

# Malak: An Intelligent, High-performance and Resilient Software Defined Wireless Sensor Network System

## ABSTRACT

Software defined network (SDN) enables artificial intelligence (AI) techniques to be implemented easily, and in turn AI techniques greatly improve the performance and resilience of SDN system. However, all the existing SDN systems in wireless sensor networks are implemented by simulations and there is no practical implementation of SDN system in WSN.

In this paper, we present Malak, the first practical SDN wireless sensor network system, which achieves intelligence, high-performance and resilience at the same time. In Malak, we utilize the unmanned aerial vehicle (UAV) to serve as the SDN controller. The UAV communicates with network nodes by one-hop communication and thus greatly reduce the network energy consumption. We designed easy-to-use interfaces and implemented five SDN applications, namely routing, multi-task, network diagnosis, AI energy exhaustion prediction, AI node selection. We also to users and the code is ready to be open source. Our evaluations show the throughput improves 80% compared with the RPL protocol by using the basic Malak routing algorithm. Meanwhile, the lifetime grows by 20% compared with RPL. And after deploying diagnosis and AI selection algorithm, the lifetime even increases by 25% and 35% more, respectively. Besides, the packet received ratio keeps over 90% in both small and large scale network. Code of Malak is ready to open source.

## 1. INTRODUCTION

Software defined network (SDN) [5, 36] paradigm is a trend that enables flexible programmability for networks by using programmable data plane and centralized network controller [45]. It shows remarkable strengths in network control, global optimization (e.g. throughputs [], diagnosis [], reliability []) and easy deployment for new applications [13].

With the support of SDN for networks, programmable artificial intelligence (AI) techniques [29, 32, 9] are easy to be implemented in networks []. In turn, AI technique makes SDN become 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.

Unfortunately, although Internet of Things (IoTs) [4] make wireless sensor networks [33] extremely pervasive and important, such as volcanic investigation [41], and agriculture monitoring [40], there is no real implementation of SDN in a WSN. Existing works [25, 10, 24, 11, 14] all design frameworks of SDN for WSNs and validate them by simulations. Even worse, these simulation SDN works construct the central controller of SDN in their database station []. The controller transmits control instructions with sensor nodes by multi-hop communications [27]. This causes a great deal of energy consumption [43], which makes it a paradox since sensor network suffering from ought to be an energy-efficient and reliable network.

Moreover, to achieve intelligent WSNs, our observation is that AI methods are only able to be conducted in the SDN controller. This is due to two major reasons. First, traditional sensor devices have limited processing capability and cannot deal with complex computations [37, 18]. Second, the sensor devices are also lack of storage space to support complicated operations. With the help of SDN, the central controller can conduct complex computations and store large-scale data information for WSNs.

We identify UAVs are the

Adler [], a recent testbed that combines WSNs [] and the unmanned aerial vehicle (UAV) [6, 31, 7] shows that UAVs enable WSNs to achieve resilience, high-performance and energy-efficiency at the same time. Specifically, UAVs are flexible to communicate with sensors by one-hop transmission and thus reduce energy consumption compared to multi-hop communications. Moreover, UAVs are capable to perform complex computations by their airborne computers with powerful CPUs and GPUs. Although Alder is not a typical network, it provides a favorable proof that UAVs are perfect to serve as the SDN controller.

Furthermore, the advantage of utilizing UAV compared to 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 re-

spectively 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 Malak<sup>1</sup>, a real UAV based SDN sensor system. In Malak, 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-task, 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-task 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 selection application to choose the node set for the multi-task application.

We use DJI M100 UAV with Intel NUC as airborne computer, and 100 TI CC2650 SensorTag to build our testbed. We evaluated the five applications of Malak on this testbed, compared with their respective baselines LEACH RPL. Our evaluation shows that:

- 1) Malak has high performance. Malak achieves 80% higher throughput than RPL;
- 2) Malak is intelligent in self optimization. Malak use AI selection and diagnosis algorithm to obtain an 80% increment compared with RPL;
- 3) Malak is resilient. As time goes by, Malak maintains over 80% packet received ratio while others drop to 50% sharply.

The main contribution of this paper is the first practical SDN wireless sensor network system, Malak. Malak realizes global optimization in a distributed WSN environment. Other contributions include several simple but effective strategies and algorithms in the implemented applications of Malak. Malak has the potential to promote the adoption of IoTs

The paper is organized as follows. We introduce related works in the next section. Design of Malak is presented in Section 3, and we implement five fundamental applications: routing, network diagnosis, AI prediction, AI node selection and multi-task 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

<sup>1</sup>In the Bible, Malak is a word used to denote a messenger from God.

There are several existing SDN approaches for sensor system, namely flow sensor [25], SDWN [10], sensor openflow [24], TinySDN [11], SDN-WISE [14]. However, they are all implemented and evaluated only by simulations.

**Flow Sensor.** The previous idea presented in paper [25] addresses reliability in the sensor networks through exploiting the OpenFlow technology[26]. Coming up with the concept of flow-sensor instead of typical sensor [23], they have successfully made it possible to achieve communications between controller, gateway and sensors. Besides, they proved flow sensor to be 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 uninterruptible 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 [10]. Given the gap that advantages of SDN and the proper ways to expand it to wireless networks are 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 networks. They made a good attempt to develop the SDWN protocol stack.

**Sensor OpenFlow.** In the paper [24], the group took 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 [11] is a TinyOS-based SDN framework aiming 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-enabled sensor node and SDN-controller node. In order to test the reliability of this framework, the authors conducted 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 [14], and the authors implement the prototype of this idea. Compared to other SDN solutions for wireless network, this solution successfully reduces the message exchange needed between

sensor nodes and controller. Besides, flexible APIs are provided by the authors, which makes the network much easier to program. Also, they implemented some experimental testbed, and found that in a power limited hardware, the response delay was still very small. SDN-WISE shows its good performance in different operation conditions.

## 2.2 Applications for Wireless Sensor Networks

There are already a rich set of the applications for WSNs. However traditional distributed fashions for WSN can not promise global optimization. Worse, no existing work can combine them to run simultaneously. To design a ecosystem for software defined sensor networks, we study five typical applications: routing, network diagnosis, AI sensor, sensor selection and multi-task.

**Routing.** Low-power and lossy network consists of devices whose processing capability, memory and energy are constrained [38]. Hence, traditional protocols cannot be used there. However, RPL is one of the standards for IPv6[12] routing in LLN. It builds the topology in form of a tree, performing DAO, DIO, ACK to generate a directed acyclic graph from one or more roots to the leaf nodes. The experiments done in the paper [39] show 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 are not intermediate enough [34, 20]. Directional Diagnosis [16] 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, thus it can produce a more accuracy prediction [30]. Specifically, the whole approach consists of four main parts: node tracing, trace collection, inference model and incremental probing. The node tracing algorithms dynamically reconstruct the topical topology and derive inner dependencies. Based on the collected data information, inference model can guide the next probe by incremental probing upon belief network. The experiments done show the diagnosis process promises high efficiency in real time.

**AI sensor.** The application of sensors often relays on the fields like sensor selection and data prediction with the help of AI algorithms. In order to select the subset of nodes to be active, related study gives a way to dynamically response to the change of sensor network based on the historical information [28]. The new sensor node has a different time factor from the old node when constructing the predictive network. Besides, the position of nodes may be sufficiently used to reduce the weight of nearby nodes in a predictive network [21]. As

for the data prediction, on the one hand, the energy consumption information can be collected. Previous work on diagnosis, for example, only performs diagnosis process when sensors almost run out of energy. On the other hand, the original data on temperature, humidity, light intensity is a reliable resources for prediction in time series[35].

**Sensor Selection.** Redundant sensor network always contains nodes that generating data stream with some overlapping parts [2]. Collected information can be useful to determine which sensors are active. The algorithms come up with in paper[22] 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 [3] so that their approach can predict more accurately and response much faster.

**Multi-task.** One single sensor node may shoulder several tasks. To save the energy consumption of the network, when a task is assigned, we need to choose which sensor to be active and which to be inactive [15]. Nowadays, works aiming to design an energy-efficient sensor management strategy tends to develop their methods in three steps: choose the proper subset of nodes to be active [1], choose the subset of active nodes to assign the task and set the sampling rate of the task. Related study has found an efficient online algorithm considering sensor activation and task mapping to deal with dynamic events during the runtime of software defined sensor network system [44].

## 3. ARCHITECTURE

### 3.1 Overview

Before introducing our SDN system architecture, we first present some basic terms in SDN system.

**Data plane:** forwarding data and basic sensing functions.

**Control plane:** the protocols used to populate the forwarding tables of data plane elements.

**Management plane:** software services. network policy is defined in management plane, the control plane enforces the policy, and the data plane executes it by forwarding data accordingly. (running applications)

**Southbound Interface (SI):** The instruction set of the forwarding devices is defined by the southbound API, which is part of the southbound interface. Furthermore, the SI also defines the communication protocol between data plane and control plane elements. This protocol formalizes the way the control and data plane elements interact.

**Northbound Interface (NI):** The middleware on control plane can offer an API to application developers. This API represents a northbound interface, i.e.,

a common interface for developing applications. Typically, a northbound interface make abstracts the low-level instruction sets used by southbound interfaces to program forwarding devices.

The architecture of Malak is shown in Fig. 1. The system is divided into three layers: data plane, control plane and management plane respectively.

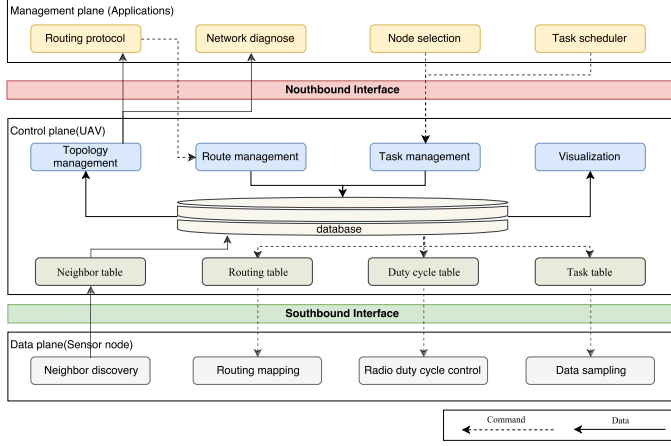


Figure 1: Architecture of the system.

To build software defined network system in WSN and achieve low energy consumption and high performance, the SDN controller ought to configure data plane within as less hops as possible. In Malak, we choose UAV as a mobile SDN controller, which communicates with sensor nodes by on-hop communication. It significantly reduces the energy consumption of nodes in Malak. To achieve high performance, the UAV controller carries a powerful airborne computer which has ability to run some intelligent applications in real-time.

### 3.2 SDN Planes

The **data plane** runs on sensor nodes while the control plane and management plane run on UAV controller. And the interface between data plane and control plane is defined as a southbound interface, while the interface between control plane and management plane is defined as a northbound interface.

In the **control plane**, in order to make our system easy to use, we implement a database on the UAV and design interfaces for data plane and SDN applications. The control plane connected data plane and management plane. The database maintain the topology gathered from the 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 as routing algorithms, network diagnose, sensor task scheduler, etc.

Different from wired network, the data plane of WSN

not only runs data forwarding function, but also executes data sampling function. For example, sensor nodes execute neighbor discovery process for getting topology, data sampling process for getting data from environment and data forwarding process for data gathering. Malak is an easy-to-use system that we design and implement a lot of interfaces for users.

### 3.3 Nouthbound Data Structures

As shown in Fig 3 and Fig 2, the **nouthbound data structure** including downstream and upstream structure fully considered the difference between wired networks and wireless sensor networks, the data structure not only contains routing related data, but also has sensor task control tables.

#### Feedback data structure

##### Neighbor information

| Neighbor list |      |                   |                   |
|---------------|------|-------------------|-------------------|
| IP            | RSSI | Connectivity rate | Traffic statistic |
| IP            | RSSI | Connectivity rate | Traffic statistic |

⋮

##### Energy statistic information

| Items | Radio | Sensor | CPU | Others |
|-------|-------|--------|-----|--------|
| Cost  | Er    | Es     | Ec  | Eo     |

Figure 2: Upstream data structure

The upstream data is input for routing protocol, it including neighbor information collected by node neighbor discovery. When node runs neighbor, it also records the received signal strength indicator (RSSI) of every neighbor nodes. To keep neighbor table freshness, node will execute neighbor discovery periodically base on user defined interval. Moreover, node also records some statistic information including the data traffic and energy consumption for control plane to optimize and diagnose the network.

The downstream data is the configuration data that control plane send to data plane. In wireless sensor networks, the configuration data including routing table, sensor sampling control table and sensor task map, the specific explanation of data structure will discuss in section 4.

### 3.4 Southbound Interface

The Table 1 shows our southbound interface for user write applications. By using our SDN Southbound Interface, users can easily write a variety of applications, such as routing protocol, sensor task scheduler, network diagnose, algorithm evaluator, etc.

Configure data structure

Routing table

|           |             |             |     |             |
|-----------|-------------|-------------|-----|-------------|
| Time slot | T1          | T2          | ... | Tn          |
| Role      | (H/M)       | (H/M)       | ... | (H/M)       |
| Next hop  | Next hop Ip | Next hop Ip | ... | Next hop Ip |

⋮

Sample frequency control:

|                  |    |    |    |    |    |    |    |
|------------------|----|----|----|----|----|----|----|
| Time slot        | T1 | T1 | T3 | T4 | T5 | T6 | T7 |
| Sample frequency |    |    |    |    |    |    |    |

Sensor enable map

|        |    |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|----|
| Sensor | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
| On/Off |    |    |    |    |    |    |    |    |

**Figure 3: Downstream data structure**

We demonstrate two application code using our south-bound interface and write the code in in Section 4.2 and Section 4.6.

## 4. APPLICATIONS IN Malak

### 4.1 Overview

Traditional applications cannot achieve complicated goals and efficiency due to the limited processing power and memory space of sensors.

In Malak, 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-

**Table 1: SDN Southbound Interface**

| Structure && Function                      | Description  |
|--|--|
| <b>Sensor Control Interface</b>            |  |
| struct node                                | Sensor node structure  |
| struct nodeset                             | A set of sensor nodes  |
| struct neighbor_list                       | Neighbor infomation  |
| struct energy_item                         | Energy statistic information   |
| struct rout_table                          | Route 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  |
| <b>UAV Application Interface</b>           |  |
| <b>Routing</b>                             |  |
| get_topology()                             | Get the topology of the network  |
| get_rout_table(node)                       | Get the route table of a node  |
| set_route(node)                            | Set the route of a node  |
| <b>AI Node selection</b>                   |  |
| nodeset simple_selection(nodeset)          | Select sensor set by location information                                  |
| nodeset SRSSS_selection(dataset)           | Select sensor set by AI algorithm based on sensing data                    |
| <b>AI Energy Prediction</b>                |  |
| model_selset(modeltype)                    | Select an AI model   |
| model.train(dataset,radio)                 | 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  |
| <b>Multi-tasks</b>                         |  |
| 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                          |
| <b>Diagnosis</b>                           |  |
| detect()                                   | Detect problematic region with probes                                      |
| get_topical_topology(nodeset)              | Construct topical topology   |
| diagnose_network(topology,nodeset)         | Diagnose the failure nodes or lossy links                                  |

hop communication directly. Thus it enables the sensor network to achieve much more intelligent applications.

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

## 4.2 Routing

### 4.2.1 Motivation

Routing is a fundamental application of SDN system. There are lots of studies on SDN in wired network, but only few researches on wireless sensor networks since the implementation of SDN routing in WSN have the following challenges. First, the wireless communication is unstable, the poor reliability of connection between controller and sensor nodes makes SDN system performance worse, especially the controller connects with nodes through multi-hop wireless approach. Second, the sensor network is energy sensitive, the deployment of SDN downward stream energy cost needs to be ignored, compare with the normal network data communication, and the frequency of routing update should low. Last but not least, achieving the robustness of SDN routing in WSN is challenging because the controller and nodes may be disconnected. When some nodes' radio fails, it may cause the entire network connection crashing.

To address the above challenges, we use a mobile controller which communicates with nodes via one-hop to improve the reliability of nodes and controller connection. Meanwhile, we design a cluster-based SDN routing protocol and a local repair mechanism to achieve both energy efficiency and robustness.

### 4.2.2 Design

In WSNs, cluster routing method is a kind of protocol, which nodes choose a header periodically, the other node around cluster header send it's data to cluster header, and the cluster header forwards it to another header, until the root node receives the data. Though cluster method will reduce the energy hole problems, the routing protocol between is running on sensor nodes locally, because sensor nodes only have local information of network and limited computing ability, it can not achieve global optimal routing, and it is difficult to modify protocol according users requirements.

We improve cluster method by using our SDN controller to update the routing protocol between cluster headers. At first, a UAV flies around the sensor field and get topology data from nodes, and then calculates the cluster header schedule to decide which node should be the cluster header at a specific time phase to balance energy consumption between nodes, and calcu-

late the routing result between current header's topology through Open Shortest Path First (OSPF) protocol base on gathered topology information, In WSN, every node only need send its data to root node, so the OSPF destination internet protocol (IP) has be set as root IP. Through OSPF, UAV get the routing table between cluster header. When UAV fly near the nodes, it send routing table to nodes, including the node role and next hop IP in every time phase.

Considering the economic and energy limitation, UAV can not keep fly above the sensing field. It only fly to sensing field to gather topology after node deployed and update nodes routing on demand. So after the routing has been deployed, the system routing repair mechanism should make sure some nodes failure don't make the network routing crashed.

By designing local routing repair mechanism, our system achieves robustness to radio failure. When a node in cluster member failure, it didn't influences the whole routing between the routing we deployed routing. On other case, when the cluster header failure, if the network don't have local repair mechanism, the whole routing between cluster headers will crash and make the network partial. So we design the local repair mechanism through local header switching.

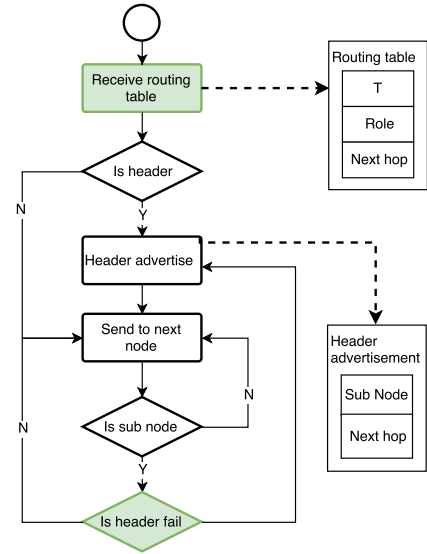


Figure 4: Routing local reparation

As shown in Figure 4, after the node get routing table from UAV, it sets its next hop IP following received table, and also uses its neighbor's RSSI information to find the nearest node as its substitute node, then node broadcast its next hop IP and substitute node IP. After the substitute node received the broadcast packet, it save the source's next hop IP and keep monitoring the source status. The substitute node will became new header when it detected the current header failed.

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

```

topology = get_topology();
//calculate route table for each node
//based on topology
for(node=0; node<nude_num; node++){
    node.routingtable =
        calculate_routable(topology);
}
//set route table for each node
for(node=0; node<nude_num; node++){
    fly_to(node);
    set_route(node.routingtable);
}

```

We implement our software defined routing and local reparation in Malak and demonstrate the code in List 1, the routing in our system achieve both energy efficiency and robustness.

### 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 Malak system. Since in Malak 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 [16], 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 sus-

picious node set. Next we set the UAV to fly through all these nodes and send beacons to them to diagnose the failed nodes. Then UAV collects the neighbor lists of every node 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 Malak releases the complicated inference computations and can achieve accurate diagnosis since the UAV can fly to the sensors to confirm the network faults. Malak 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 Energy Prediction

#### 4.4.1 Motivation

Traditional wireless sensor networks (WSN) generally carry out data collection by multi-hop transmission, and due to limited computation capability and storage capacity, sensors are incapable to perform machine learning methods for data mining. Hence, intelligent self energy consumption optimization for a sensor is hard to achieve.

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

An ideal solution to the energy exhausted problem is to find a self-adjusted routing schedule which makes adjustments according to the energy status of each sensor. Actually, this is a kind of dynamic programming problem and the optimization target is to achieve the longest lifetime of the whole network under complex constrains.

In Malak, the UAV controller stores the global energy information and can perform large-scale computations, laying the foundation for the intelligent energy prediction. The AI prediction in turn can contribute to the whole system network diagnosis application. This kind of cooperation and mutual assistance makes Malak an effective ecosystem.





Figure 5: AI prediction

#### 4.4.2 Design

In Malak, we implement an energy prediction application to forecast lifetime of sensors. The architecture of this application can be seen in Fig. 5.

We utilize the popular machine learning model, Multi-layer Perceptron (MLP) [17], to conduct the energy prediction process. The AI-based prediction architecture is showed in Fig. 5. After the information collected by the sensor is transmitted to database, it is used as the input of the incremental learning model, which outputs the life expectancy of each sensor.

A sample input of energy information contains at least four components: sensors' duty cycle, packet-sending time, data receiving time and other energy consumption. Data collection module records these information with time stamp as well as routing information. We use the database to maintain our experimental data, and use the sql query to manage training data. In this way, we can easily get each sensor's routing flows and energy consumption at any time as input. A forgetting factor is used to avoid overfitting. Then the model will output the prediction of a energy consumption list in time sequences. The sensor which is most likely to run out of energy will rank in the first place of the list. Combined with our diagnosis and routing application, we can make use of our prediction result to prolong the network lifetime.

### 4.5 AI Node Selection

#### 4.5.1 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 Malak, 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.

#### 4.5.2 Design

Our Malak 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 Malak. The evaluations in section 6 wshow it greatly saves the sensors' energy and thus prolongs the network lifetime.

---

#### Algorithm 1 Greedy Selection Algorithm

---

- 1: Input: Sensor set  $N$ , Selected set  $M$ , Target area  $\Omega$ , Covering area  $\Phi$ ;
  - 2: Initialize :  $M = \emptyset$ ,  $\Phi = \emptyset$
  - 3: **while**  $M \neq N$  **do**
  - 4:   **if**  $\Phi = \Omega$  **then**
  - 5:     break; \\ Selected set has been found
  - 6:   **end if**
  - 7:   **if**  $\forall n_i \in (N - M) : range(n_i) \subset \Phi$  **then**
  - 8:     break; \\ Cannot cover the target area;
  - 9:   **end if**
  - 10:   Find  $n_i : argmax(\Phi \cap range(n_i))$ ,  $n_i \in (N - M)$ ;
  - 11:    $\Phi = \Phi \cup n_i$
  - 12: **end while**
  - 13: 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) [22].



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 Malak, 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:

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

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 [22]. With the AI sensor selection process, Malak becomes a smarter and adaptive sensor system.

## 4.6 Multi-task

### 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-task due to the limitations of computation complexity for task arrangement of each node.

In our Malak system, we implement the multi-task 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 network is usually assigned with different data collection requirement. In Malak, we design and implement multi-task application and provide easy-to-use interfaces to users.

When a user assigns a task to Malak, the UAV controller will first check out the energy and storage constraints of the required sensor set, as described in Alg. 2. If the task requirement exceeds the capacity of any selected sensor node, it will be sent to a task queue; Otherwise it will be put into the task buffer to conduct.

---

#### Algorithm 2 Sensor Constraint Detection

---

```

1: Input: Sensor set  $N$ , Task  $T$ ;
2: Input: Remaining Energy for each node  $:Energy(u_i)$ ;
3: Input: Remaining Storage for each node  $:Energy(u_i)$ ;
4: Initialize: Energy capacity  $\eta$ , Storage capacity  $\xi$ ;
5: Calculate the energy cost of  $T$  for each sensor as  $:\phi$ ;
6: Calculate the storage cost of  $T$  for each sensor as  $:\varphi$ ;
7: for Each node  $u_i$  in  $N$  do
8:   if  $Energy(u_i) + \phi \geq \eta$  or  $Storage(u_i) + \varphi \geq \xi$  then
9:     Set  $T$  to task queue;
10:  end if
11: end for
12: Set  $T$  to task buffer;

```

---

We implement AI node selection and multi-task in Malak, and demonstrate the code in List 2.

#### Listing 2: An example of AI selection and Multi-tasks

```

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

    for(task=taskset.head; task < taskset.len;
        ↪ task=task.next)
        scheduler.task_buffer_add(
            task,

```

**Table 2: Task Buffer**

| Task ID | Node set                                   | Sensing rate | Sensing range | Sensing duration |
|---------|--|--------------|---------------|------------------|
| 010     | $\{u_2, u_6, \dots, u_{88} \dots u_{99}\}$ | 0.2          | 4             | 2046             |
| ...     | ...  | ...          | ...           | ...              |
| 101     | $\{u_3, u_4, \dots, u_{34} \dots u_{67}\}$ | 0.2          | 5             | 4096             |
| ...     | ...  | ...          | ...           | ...              |

```

        defaultset);

    scheduler.task_update();
    ...
    ...
    while((task = scheduler.task_next) != NULL){
        data = get_collected_data();
        nodeset =
            SRSSS_selection(dataset);
        scheduler.task_buffer_update(
            task,
            nodeset);
    }
    scheduler.task_update();
}

```

Each task has a unique task ID. The task buffer maintains the task information of node set, sensing rate, sensing range and sensing duration. Every time a task is removed or updated by the user, or it runs out of its duration time, the task queue will check to release new tasks into the task buffer.

## 5. IMPLEMENTATION

### 5.1 Data Plane

We run modified contiki-ng, a system for next generation IoT (Internet of Things), on data plane. We modify the network layer of contiki-ng, and abstract data plane interface to separate network control functions from data plane. The details are described as follows.

**Neighbor Discovery:** After deployment, nodes first start IPv6 neighbor discovery protocol to find the nodes which can be reached within one hop, and then save the neighbor information to neighbor table and wait for UAV arriving to gather the topology information. Every neighbor's information consists of the neighbor's IP and RSSI.

**Packet Forwarding and Data Sampling:** The nodes get routing table from UAV through northbound interface. When a node receive a data packet from other nodes, it just sends it to the next hop node according the routing table.

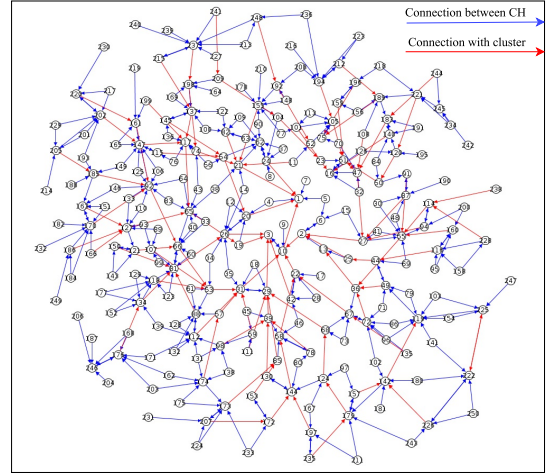
Nodes read task table received from UAV to decide when to wake up sensors to make sure that they cost minimal energy to get the same quality data.

### 5.2 Control Plane

We run our SDN control plane a power airborne computer carried by UAV. The UAV's flight controller runs on ROS (Robot Operating System) base on UAV's SDK. And we use MySQL as our SDN database. UAV gathers topology from each node and save it into database, so that users can easily read the whole network data from database. For the same reason, after a user gets routing algorithm results and saves it into database via northbound interface, our control plane will automatically divides the whole network routing results into individual node's routing table, and then sends it to specific nodes.

### 5.3 Hardware Platform

Based on our software framework, we use DJI M100 UAV carrying Nvidia TX-2 airborne computer, and adapt 250 TI CC2650 SensorTag to build our testbed. The TX-2 built around an NVIDIA GPU and loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth. It is most power-efficient embedded AI computing device which is suitable for UAV onboard computing. The SensorTag communicate with UAV and other nodes through Zigbee stack, it also equip with 10 different kind of sensors which is suitable for large scale wireless sensor networks.

**Figure 6: Network Run-time topology**

According to our UAV gathered network run-time topology, we draw Figure 6, the blue lines represent the connection between cluster members with cluster headers, and the red lines donate the route between cluster

headers deployed by UAV.

## 6. EVALUATION

### 6.1 Evaluation Setup

In this section, we describe our performance evaluation in simulation and real-scene experiments. We distribute 100 TI SensorTag sensors randomly deploy in a  $250 \times 250$  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. When testing the network, we make nodes send packets every 10 seconds, and the packet size is 60 bytes. Unless otherwise specified, we set the evaluation time from 0 to 5000 seconds.

### 6.2 Evaluation Metrics

**Packet received ratio** is defined as the proportion of the total data packets received by data center and the the total data packets sent by all nodes, we adopt the model in [8] and it can be formulated as

$$L = \frac{\sum_{i=0}^N S_i}{R} \quad (2)$$

where  $L$  represents the throughput, and  $S_i$  and  $R$  denotes the number of packets sent by the  $i$ -th node and the number of packets received by data center, respectively.

**Throughput** is defined as the the total data packets received by data center in unit time, and it measures the capability and scalability of a network.

**Latency** is define as the time interval between node sent data package and data center received the package.

**Network Lifetime** is define as the elapsed time when total package received ratio is less than 50%, which is an important metric to measures the reliability of network.

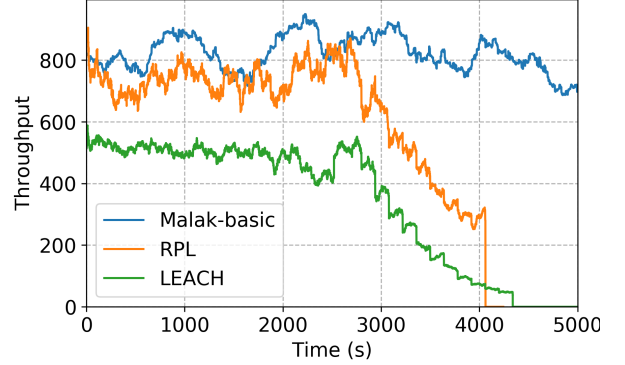
**Energy Consumption** is estimated by multiplying different coefficients on CPU running time and radio listening and transmitting time and summing them up. The coefficients are proportional to the working current described in sensor's data sheet. Average energy consumption is the energy consumption of a node in unit time.

### 6.3 Basic Routing

In order to fairly compare with existing works, we first only use our basic Malak routing protocol without any intelligent approaches. We evaluate routing performance by throughput, package received ratio and end-to-end latency. Meanwhile, We also evaluate the scalability of our network through network lifetime.

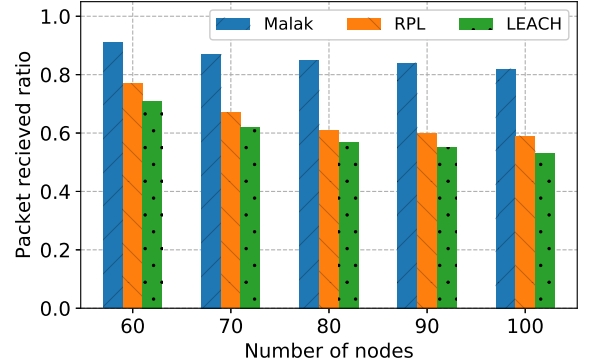
Figure 7 compares network throughput by deploying various routing algorithms. As our Malak uses OSPF, which finds the optimal path from sensors to data center, to calculate the route table, its throughput ex-

ceeds RPL and LEACH. Besides, in Malak, sensors' lifetime increases a lot more than others, because all computation-intensive tasks are done by UAV and sensors do not need to send packets to negotiate route path, which decrease the energy consumption with no doubt.



**Figure 7: Routing throughput.** The figure shows throughput corresponding to three kinds of Routing algorithms. Throughput can drop to zero when some key nodes are dead in the network.

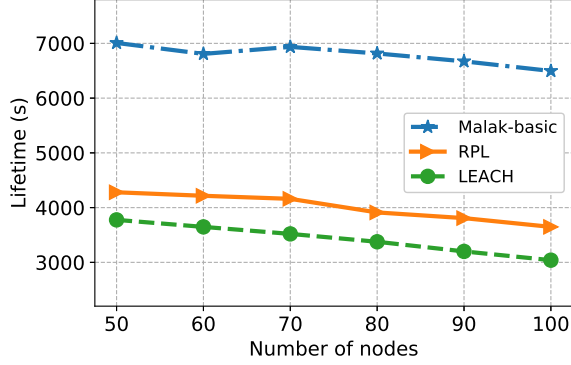
Figure 8 shows two remarkable advantages of Malak: (1) The packet received ratio exceeds state-of-the-art algorithms including RPL [42] and LEACH [19] by 20%; (2) With the increasing of network size, the Malak's packet received ration is still much higher than others'. The reasons are that Malak uses global information to achieve optimal routing and intelligent cluster header selection balances the energy consumption between nodes.



**Figure 8: Packet received ratio improved by Malak**

Our Malak system is capable in large scale wireless sensor networks as Figure 9 shows. Its lifetime surpasses RPL and LEACH in both small and large network sizes.

When testing the latency of the network, our Malak-basic routing method performs best due to using the global topology information to find the optimal route

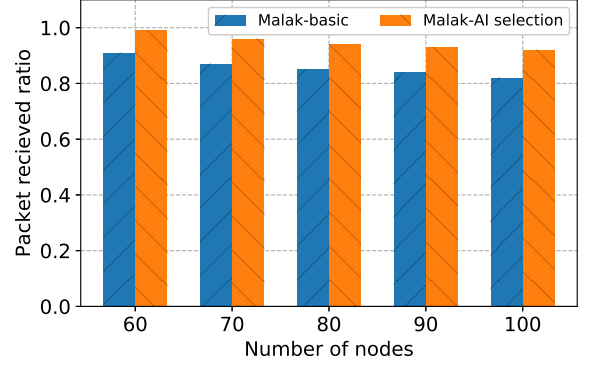


**Figure 9: Network lifetime using various routing algorithm** Malak succeeds RPL and leach in different network sizes.

path. As shown in Figure 10, as the distance from node to data center increases, the latency grows about 4 times and 1.5 times slower than LEACH and RPL, respectively.

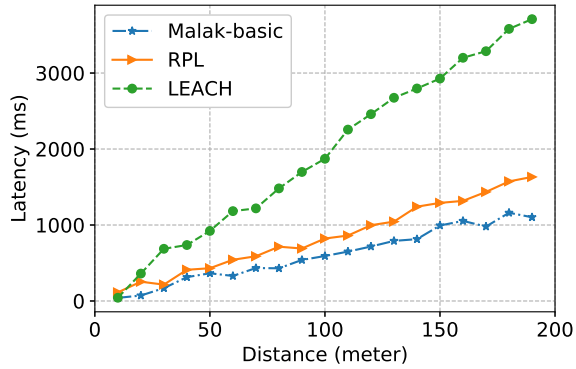
#### 6.4 Intelligence

By deploying intelligent node selection algorithm, we use least waken sensors but collect most necessary sensing data covering the whole area and reduce the redundant ones in the meantime. Besides, less sensors means less collision and interference which further improved the reliability of network. And the packet received ratio achieves almost 100%.

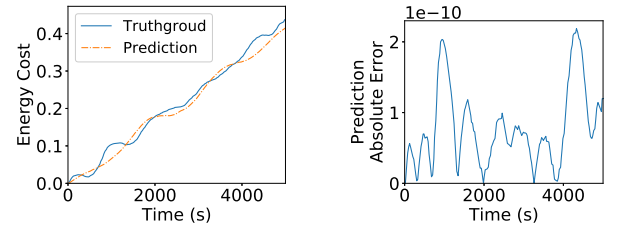


**Figure 11: Packet received ratio improved by intelligent node selection**

By deploying MLP on UAV, Malak can predicts the energy consumption precisely as Figure 11 shows, and the absolute error can be restricted in  $3 \times 10^{-10}$ , which enable us to predict and fix energy failure in advance.



**Figure 10: Latency of different routing algorithms and distance**



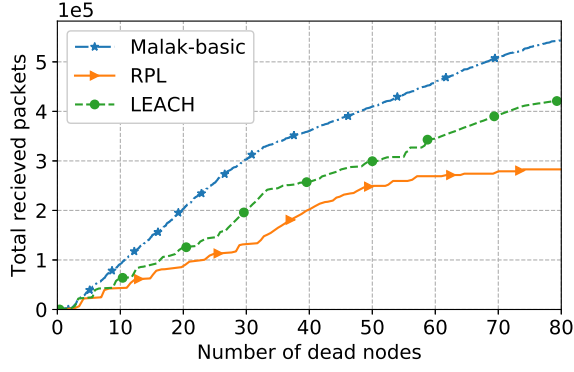
(a) Energy prediction

(b) Prediction error

**Figure 12: Energy prediction** We use regression method to predict energy consumption and the prediction result is consistent with the truth-ground. The absolute error of energy prediction never exceeds  $3 \times 10^{-10}$  with normalized energy (i.e. the maximum energy of a sensor is 1).

#### 6.5 Resilience

We compare the impact caused by death of nodes using throughput among LEACH, RPL and Malak and the results are shown in Figure 13. The total received



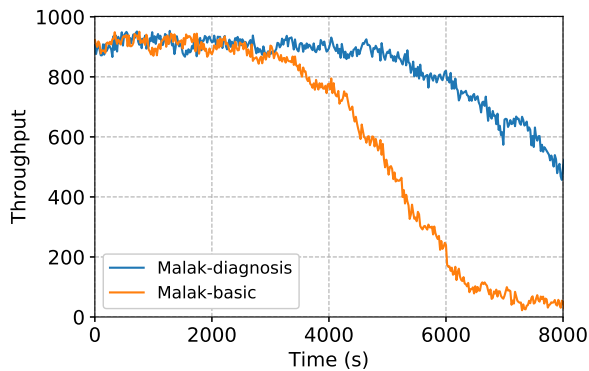
**Figure 13: Fault tolerance** RPL and Malak are compared on resilience, and Malak perform slightly better than RPL.

packets of Malak increases stably due to its resilience to faults of nodes. The resilience is achieved by 2 approaches: (1) cluster based protocol is immune from failures of cluster members; (2) Local repair enable the network recover from failures of cluster headers.

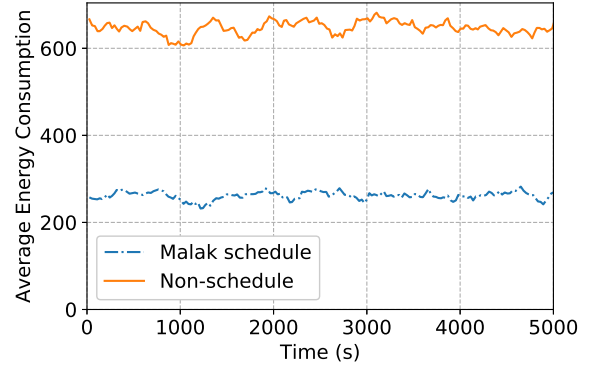
**Table 3: Failure detection rate for various failure type and number**

| Type    | Failures |     |     |     |     |
|---------|----------|-----|-----|-----|-----|
|         | 1        | 2   | 3   | 4   | 5   |
| Sensing | 98%      | 95% | 93% | 92% | 90% |
| Energy  | 100%     | 93% | 92% | 90% | 88% |
| Radio   | 100%     | 90% | 88% | 84% | 80% |

Table 3 shows the failure detection rate among sensing, energy and radio failures. Controller proactively adjusts the routing based on the preliminary diagnosis results, which make it possible to decrease the routing failure before it happens and improve the throughput as shown in Figure 14.



**Figure 14: The performance improvement of automatic network repair**



**Figure 15: Average energy consumption with and without multi-task schedule** When with multi-task schedule, sensor consumes half of the energy comparing to when without multi-task schedule.

By applying multi-task schedule, we reduce the working sensors on one node and the size of transmitted packets to cut down the energy consumption of sensing and communication over 50%.

## 7. CONCLUSION

In this paper, we present Malak, the first practical SDN wireless sensor network system, which achieves intelligence, high-performance and resilience at the same time for general applications. Five typical applications: routing, network diagnosis, AI energy prediction, AI node selection and multi-task are realized in Malak. Our experiments show that Malak provides easy-to-use interfaces and it realizes global optimization and can further promote the deployment of IoTs.

## 8. REFERENCES

- [1] AGHDASI, H. S., BISADI, P., MOGHADDAM, M. E., AND ABBASPOUR, M. High-resolution images with minimum energy dissipation and maximum field-of-view in camera-based wireless multimedia sensor networks. *Sensors* 9, 8 (2009), 6385–6410.
- [2] ALI, Y., AND NARASIMHAN, S. Redundant sensor network design for linear processes. *Aiche Journal* 41, 10 (1995), 2237–2249.
- [3] ASTROM, K. J., AND WITTENMARK, B. Adaptive control. *Technometrics* 33, 4 (1989).
- [4] ATZORI, L., IERA, A., AND MORABITO, G. The internet of things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805.
- [5] BENZEKKI, K., FERGOUGUI, A. E., AND ELALAOUI, A. E. Software-defined networking (sdn): a survey. *Security & Communication Networks* 9, 18 (2016), 5803–5833.

- [6] CAMBONE, S. Unmanned aircraft systems roadmap 2005-2030. *United States.dept.of Defense.office of the Secretary of Defense* (2005).
- [7] CATHCART, G. P., BARBER, T. A., AND WORTH, D. R. Method of controlling operation of an unmanned aerial vehicle, 2014.
- [8] CHEN, Z., LIU, A., LI, Z., CHOI, Y.-J., SEKIYA, H., AND LI, J. Energy-efficient broadcasting scheme for smart industrial wireless sensor networks. *Mobile Information Systems 2017* (2017).
- [9] COCKBURN, D., AND JENNINGS, N. R. Archon: A distributed artificial intelligence system for industrial applications. In *Foundations of Distributed Artificial Intelligence* (1996), pp. 319–344.
- [10] COSTANZO, S., GALLUCCIO, L., MORABITO, G., AND PALAZZO, S. Software defined wireless networks: Unbridling sdn. In *Software Defined Networking (EWSDN), 2012 European Workshop on* (2012), IEEE, pp. 1–6.
- [11] DE OLIVEIRA, B. T., GABRIEL, L. B., AND MARGI, C. B. Tinsydn: Enabling multiple controllers for software-defined wireless sensor networks. *IEEE Latin America Transactions* 13, 11 (2015), 3690–3696.
- [12] DEERING, S., AND HINDEN, R. *Internet Protocol, Version 6 (IPv6) Specification*. RFC Editor, 1998.
- [13] FEAMSTER, N., REXFORD, J., AND ZEGURA, E. *The road to SDN: an intellectual history of programmable networks*. ACM, 2014.
- [14] 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.
- [15] GEORGES, D. Energy minimization and observability maximization in multi-hop wireless sensor networks. In *World Congress* (2011), pp. 13918–13923.
- [16] 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.
- [17] HARVEY, R. L. An introduction to multi-layer perceptrons. *Neural Networks in A Softcomputing Framework* 6, 3 (1988), 265–277.
- [18] HELLER, B., SHERWOOD, R., AND MCKEOWN, N. The controller placement problem. *Acm Sigcomm Computer Communication Review* 42, 4 (2012), 473–478.
- [19] KAUR, S. P. Wsn’s leach protocol: A survey. *Wireless Communication* 8, 3 (2016), 113–116.
- [20] KHAN, M. M. H., LE, H. K., LEMAY, M., MOINZADEH, P., WANG, L., YANG, Y., DONG, K. N., ABDELZAHER, T., GUNTER, C. A., AND HAN, J. Diagnostic powertracing for sensor node failure analysis. In *ACM/IEEE International Conference on Information Processing in Sensor Networks* (2010), pp. 117–128.
- [21] KUMAR, S., SPEZZANO, F., SUBRAHMANIAN, V. S., AND FALOUTSOS, C. Edge weight prediction in weighted signed networks. In *IEEE International Conference on Data Mining* (2017), pp. 221–230.
- [22] 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).
- [23] LIU, J. *Thermoresistive Flow Sensors*. Springer New York, 2015.
- [24] 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.
- [25] 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.
- [26] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow:enabling innovation in campus networks. *Sigcomm Computer Communication Review* 38, 2 (2008), 69–74.
- [27] MIZUYAMA, K., TAENAKA, Y., AND TSUKAMOTO, K. Estimation based adaptable flow aggregation method for reducing control traffic on software defined wireless networks. In *IEEE International Conference on Pervasive Computing and Communications Workshops* (2017), pp. 363–368.
- [28] MO, H., AND MUSSELLE, C. J. Dynamic collaborative change point detection in wireless sensor networks. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* (2013), pp. 332–339.
- [29] NORVIG, P., AND RUSSELL, S. J. Artificial intelligence :a modern approach. *Applied Mechanics & Materials* 263, 5 (1995), 2829–2833.
- [30] PAVLOU, M., AMBLER, G., SEAMAN, S. R., GUTTMANN, O., ELLIOTT, P., KING, M., AND OMAR, R. Z. How to develop a more accurate risk prediction model when there are few events.
- [31] PERRY, C. D. Unmanned aerial vehicle: A tool for the operational commander. *Unmanned Aerial*



- Vehicle A Tool for the Operational Commander* (2000).
- [32] POOLE, D. L., AND MACKWORTH, A. K. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, 2010.
  - [33] POTDAR, V., SHARIF, A., AND CHANG, E. Wireless sensor networks: A survey. In *International Conference on Advanced Information NETWORKING and Applications Workshops* (2009), pp. 636–641.
  - [34] RAMANATHAN, N., CHANG, K., KAPUR, R., GIROD, L., KOHLER, E., AND ESTRIN, D. Sympathy for the sensor network debugger. In *SENSYS* (2005), pp. 255–267.
  - [35] RAZA, U., CAMERRA, A., MURPHY, A. L., PALPANAS, T., AND PICCO, G. P. Practical data prediction for real-world wireless sensor networks. *IEEE Transactions on Knowledge & Data Engineering* 27, 8 (2014), 2231–2244.
  - [36] SEZER, S., SCOTT-HAYWARD, S., CHOUHAN, P. K., FRASER, B., LAKE, D., FINNEGAN, J., VILJOEN, N., MILLER, M., AND RAO, N. Are we ready for sdn? implementation challenges for software-defined networks. *IEEE Communications Magazine* 51, 7 (2013), 36–43.
  - [37] SHARMA, G. Security frameworks for wireless sensor networks-review. In *International Conference on Communication, Computing & Security* (2012), pp. 978–987.
  - [38] THUBERT, P., WINTER, T., BRANDT, A., HUI, J., KELSEY, R., LEVIS, P., PISTER, K., STRUIK, R., VASSEUR, J. P., AND ALEXANDER, R. Rpl: Ipv6 routing protocol for low power and lossy networks. *Internet Requests for Comment rfc 6550*, 5 (2012), 853–861.
  - [39] TSIFTES, N., ERIKSSON, J., AND DUNKELS, A. Low-power wireless ipv6 routing with contikirpl. 406–407.
  - [40] WANG, Y., WANG, Y., QI, X., XU, L., CHEN, J., AND WANG, G. L3sn: A level-based, large-scale, longevous sensor network system for agriculture information monitoring. *Wireless Sensor Network* 2, 9 (2010), 655–660.
  - [41] WERNERALLEN, G., LORINCZ, K., WELSH, M., MARCILLO, O., JOHNSON, J., RUIZ, M., AND LEES, J. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing* 10, 2 (2006), 18–25.
  - [42] WINTER, T. Rpl: Ipv6 routing protocol for low-power and lossy networks.
  - [43] YANG, M., LI, Y., JIN, D., WU, X., WU, X., AND VASILAKOS, A. V. Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks & Applications* 20, 1 (2015), 4–18.
  - [44] ZENG, D., LI, P., GUO, S., MIYAZAKI, T., HU, J., AND XIANG, Y. Energy minimization in multi-task software-defined sensor networks. *IEEE Transactions on Computers* 64, 11 (2015), 3128–3139.
  - [45] ZILBERMAN, N., WATTS, P. M., ROTSOS, C., AND MOORE, A. W. Reconfigurable network systems and software-defined networking. *Proceedings of the IEEE* 103, 7 (2015), 1102–1124.