# UAV based SDN system for wireless sensor networks


## ABSTRACT

..
..
..
..
..
..
..
..
..
..
..
..
..
..
..
..
..


## 1. INTRODUCTION

Software defined network (SDN) and Artificial intelligence (AI) are two popular topics in recent research field, both of which greatly improve the performance of traditional networks.

On the one hand, SDN enables flexible programmability on traditional network system by using programmable data plane and centralized network controller []. It shows remarkable advantages in good management of network control, less operating costs and easy deployment for new applications []. On the other hand, AI technology promises a smarter and more effective network system due to its high performance in predicting and optimizing network capacity, e.g. network throughput, energy consumption, etc.

In this paper, we study integrating these two elegant techniques into the wireless sensor networks (WSN). Our fundamental observation is that conducting AI methods in WSN requires the support of SDN. This is due to two major reasons. First, traditional sensor device has limited processing capability and cannot deal with complex computations. Second, the sensor device is also lack of storage space to support complicated operations. With the help of SDN, the central controller can conduct complex computations and store huge data information instead.

Unfortunately, implementing a real SDN in sensor system is still an open problem. The existing work [7, 1, 6, 2, 3] all design framework of SDN for WSN and validate them by simulations. Even worse, these simulational SDN works construct the central controller of SDN in their database station. The controller transmits control instructions with sensor nodes by multi-hop communications. This causes a great deal of energy consumption, which makes it a paradox since sensor network ought to be an energy-efficient network.

Our insight is that, since it is unpractical to deploy SDN for sensor system by multi-hop control, we refer to various equipments and techniques, and finally find the unmanned aerial vehicle (UAV) perfect to serve as the SDN controller. The flexibility of UAV can promise one-hop communication between controller and nodes. Furthermore, the reason why we choose UAV instead of terrestrial mobile controller is that, the air-ground communication range is almost 10 times than that of ground-ground communication under ZigBee protocol, which are around 145 meters and 15 meters respectively tested by our practical experiments. This is due to the air-ground communication suffers from less interference of the terrestrial WSN.

In this paper, we present XX[1], a real UAV based SDN sensor system. In XX, UAV plays the role of the central controller to monitor the sensor system. We implemented five fundamental applications: routing, network diagnosis, AI energy prediction, AI node selection and multi-tasks, and provide users with easy-to-use interfaces. Routing application allows users to update the flow table of nodes. Network diagnosis is used to find the fault source when network exception is detected. Multi-tasks application enables sensor to perform multiple tasks simultaneously. All the applications are within a ecosystem and can be combined to achieve synergistic effect. For example, users can execute the AI node se-

[1]XX is .

lection application to choose the node set for the multi-task application.

We implemented XX with DJI M100 UAVs and CC2560 Sensor tags in real experiments, and we conducted extensive experiments for a large-scale sensor network. We deploy 130 sensors .....

The main contribution of this paper is the first practically implemented SDN for sensor system, XX. We adopt AI tools in XX and realize five applications with easy-to-use interfaces. We then conduct practical experiments and the evaluations validate the high performance of XX.

The paper is organized as follows. We introduce related works in the next section. Design of XX is presented in Section 3, and we implement five fundamental applications: routing, network diagnosis, AI prediction, AI node selection and multi-tasks in Section 4. Section 5 gives the implement setup. The evaluation results are provided in Section 6 and we conclude the paper in Section 7.

## 2. RELATED WORK

### 2.1 Software Defined Wireless Sensor Networks

There are several existing SDN approaches for sensor system, namely flow sensor [7], SDWN [1], sensor openflow [6], TinySDN [2], SDN-WISE [3]. However, they are all implemented and evaluated by simulations.

**Flow Sensor.** The previous idea presented in paper [7] addresses reliability in the sensor networks through exploiting the OpenFlow technology. Coming up with the concept of flow-sensor instead of typical sensor, they have successfully made it possible to communicate between controller, gateway and sensors. Besides that, they proved flow sensor more reliable since data packets, control packets and the sensor nodes themselves can be easily monitored, regulated and routed whenever required. Therefore, a robust routing and uninterruptable messages flow of sensors are achieved. The results described in this paper shows flow-sensor is able to display much better performance even for large networks.

**SDWN.** The solution of supporting the SDN approach in LR-WPANS is first presented in [1]. Given the gap that advantages of SDN and the proper ways to expand that to wireless network were not clear enough, the group analyzes the opportunities of SDN-related wireless network and illustrates the requirements should be considered to utilize SDN solution for wireless network. They made a good attempt to develop the SDWN protocol stack.

**Sensor OpenFlow.** In the paper [6], the group toke a radical, yet backward and peer compatible, approach to tackle the problems existing in WSN such as resource underutilization, counterproductive, rigidity to policy change and manage difficulty. They propose SD-WSN with a separation between data plane performing flow-based packet forwarding and control plane performing network control. The core part designed is Sensor OpenFlow(SOF), which gives a standard protocol for the communication between data and control part. Based on the whole architecture they gave, the underlying network becomes programmable by using SOF.

**Tiny SDN.** TinySDN [2] is a TinyOS-based SDN framework aims to address the problem that only single controller can be coupled to the sink. Besides, TinySDN is a hardware independent framework that comprises two parts: SDN-enable sensor node and SDN-controller node. In order to test the reliability of this framework, the authors done some experiments on COOJA. Through analyzing results concerning delay and memory footprint, they found it is feasible in communication provided by SDN paradigm.

**SDN-WISE.** One solution for wireless sensor network is introduced in the paper [3], and the authors implement the prototype of this idea. Compared to other SDN solutions for wireless network, this solution successfully reduces the messages exchange needed between sensor node and controller. Besides, flexible APIs are provided by the authors, thus making the network much easier to program. Also, they done some experimental testbed, found that in a power limit hardware, the response delay is still very small, SDN-WISE shows its good performance in different operation conditions.

### 2.2 Applications for Wireless Sensor Networks

To design a ecosystem for sensor network, we are to implement several applications. We make a survey of the traditional sensor networks and find some typical applications: routing, network diagnosis, AI sensor, sensor selection and multi-tasks.

**Routing** Low-power and lossy network consist of devices whose processing, memory and energy are both constrained. Hence, traditional protocols canâĂŹt be used there. However, RPL is one of the standards for ipv6 routing in LLN. It builds the topology like a tree, performing DAO, DIO, ACK to generate a directed acyclic graph from one or more root to the leaf nodes. The experiments done in the paper [Low-Power Wireless IPv6 Routing with ContikiRPL] shows that Ipv6 routing with ContikiRPL is both lightweight and power-efficient.

**Network Diagnosis** Previous diagnosis algorithms share the same drawback that their process of gathering information is static and the reports sent to be analysis is not intermediate enough. Directional Diagnosis is an online diagnosis approach to dynamically recognize the most useful information according to an engine designed in the approach. Besides, the diagnosis process only focuses on the problematic area, so that it can produce a more accuracy prediction. To be specify, the whole ap-

proach consist of four main parts: Node Tracing, Trace Collection, Inference Model and Incremental Probing. The node tracing algorithms dynamically reconstruct topical topology and derive inner dependencies. Based on the data information collected, inference model can guide the next probe by Incremental Probing applying Belief network. The experiments done show the diagnosis result is efficient and in real time.

**AI sensor** The application of sensors often relays on the filed like sensor select, data prediction combining with AI algorithms. In order to select the subset of nodes to be active, related study give a way to dynamically response to the change of sensor network based on the historical information, the new sensor node has a different time factor from the old node when construct the predict network. Besides, the position of nodes may be highly used to reduce the weight of nearby nodes in a predict network. As for data prediction, on the one hand, the energy consuming information can be collected. Previous work on diagnosis, for example, only performs diagnosis when sensor almost run out of energy. On the other hand, the original data about temperature, humidity, light intensity is a reliable resources for prediction in time series.

Sensor Select Redundant sensor network always contains nodes that generating data stream with some overleap parts. Collected information can be useful to determine which sensors are active. The algorithms come up with in paper[5] are called Spatially Regularized Streaming Sensor Selection(SRSSS). The authors not only consider the spatial information but also the historical obversion of the sensor network. They apply higher weights to nearby sensors and introduce a time-forgetting factor so that their approach can predict more accuracy and response much faster than others.

**Multitask** One single sensor node may shoulder several tasks. To achieve power saving, when a task performed, we need to choose which sensor to be active and which to be inactive. Nowadays, work aims to design an energy-efficient sensor management strategy tends to develop their method in three ways: choose the proper subset of nodes to be active, which sensors to assign the task and the sampling rate on one node when performing task. Related study has found an efficient online algorithm considering sensor activation and task mapping to deal with dynamic events during runtimes of SDSNS.

# 3. ARCHITECTURE

To build software defined network system in WSN and achieve low energy consumption and high performance, the SDN controller must configure data plane within several hop. So we choose UAV as a mobile SDN controller. In our SDN system, UAV is a control plane which communicate with sensor nodes within on hop, it significantly reduce sensor nodes energy consumption in SDN system. At same time, to achieve high performance, the SDN controller should has much more intelligent than sensor nodes can to run real-time decision, our UAV carry a powerfully airborne computer which has ability to run some intelligent applications in real-time.

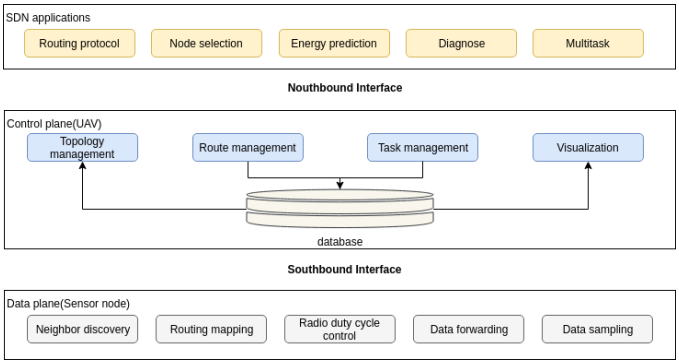


**Figure 1: Architecture of the system.**

The architecture of the UAV based SDN system for wireless sensor networks shown as Fig. 1. The system divided into three layers, data plane, control plane and management plane, respectively . The data plane running on sensor nodes, the controller plane and management plane running on UAVs. And the interface between data plane and control plane is defined as southbound interface, the interface between control plane and management plane is southbound interface.

Different from wired network, sensor network data plane not only runs data forwarding function, but also runs data sampling function, so user can define routing and sensing application at same time, on sensor nodes, it runs neighbor discovery for getting topology, data sampling for get data form environment and data forwarding for sensor data gathering.

On control plane, in order to make our system easy to use, we implement a SDN database and designed interface abstraction for data plane and SDN applications. The control plane connected data plane and management plane. The database manage the topology gathered from data plane, route table inputed form applications, and sensor task schedule table form management application.

On management plane, user can run tons of applications base on our well-defined southbound APIs, such

**Table 1: System API**

| Structure && Function | Description |
|---|---|
| **Sensor Control Interface** | |
| struct node | Sensor node structure |
| struct nodeset | A set of sensor nodes |
| struct neighbor_list | Neighbor infomation |
| struct energy_item | Energy statistic information |
| struct flow_table | Flow table |
| struct duty_cycle_table | Duty cycle control table |
| struct sensor_enable_table | All the nodes's states. Node state: {on,off} |
| switch_node(node,state) | Turn on or turn off the node |
| get_node_info(node) | Get node's information, including node's position, duty cycle, power, etc. |
| set_node_attr(node,attrTag,value) | Set node attribute, including duty cycle, radio strength, etc. |
| get_neighborlist(node) | Get the neighbor list of a node |
| **UAV Application Interface** | |
| **Routing** | |
| get_topology() | Get the topology of the network |
| get_flow_table(node) | Get the flow table of a node |
| set_flow(flow,node) | Set the flow of a node |
| **AI Node selection** | |
| nodeset simple_selection(nodeset) | Select sensor set by location information |
| nodeset SRSSS_selection(dataset) | Select sensor set by AI algorithm based on sensing data |
| **AI Energy Prediction** | |
| model_selsct(modeltype) | Select an AI model |
| model.train(dataset,ratio) | Train an AI model with learning ratio on the data set |
| model.test(dataset) | Test the AI model on the data set |
| model.predict(node) | Do the energy prediction for a node |
| **Multi-tasks** | |
| create_scheduler() | Create a task scheduler |
| scheduler.create_buffer() | Create a task buffer |
| scheduler.task_buffer_add(task,nodeset) | Add a new task to task buffer |
| scheduler.task_buffer_remove(task) | Remove a new task to task buffer |
| scheduler.task_buffer_update(task,nodeset) | Update a task to task buffer with a new nodeset |
| scheduler.task_update() | Schedule the added or removed tasks in the buffer |
| **Diagnosis** | |
| detect() | Detect problematic region with probes |
| get_topical_topology(nodeset) | Construct topical topology |
| diagnose_network(topology,nodeset) | Diagnose the failure nodes or lossy links |

as routing algorithms, network diagnose, sensor task scheduler, tec.

The Table 1 shows our southbound interface for user write applications.

The List. 1 demonstrate a deploy routing algorithm based on our southbound APIs. And the List. 2 demonstrate a AI selection and Muti-tasks application.

**Listing 1: An example of deploy routing algorithm**

```
topology = get_topology();
//calculate flowtable for each node
//based on topology
for(node in nodeset){
   node.flowtable =
      calculate_flowtable(topology);
}
//set flowtable for each node
for(node in nodeset){
   UAV fly to node;
   for(flow in node.flowtable)
        set_route(flow);
}
```

**Listing 2: An example of AI selection and Muti-tasks**

```
AI_Multitasks(taskset){
   create_scheduler();
   scheduler.create_buffer();

   for(task in taskset)
      scheduler.task_buffer_add(
         task,
```

```
            defaultset);

    scheduler.task_update();
    ...
    ...
    for(task in scheduler.task_buffer){
        data = get_collected_data();
        nodeset =
            SRSSS_selection(dataset);
        scheduler.task_buffer_update(
            task,
            nodeset);
    }
    scheduler.task_update();
}
```

# 4. APPLICATIONS

## 4.1 Overview

Traditional applications can not achieve complicated and efficient goals due to the limited processing power and memory space of sensors.

In XX, applications for wireless sensor networks are inspired by greater potential with the UAV based SDN controller. The central controller helps sensors execute complex calculations such as AI model training, as well as store global information. Besides, UAVs have flexible features and can deploy tasks to sensors by one-hop communication directly. Thus it enables the sensor network to achieve much more intelligent applications.

In XX, applications can be found for a variety of purposes, including routing, AI node selection, AI energy prediction, multi-tasks and network diagnosis. We design all these applications and provide easy-to-use interfaces to users as in Table 1.

## 4.2 Routing

one round
cluster header

**Table 2: Flow Table**

| Header Fields | Counters | Actions |
|---|---|---|

**Table 3: Header Fields**

| Ingress port | Ether Source | Ether Dst | IP src | IP dst |
|---|---|---|---|---|

Actions:

- Forward
- Drop
- Report

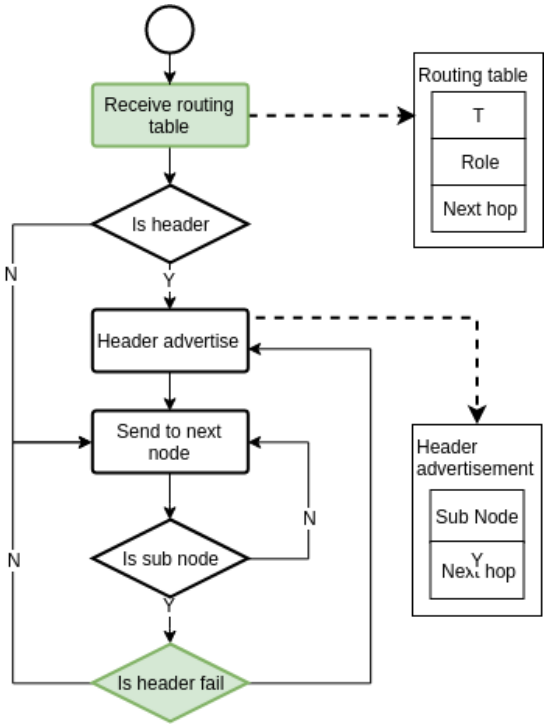


**Figure 2: Routing repair**

## 4.3 Network Diagnosis

### *4.3.1 Motivation*

Sensor nodes are provisioned with low-capacity batteries and will run out of energy in the end. Besides, uncertain environmental factors will lead to the failure of communications. Upon these two factors, there occurs network faults such as failure nodes and lossy links from time to time.

Network fault diagnosis is then designed to help network administrators monitor the network operational status and maintain a sensor network system. The key idea of the existing works on fault diagnosis is to collect running information from nodes and deduce root causes of network exceptions. The running information is collected by a mechanism of probing.

We implement the network diagnosis process as an application in our XX system. Since in XX we have our mobile UAV as a controller, a more flexible way is to infer the suspicious nodes of the network fault in traditional way first and then set the UAV to check out the fault sources.

### *4.3.2 Design*

We first introduce a state-of-the-art algorithm named DID [4], which is a directional diagnosis approach. We utilize the node tracing module and the tracing collection module of DID to infer the suspicious nodes in our network diagnosis application, and then set our UAV controller to confirm the network elements being faulty.

The node tracing module is conducted in each sensor

node. Every time a packet arrives at a node, it counts the source of the packet . The tracing collection module is set in the UAV controller. When network exception is detected, it gathers the tracing information in the tracing module of the relevant nodes and infers a suspicious node set. Next we set the UAV to fly through the node set and check each node first to diagnose the failed nodes. Then UAV collects the neighbor lists of all the nodes in the suspicious node set. With the collected neighbor lists, the UAV controller reconstructs the topical topology and compares it with the default topology to find the failed links.

Compared to DID, the diagnosis application in XX releases the complicated inference computations and can achieve accurate diagnosis since the UAV can fly to the sensors to confirm the network faults. XX can realize the four types of fault sources the same as DID:

- Node failure. This network failure is caused by the node itself.
- Link failure. This network failure is caused by the communication links between nodes, mainly relating to traffic flow in networks.
- Temporary failure. This network failure is caused by complex interior or exterior interferences and quick self-recovery
- Multiple failures. This network failure is caused by multiple failures above.

## 4.4 AI Node Selection

### 4.4.1 Motivation

It is inevitable that there will be a part of redundant sensors when deploying a practical wireless sensor network. These redundant nodes have overlaps of observation regions, and what makes the matter worse is that redundant nodes may cause great communication interference. Therefore it is significant to select proper sensors to avoid data redundancy and save the sensor network energy consumption.

In XX, we provide the node selection application to users. The SDN controller executes the selecting algorithm and send the control instructions to activate the selected nodes.

### 4.4.2 Design

Our XX system provide two main node selecting methods: greedy selection algorithm and SRSSS algorithm. This application will be extended to more elegant algorithms in our future work.

**Greedy selection algorithm.** We first provide a simple method to select the redundant nodes by a greedy selection algorithm, as described in Alg. 1. The key idea is to select nodes as less as possible to coverage the whole area based on the location and sensing range.

We implement the greedy selection algorithm in XX. The evaluations in section 6 show it greatly saves the sensors' energy and thus prolongs the network lifetime.

**Algorithm 1** Greedy Selection Algorithm

```
Input: Sensor set N, Selected set M, Target area Ω, Covering area Φ;
Initialize : M = ∅, Φ = ∅
while M ≠ N do
    if Φ = Ω then
        break; \\ Selected set has been found
    end if
    if ∀n_i ∈ (N − M) : range(n_i) ⊂ Φ then
        break;\\ Cannot cover the target area;
    end if
    Find n_i : argmax(Φ ∩ range(n)), n_i ∈ (N − M);
    Φ = Φ ∪ n_i
end while
Output: M;
```

**Spatially regularized streaming sensor selection (SRSSS).** To realize more intelligent and effective sensor selection, we introduce a state-of-the-art AI algorithm named spatially regularized streaming sensor selection (SRSSS) [5].

Different from the greedy selection algorithm, SRSSS is a multi-variate interpolation framework and focuses on selecting a subset of sensors in a streaming scenario to minimize collected information redundancy.

Traditional wireless sensor network is not suitable to implement an AI selection approach due to the limited computational capability of sensors. Some work use the database to collect data and make decisions by multi-hop communications. However, in this way the network will use up a great deal of energy. In our XX, the UAV fly through the nodes to pick up the collected data and executes the computations. Then it sends the control instructions to the nodes by one-hop communication and greatly saves the network energy.

The aim of SRSSS is to optimize its objective function which is an equation given certain constraints of collected information, location and energy consumption. The objective function is formulated as:

$$
\begin{aligned}
&(W_{k+1}, z_{k+1}) \\
&= arg\min_{W,z} \sum_{i=1}^{k} \mu^{k-i} \|X_k^i D_z W(I - D_z) - X_k^i (I - D_z)\|_2^2 \\
&+ \alpha \sum_{i,j=1}^{n} \|y_i - y_j\|_2 |W_{i,j}| - \beta \sum_{i,j=1}^{n} \|y_i - y_j\|_2 z_i z_j \\
&+ \lambda \|W\|_F^2 \\
&s.t. z = [z_1, ..., z_n] \in \{0,1\}^n, c^T z \leq P
\end{aligned}
\tag{1}
$$

The first term in (1) is to minimize the prediction error of the collected data and the following two terms incorporate spatial information. The last term constrains the complexity of the learned matrix $W$. The energy constraint is controlled by the inequality in (1). Because of the limitation of length, we leave out all the details. The meaning of the parameters and the mechanism of SRSSS can be seen in [5].

With the AI sensor selection process, XX becomes a smarter and adaptive sensor systems.

## 4.5 AI Energy Prediction

### *4.5.1 Motivation*

Wireless sensor networks (WSN) generally use multihop for data collection, and due to limited compute capability and storage capacity, sensors are unable to do data mining. Hence, self optimization is impossible.

The sensor next to data center always has the most routing task, which leads to the heaviest workload and the shortest battery life. Worse more, after the first sensor run out of energy, a new sensor will be selected to take the place of it with more workload. As a result, its lifetime is shorter and the whole sensor network will soon run out of energy. We call this Energy exhausted problem.

An ideal solution for Energy exhausted problem is to find a self-adjust routing schedule which according to Energy status of each sensors. Actually, it is a kind of dynamic programming problem, the optimization target is the longgest lifetime of the whole network, with complex constrains. We modified linear and simple MLP deep learning model to simulate and simplified this constrains. Challenges occurs in traditional WSN because of lack of global information energy information. With the help of UAV, our XX system can do this optimization

### *4.5.2 Design*

## 4.6 Multi-tasks

### *4.6.1 Motivation*

Wireless sensor networks (WSN) generally comprise of a group of spatially dispersed sensors. In a wireless sensor network, sensor nodes are equipped with various types of sensors monitoring and recording environmental conditions like temperature, sound, sunlight, humidity, etc.

A given sensing task involves multiple sensors to achieve a certain quality-of-sensing. Generally, an efficient task scheduling for the nodes is that nodes are able to perform multiple tasks simultaneously. For example, sensors deployed in a grove are assigned tasks to collect sunlight, temperature and humidity data and these tasks require different number of nodes with respective sensing range, rate and duration. However, traditional sensor networks are not suitable to conduct this multi-tasks due to the limitations of computation complexity for task arrangement of each node.

In our XX system, we implement the multi-tasks application with the help of the central controller. The SDN controller maintains programmable task scheduling and management modules while sensor nodes are loaded with interfaces to receive task control instructions.

### *4.6.2 Design*

A deployed wireless sensor networks are usually assigned with different data collection requirement. In XX, we design and implement multi-task application and provide easy-to-use interfaces to users.

When a user assign a task to XX, the UAV controller will first check out the energy and storage constraints of the required sensor set, as described in Alg. **??**. If the task requirement exceeds the capacity of the sensor set, it will be sent to a task queue; Otherwise it will be put into the task buffer to conduct.

**Algorithm 2** Sensor Constraint Detection

```
Input: Sensor set N, Task T;
Input: Remaining Energy for each node :Energy(u_i);
Input: Remaining Storage for each node :Energy(u_i);
Initialize: Energy capacity η, Storage capacity ξ;
Calculate the energy cost of T for each sensor as : φ;
Calculate the storage cost of T for each sensor as :ϕ;
for Each node u_i in N do
    if Energy(u_i)+φ ≥ η||Storage(u_i)+ϕ ≥ ξ then
        Set T to task queue;
    end if
end for
Set T to task buffer;
```

A sensor node may have different sensing ranges for different tasks.

There are several practical requirements.

Different tasks have different requirements, including time, sensing range, sensing ratio, etc.

For example tasks like sunlight collection only need to be carried out during the daytime.

Our system provide a task scheduling to

Sensors are usually assigned multi-tasks.

Sensors are assigned tasks to monitor a specific area.

Different tasks have different requirements, i.e.

- **Node set.** Users can assign tasks to

Table 4: Task Buffer

| Task ID | Node set | Sensing rate | Sensing range | Sensing duration |
|---|---|---|---|---|
| Task ID | Node set | Sensing rate | Sensing range | Sensing duration |
| ... | ... | ... | ... | ... |
| Task ID | Node set | Sensing rate | Sensing range | Sensing duration |
| ... | ... | ... | ... | ... |

- **Sensing rate.**
- **Sensing range.** The maximum distance that a node can detect.
- **Sensing duration.** The sensing time from start to end. There is no need to collect sunlight data at night.

Task scheduler do the arrangement.
Task buffer.
Task queue.
Scheduling table.
...

# 5. IMPLEMENTATION

We run modified contiki-ng, a system for next generation IOT (Internet of things), on data plane. we modified the network layer of contiki-ng, and abstract data plane interface to separate network control functions from data plane.

Neighbor discovery, After deployment, nodes start IPv6 neighbor discovery protocol to find the nodes which communicate with it within one hop, and save neighbor information to neighbor table and wait for UAV arrive and gather the topology information. Every neighbor's information include the neighbor's IP and RSSI.

Packet forwarding, the nodes get routing table from UAV through northbound interface, when it receive a data packet from other nodes, it just send it to the next hop node according the routing table.

Data sampling, node read task table received from UAV to decide when to wake up sensors to make sure node cost minimal energy to get same quality data.

On UAV, we run our UAV flight controller on ROS and use MySQL as our SDN database, UAV gathered topology data from every node and save in database, so user can easily read the whole network data from database, For the same reasoning, after user get routing algorithm result and save it to database via northbound interface, our middleware will automatically divide the whole network routing result to individual node routing table, and then send it to specific node through southbound interface. Base on our SDN framework, we provide well-defined northbound interface, applications will easily implement.

# 6. EVALUATION

In this section, we describle our performance evaluation in real-scene experiments. We distribute 100 TI SensorTag sensors in an about 250 $\times$ 250 square meters area, with contiki-ng as its operating system. We use different metrics for various applications and compare our works with other state-of-the-art works.

## 6.1 Routing

**Energy efficiency**

We estimate sensor energy cost by multipling different coefficients on CPU running time and radio listening and transmitting time and summing them up. The coefficients are proportional to the working current described in sensor's DataSheet.

**Fault-tolerance**

**Scalability**

## 6.2 Network Diagnosis

**Robustness**

## 6.3 AI Node Selection

**Scalability**

## 6.4 AI Energy Prediction

**Prediction accuracy**

## 6.5 Multi-tasks

**Energy Performance Scalability**

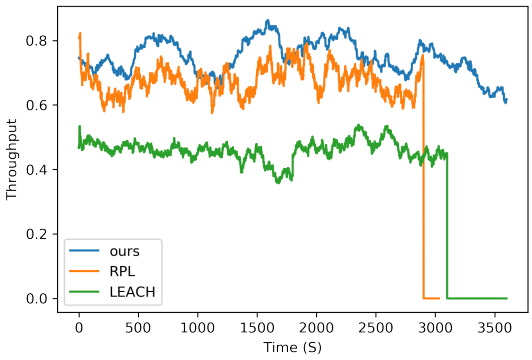


**Figure 3: Routing throughput**

# 7. CONCLUSION

Conclusion goes here.

# 8. REFERENCES

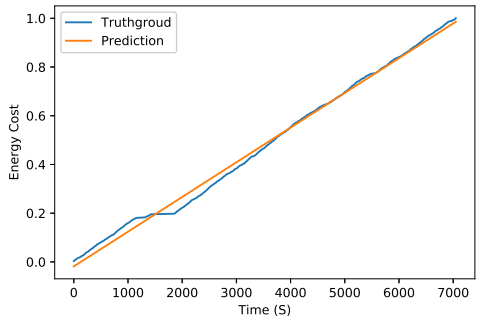


**Figure 4: energy prediction**

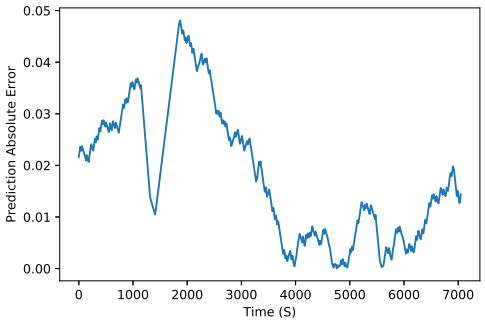


**Figure 5: energy prediction error**


[1] Costanzo, S., Galluccio, L., Morabito, G., and Palazzo, S. Software defined wireless networks: Unbridling sdns. In *Software Defined Networking (EWSDN), 2012 European Workshop on* (2012), IEEE, pp. 1–6.

[2] De Oliveira, B. T., Gabriel, L. B., and Margi, C. B. Tinysdn: Enabling multiple controllers for software-defined wireless sensor networks. *IEEE Latin America Transactions 13*, 11 (2015), 3690–3696.

[3] Galluccio, L., Milardo, S., Morabito, G., and Palazzo, S. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *Computer Communications (INFOCOM), 2015 IEEE Conference on* (2015), IEEE, pp. 513–521.

[4] Gong, W., Liu, K., and Liu, Y. Directional diagnosis for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 26*, 5 (2015), 1290–1300.

[5] Li, C., Wei, F., Dong, W., Wang, X., Yan, J., Zhu, X., Liu, Q., and Zhang, X. Spatially regularized streaming sensor selection. In *Thirtieth AAAI Conference on Artificial Intelligence* (2016).

[6] Luo, T., Tan, H.-P., and Quek, T. Q. Sensor openflow: Enabling software-defined wireless sensor networks. *IEEE Communications letters 16*, 11 (2012), 1896–1899.

[7] Mahmud, A., and Rahmani, R. Exploitation of openflow in wireless sensor networks. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on* (2011), vol. 1, IEEE, pp. 594–600.