: https://www.openvswitch.org/releases/openvswitch-2.11.1.tar.gz

[51] K. Qui, J. Yuan, J. Zhao, X. Wang, S. Secci, and X. Fu, “FastRule: Efficient flow entry updates for TCAM-based OpenFlow switches,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 484–498, Mar. 2019.

[52] G. Nirmala and K. K. Thyagharajan, “A modern approach for image forgery detection using BRICH clustering based on normalised mean and standard deviation,” in *Proc. Int. Conf. Commun. Signal Process. (ICCSSP)*, Apr. 2019, pp. 0441–0444.

[53] X. Wang and D. Pao, “Memory-based architecture for multicharacter Aho–Corasick string matching,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 1, pp. 143–154, Jan. 2018.

[54] *ODL*. Accessed: Jun. 27, 2019. [Online]. Available: https://www.opendaylight.org/

[55] H. Khan, M. Aslam, and M. A. Khan, “Properties of exponential ratio type estimators in equal probability sampling: A simulation study,” *Commun. Math. Statist.*, vol. 6, no. 1, pp. 91–118, Mar. 2018.

[56] Q. Liu, W. Cai, J. Shen, B. Wang, Z. Fu, and N. Linge, “VPCH: A consistent hashing algorithm for better load balancing in a Hadoop environment,” in *Proc. 3rd Int. Conf. Adv. Cloud Big Data*, Oct. 2015, pp. 69–72.

[57] *SIGNET Lab*. Accessed: Apr. 18, 2019. [Online]. Available: http://telecom.dei.unipd.it/en/signet/.

[58] Z. Wu, Y. Jiang, and S. Yang, “An efficiency pipeline processing approach under underwater acoustic networks in the high north,” *J. Alves, S. Blouin, and S. Pecknold, “A simulation study for long-range underwater acoustic networks in the high north,” IEEE J. Ocean. Eng.*, vol. 44, no. 4, pp. 850–864, Oct. 2019.

[59] S. N. Batalama, “A high-rate software-defined underwater acoustic networking,” in *Proc. IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 484–498, Mar. 2018.

**LY YINGYING** was born in Xinxiang, Henan, China, in 1985. She received the master’s degree from the School of Information Engineering, Zhengzhou University of Technology, Zhengzhou, China, in 2011. She is currently a Lecturer with the School of Information Engineering, Henan Institute of Technology. Her research interests include underwater acoustic networks, signal preprocessing, and network protocol.

**WEI CHEN** (Senior Member, IEEE) was born in Wuhan, China, in 1963. He received the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005. He is currently a Professor, a Doctoral Supervisor, and the Dean of the School of Automation, Wuhan University of Technology. His research interests include underwater acoustic sensor networks, software-defined networks, channel measurement and modeling, as well as massive multi-input multi-output technologies. He is a member of the Communications and Navigation Standardization Technical Committee of the Ministry of Transport of China, a member of the Academic Committee on Communications and Navigation of the China Institute of Navigation, and a Senior Member of the China Electronics Society.

**GAO GUOHONG** was born in Zhengzhou, Henan, China, in 1975. He received the M.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2008. He is currently pursuing the Ph.D. degree with the Wuhan University of Technology. He is also an Associate Professor and the Vice President of the School of Information Engineering, Henan Institute of Science and Technology. His research interests lie in the areas of software-defined networks and wireless sensor networks.

**QU PEIXIN** was born in Nanyang, Henan, China, in 1977. He received the M.S. degree from the Radio Physics Department of Physical Electronics, Belarusian State University, Minsk, Belarus, in 2007. He is currently pursuing the Ph.D. degree with the Wuhan University of Technology. He is also an Associate Professor and the Vice President of the School of Information Engineering, Henan Institute of Science and Technology. His research interests include the areas of software-defined networks and wireless sensor networks.

**ZHOU YU** was born in Jiaozuo, Henan, China, in 1963. He received the M.S. degree from the School of Electronic Information and Communications, Wuhan University of Technology, Wuhan, China, in 1994. He is currently a Professor with the School of Engineering, Henan Institute of Science and Technology. His research interests include software-defined radio and wireless sensor networks.

**WANG JIANPING** was born in Baoji, Shaanxi, China, in 1981. He received the Ph.D. degree from the Wuhan University of Technology, Wuhan, China, in 2019. He is currently an Associate Professor with the School of Information Engineering, Henan Institute of Science and Technology. His research interests include the areas of underwater acoustic sensor networks, software-defined networks, and wireless sensor networks.

**WANG JIANPING** was born in Xinxiang, Henan, China, in 1985. She received the master’s degree from the School of Information Engineering, Zhengzhou University of Technology, Zhengzhou, China, in 2011. She is currently a Lecturer with the School of Information Engineering, Henan Institute of Technology. Her research interests include underwater acoustic networks, signal preprocessing, and network protocol.

**WEI CHEN** (Senior Member, IEEE) was born in Wuhan, China, in 1963. He received the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2005. He is currently a Professor, a Doctoral Supervisor, and the Dean of the School of Automation, Wuhan University of Technology. His research interests include underwater acoustic sensor networks, software-defined networks, channel measurement and modeling, as well as massive multi-input multi-output technologies. He is a member of the Communications and Navigation Standardization Technical Committee of the Ministry of Transport of China, a member of the Academic Committee on Communications and Navigation of the China Institute of Navigation, and a Senior Member of the China Electronics Society.

**ZHEN GUOHONG** was born in Zhengzhou, Henan, China, in 1975. He received the M.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2008. He is currently pursuing the Ph.D. degree with the Wuhan University of Technology. He is also an Associate Professor and the Vice President of the School of Information Engineering, Henan Institute of Science and Technology. His research interests lie in the areas of software-defined networks and wireless sensor networks.

**QU PEIXIN** was born in Nanyang, Henan, China, in 1977. He received the M.S. degree from the Radio Physics Department of Physical Electronics, Belarusian State University, Minsk, Belarus, in 2007. He is currently pursuing the Ph.D. degree with the Wuhan University of Technology. He is also an Associate Professor and the Vice President of the School of Information Engineering, Henan Institute of Science and Technology. His research interests include the areas of software-defined networks and wireless sensor networks.

**ZHOU YU** was born in Jiaozuo, Henan, China, in 1963. He received the M.S. degree from the School of Electronic Information and Communications, Wuhan University of Technology, Wuhan, China, in 1994. He is currently a Professor with the School of Engineering, Henan Institute of Science and Technology. His research interests include software-defined radio and wireless sensor networks.