Malak: an intelligent software defined wireless sensor system

ABSTRACT

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

1. INTRODUCTION

Software defined network (SDN) [4, 30] and artificial intelligence (AI) [24, **?**, **?**] 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 [37]. It shows remarkable advantages in good management of network control, less operating costs and easy deployment for new applications [10]. 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) [8, 32]. 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 [31, 15]. 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 [21, 7, 20, 9, 11] 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 [22]. This causes a great deal of energy consumption [35], 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 to support one-hop communication, and finally find the unmanned aerial vehicle (UAV) [5, 26, 6] perfect to serve as the SDN controller. The flexibility of UAV can promise one-hop communication between controller and nodes [29]. 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 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 Malak[1], 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

[1]In the Bible, Malak is a word used to denote a messenger from God.

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 implemented Malak 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, Malak. We adopt AI tools in Malak and realize five applications with easy-to-use interfaces. We then conduct practical experiments and the evaluations validate the high performance of Malak.

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

There are several existing SDN approaches for sensor system, namely flow sensor [21], SDWN [7], sensor openflow [20], TinySDN [9], SDN-WISE [11]. However, they are all implemented and evaluated by simulations.

Flow Sensor. The previous idea presented in paper [21] addresses reliability in the sensor networks through exploiting the OpenFlow technology[**?**]. Coming up with the concept of flow-sensor instead of typical sensor [19], 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 [7]. 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 [20], 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 [9] 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 [11], 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

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-task.

Routing. Low-power and lossy network consists of devices whose processing capability, memory and energy are constrained [33]. Hence, traditional protocols cannot be used there. However, RPL is one of the standards for ipv6 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 [34] 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 [27, 16]. Directional Diagnosis [13] 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 [25]. 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 [23]. 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 [17]. 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[28].

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[18] 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 [12]. 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 [36].

3. ARCHITECTURE

Before introduce our SDN system architecture, we first lead into 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 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.

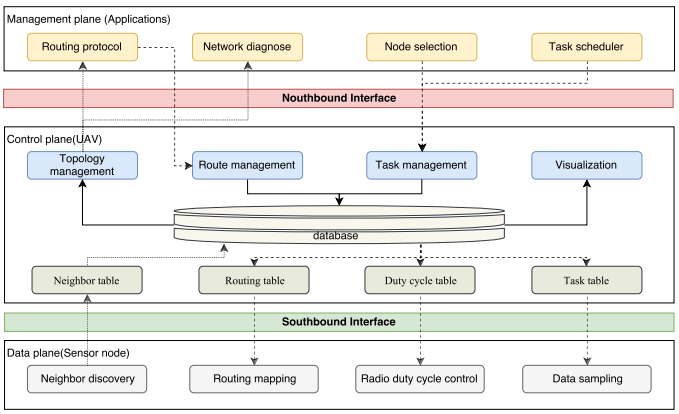


Figure 1: Architecture of the system.

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.

The **data plane** runs on sensor nodes while the control plane and management plane run on UAV con-

troller. And the interface between data plane and control plane is defined as southbound interface, while the interface between control plane and management plane is defined as southbound 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. In Malak is a easy-to-use system that we design and implement a lot of interfaces for users.

As shown in Fig 2 and Fig 3, the southbound data 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.

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 2: Downstream data structure

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

We demonstrate two application code using our southbound interface. The List. 1 shows a deployed routing algorithm based on our SI. And the List. 2 demonstrate a AI selection and Muti-tasks application.

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 3: Upstream data structure

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);
}
```

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

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

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

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 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-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 implementation of SDN routing in WSN have following challenges. First, the wireless communication is unsta-

Table 1: SDN Southbound Interface

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 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
UAV Application Interface	
Routing	
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
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

ble, the poor reliability of connection between controller and sensor nodes make SDN system performance worse, especially the controller connect with nodes through multi-hop wireless approach. Second, the sensor network is energy sensitive, the deployment of SDN downward stream energy cost need be ignore compare with the normal network data communication, and the frequency of routing update should low. Last but not least, the robust of SDN routing mechanism in WSN is difficult to achieve, since the controller and nodes not may be disconnected, if some nodes radio failure, it may cause all network connection crash.

To address above challenges, we use mobile controller which communicate with nodes via one-hop to improve the reliability of nodes and controller connection. Meanwhile, we design cluster-base SDN routing protocol and 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 forward it to anther header, until root node receive the data. Through cluster method will reduce the energy hole problems, but 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, UAV fly around sensor field and get topology data from nodes, and then calculate the cluster header schedule do decide which node should be cluster header at specific time phase to balance energy consumption between nodes, and calculate 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.

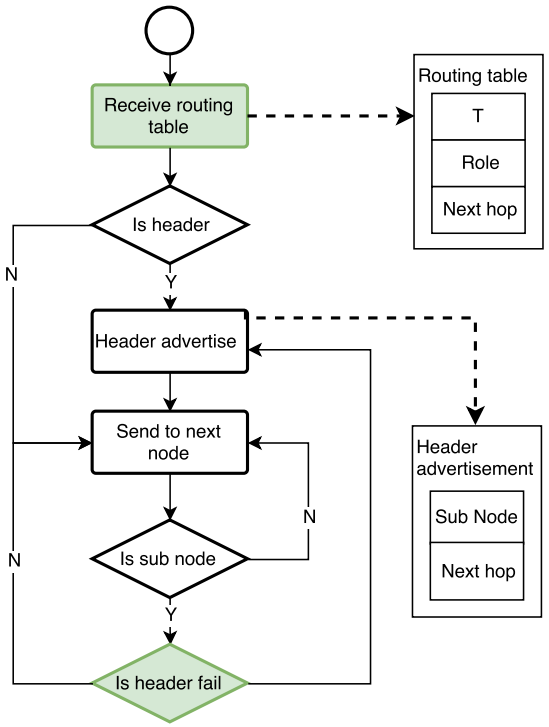


Figure 4: Routing local reparation

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.

We implement our software defined routing and local reparation, 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 [13], 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 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 deep 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 term can contribute to the whole system network diagnosis application This kind of cooperation and mutual assistance makes Malak an effective ecosystem.

These predicted information contributes to other application.

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, Multilayer Perceptron (MLP) [14], to conduct the energy prediction process. The AI-based prediction architecture enables users to analyze the sensor data in real time. 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

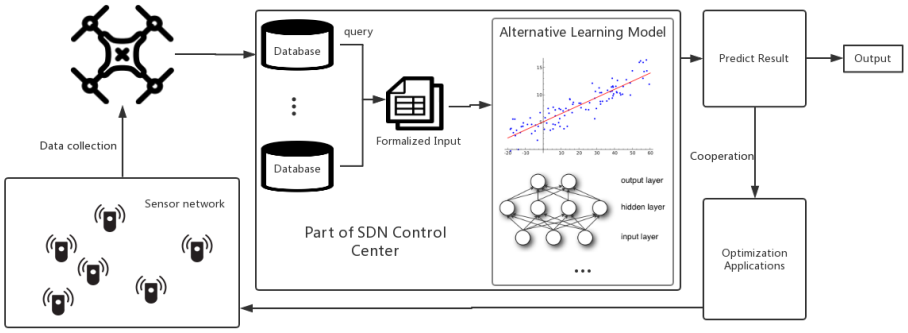


Figure 5: AI prediction

energy will rank in the first place of the list. Combined with our diagnosis 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.

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) [18].

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.

Algorithm 1 Greedy Selection Algorithm

1: Input: Sensor set N, Selected set M, Target area Ω, Covering area Φ;
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 $n_i : argmax(\Phi \cap range(n))$, $n_i \in (N - M)$;
11: $\Phi = \Phi \cup n_i$
12: **end while**
13: Output: M;

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:

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

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 [18]. 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 networks are 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 assign 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. **??**. 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-task.

Sensors are assigned tasks to monitor a specific area. Different tasks have different requirements, i.e.

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

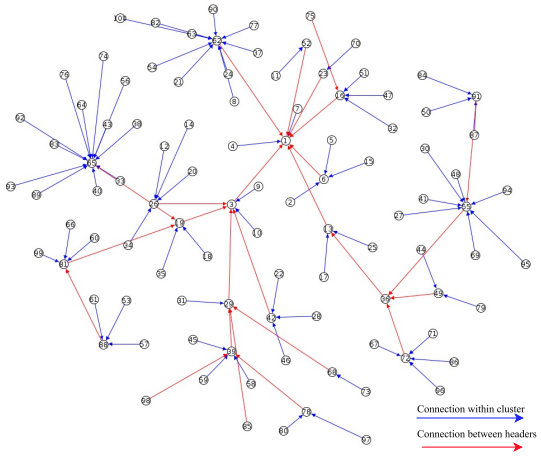


Figure 6: Network topology

Based on our software framework, we using DJI M100 UAV with Intel NUC as airborne computer, and 100 TI CC2650 SensorTag build our testbed. According our UAV gathered topology, we draw Fig. 6, the bule lines represent the connection between cluster member with cluster header, and the red lines donate the routing the connection between cluster headers deployed by UAV.

6. EVALUATION

In this section, we describle our performance evaluation in simulation and real-scene experiments. We distribute 130 TI SensorTag sensors in an about 250 × 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

Packet loss ratio

Packet loss ratio is defined as the proportion of the total data packets recieved by data sink and the the total data packets sent by all nodes, and it can be formulated as

$$L = \frac{\sum_{i=0}^{N} S_i}{R} \tag{2}$$

where L reprents the throughput, and S_i and R denotes the number of packets sent by the i-th node and the number of packets recieved by data sink, respectively.

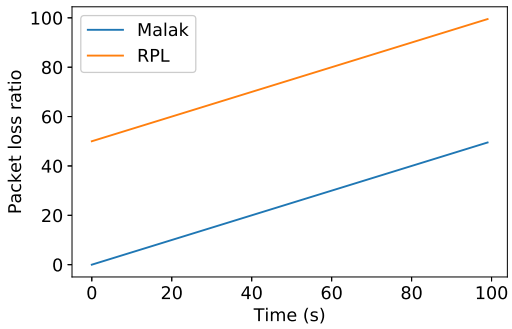


Figure 7: Packet loss ratio in time

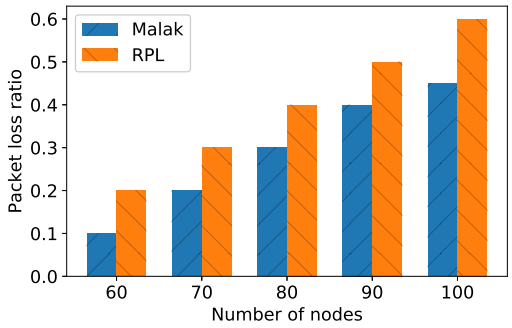


Figure 8: Packet loss ratio in various network sizes

Total throughput

Throughput is defined as the the total data packets recieved by data sink, and it measures the capability and scalability of a network.

Figure 9 compares network throughput by deploying various routing algorithms. As our Malak uses OSPF, which finds the optimal path from sensors to data sink, to caculate the route table, its throughput exceeds RPL

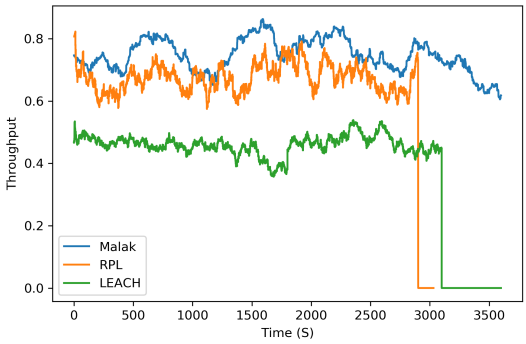


Figure 9: Routing throughput. The figure shows throughput corresponding to three kinds of Routing algorithms. Throughput can drop to zero when some key nodes are dead in the network.

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.

Fault tolerance

Fault tolenrance is a significant metric to evaluate the performance of routing algorithms. It measures the throughput with various number of node died. Throughput can drop sharply if a node which most traffic passes by dies and can be influenced only slightly by death of nodes which no traffic passes through.

We compare the impact caused by death of nodes using throughput between RPL and Malak and the results are shown in Figure 10. Throughput can be low when the number of death is small, because we distribute the sensors densely and the collision problem is severe. And the throughput grows as some node died. But after some key nodes died, the throughput falls down quickly, however, our Malak still performs better than PRL.

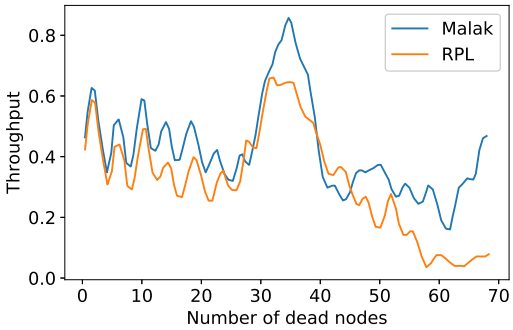


Figure 10: Fault tolerance RPL and Malak are compared on fault tolerance, and Malak perform slightly better than PRL.

Scalability

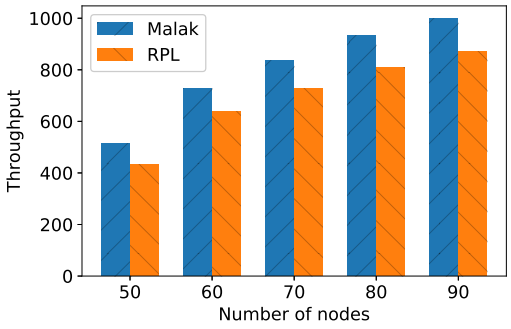


Figure 11: Scalability Malak succeeds RPL in different network sizes.

6.2 Network Diagnosis

Robustness

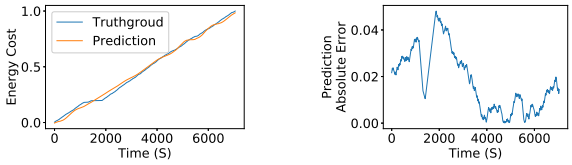
6.3 AI Node Selection

Scalability

6.4 AI Energy Prediction

We estimate sensor energy consumption 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.

Prediction accuracy



(a) Energy prediction (b) Energy prediction error

Figure 12: Energy prediction We use regression method to predict energy consumption and the prediction result is consistent with the truthground. The absolute error of energy prediction never exceeds 0.05 with normalized energy (i.e. the maximum energy of a sensor is 1).

6.5 Multi-tasks

Energy Performance Scalability

7. CONCLUSION

Conclusion goes here.

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

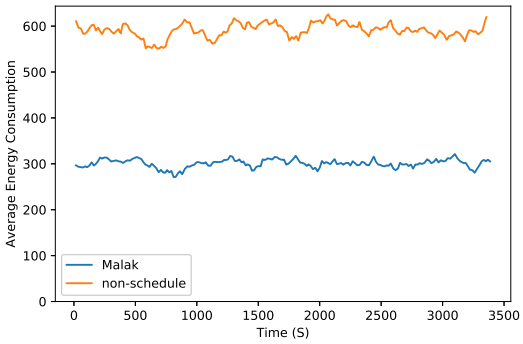


Figure 13: Average energy consumption with and without multitask schedule When with multitask schedule, sensor consumes half of the energy comparing to when without multitask schedule.

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), 2237âĂŞ2249.

[3] Astrom, K. J., and Wittenmark, B. Adaptive control. *Technometrics 33*, 4 (1989).

[4] Benzekki, K., Fergougui, A. E., and Elalaoui, A. E. SoftwareâĂŘdefined networking (sdn): a survey. *Security & Communication Networks 9*, 18 (2016), 5803–5833.

[5] Cambone, S. Unmanned aircraft systems roadmap 2005-2030. *United States.dept.of Defense.office of the Secretary of Defense* (2005).

[6] Cathcart, G. P., Barber, T. A., and Worth, D. R. Method of controlling operation of an unmanned aerial vehicle, 2014.

[7] 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.

[8] Dargie, W., and Poellabauer, C. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley Publishing, 2010.

[9] 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.

[10] Feamster, N., Rexford, J., and Zegura, E. *The road to SDN: an intellectual history of programmable networks*. ACM, 2014.

[11] 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.

[12] Georges, D. Energy minimization and observability maximization in multi-hop wireless sensor networks. In *World Congress* (2011), pp. 13918–13923.

[13] 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.

[14] Harvey, R. L. An introduction to multi-layer perceptrons. *Neural Networks in A Softcomputing Framework 6*, 3 (1988), 265âĂŞ277.

[15] Heller, B., Sherwood, R., and Mckeown, N. The controller placement problem. *Acm Sigcomm Computer Communication Review 42*, 4 (2012), 473–478.

[16] 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.

[17] 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.

[18] 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).

[19] Liu, J. *Thermoresistive Flow Sensors*. Springer New York, 2015.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] Norvig, P., and Russell, S. J. Artificial intelligence :a modern approach. *Applied Mechanics & Materials 263*, 5 (1995), 2829–2833.

[25] 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.

[26] Perry, C. D. Unmanned aerial vehicle: A tool for the operational commander. *Unmanned Aerial Vehicle A Tool for the Operational Commander* (2000).

[27] 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.

[28] 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.

[29] Sara, M., Jawhar, I., and Nader, M. A softwarization architecture for uavs and wsns as part of the cloud environment. 13–18.

[30] 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.

[31] Sharma, G. Security frameworks for wireless sensor networks-review. In *International Conference on Communication, Computing & Security* (2012), pp. 978–987.

[32] Sohraby, K., Minoli, D., and Znati, T. Wireless sensor networks: Technology, protocols, and applications. pp. pÃągs. 398–410.

[33] 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.

[34] Tsiftes, N., Eriksson, J., and Dunkels, A. Low-power wireless ipv6 routing with contikirpl. 406–407.

[35] 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.

[36] 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.

[37] 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.