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 in simulators and there is no practical implementation.

In this paper, we present Malak, the first practical SDN wireless sensor network system, which achieves intelligence, high-performance and resilience simultaneously. In Malak, we utilize a set of unmanned aerial vehicles (UAVs) to serve as the SDN controllers. UAVs communicate with sensor 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. Malak is a ecosystem that all applications can run simultaneously to benefit each other. Our evaluations show the throughput improves by 1.2 times compared with two notable protocols. Meanwhile, the average lifetime of sensor nodes grows by 81.2%. By running energy prediction application, Malak further increases the lifetime by 83.9%. All source code and results of Malak are ready to open source.

1. INTRODUCTION

Software defined network (SDN) [6, 41] paradigm enables flexible programmability for networks by using programmable data plane and centralized network controller [52]. It shows remarkable strengths in network control, global optimization (e.g. throughputs [3], energy consumption [31], network diagnosis [19]) and easy deployment for new applications [16].

With the support of SDN for networks, programmable artificial intelligence (AI) techniques [34, 37, 10] are easy to be implemented in networks [29]. 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[29], e.g. network throughput [3], energy consumption [31], etc.

Unfortunately, although Internet of Things (IoTs) [5] make wireless sensor networks [38] increasingly pervasive and important (such as volcanic investigation [47], and agriculture monitoring [46]), there is no real implementation of SDN in a WSN. Existing works [29, 11, 28, 12, 17] all design frameworks of SDN for WSNs and validate them in . Even worse, these simulational SDN works construct the central controller of SDN in their database station [38]. The controller transmits control instructions with sensor nodes by multi-hop communications [32]. This causes a great deal of energy consumption [49], 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 controllers due to two major reasons. First, sensor nodes have limited processing capability and cannot deal with complex computations [42, 21]. Second, the sensor devices are also lack of storage space to support complicated operations. With the help of SDN, the controllers can conduct complex computations and store large-scale data information for WSNs.

Adler [14], a recent testbed that combines WSNs and the unmanned aerial vehicle (UAV) [7, 36, 8], 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 greatly 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 Adler is a traditional sensor system instead of a SDN system, it provides a favorable proof that UAVs are perfect to serve as the SDN controller.

Another advantage of utilizing UAV compared to terrestrial mobile controllers [43] is that, the air-ground communication range is almost 10 times than that of ground-ground communication under ZigBee protocol [15], 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], the first UAV based SDN sensor system. In Malak, UAV plays the role of the central controllers to monitor the sensor system. We designed routing interfaces, multi-task interfaces and AI interfaces in Malak to support 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 synergistically. For example, users can execute the AI node selection application to choose the node set for the multi-task application.

We implement Malak, using 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 LEACH [22] and RPL [48]. Our evaluation shows that:

1) Malak has high performance. Malak achieves overall 1.2 times higher throughput than RPL via the routing application ;

2) Malak has intelligent features. Malak achieves absolute error of energy prediction within 3×10^{-10} and obtain an 83.9% increment of network lifetime compared with RPL;

3) Malak is resilient. As time goes by, Malak maintains over 78.2% overall package received ratio while others drop to nearly 50% sharply.

The main contribution of this paper is Malak, the first practical SDN wireless sensor network system. Malak takes the first significant step to realize global optimization in a distributed WSN environment. Other contributions include several effective strategies and algorithms in the five implemented applications of Malak. Contrast to all the existing works, the applications in Malak can run together and benefit each other.

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

[1]In Hebrew, Malak is the semantic word of angels, who fly in the sky and bring goodness to all things on earth.

2.1 Software Defined Wireless Sensor Networks

There are several existing SDN approaches for sensor system, namely flow sensor [29], SDWN [11], sensor openflow [28], TinySDN [12], SDN-WISE [17]. However, they are all implemented and evaluated only in simulators.

Flow Sensor. The previous idea presented in paper [29] addresses reliability in the sensor networks through exploiting the OpenFlow technology[30]. Coming up with the concept of flow-sensor instead of typical sensor [27], 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 [11]. 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 [28], 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 [12] 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 [17], and the authors implement the prototype of this idea. Compared to other SDN solutions for wireless network, this solution suc-

cessfully 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. However, SDN-WISE only validates its performance in simulators.

2.2 Applications for Wireless Sensor Networks

There are already a rich set of the applications for WSNs. However, we study the typical applications and find that distributed paradigms for WSN cannot promise global optimization. Worse, no existing work can combine different applications to run together. To design a ecosystem for software defined sensor networks, we implement five applications including 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 [44]. Hence, traditional protocols cannot be used there. However, RPL is one of the standards for IPv6[13] 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 [45] 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 [39, 23]. Directional Diagnosis [19] 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 [35]. 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 [33]. 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 [25]. 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[40].

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[26] 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 [4] 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 [18]. 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 [50].

3. ARCHITECTURE

3.1 SDN Preliminaries

Before introducing our SDN system architecture, we first present some general terms including three planes and two interfaces in SDN system [24].

A data plane is the part of a network that carries user traffic and basic sensing functions. And a control plane is the part of a network that carries signaling traffic and is responsible for routing. The top layer of SDN system is management plane which includes the software services, such as simple network management protocol. It used to remotely monitor and configure the control functionality. The user implement their applications on it.

Southbound interface (SI) is 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) is the middleware on control plane can offer an API to application developers.

3.2 Overview of Malak

The architecture of Malak is shown in Fig. 1. The system is divided into three layers and two interface abstraction, the southbound interface connect data plane and control plane, and the nouthbound interface connect control plane and management plane. In our system, the data plane runs on sensor nodes. The control plane and management plane run on UAVs.

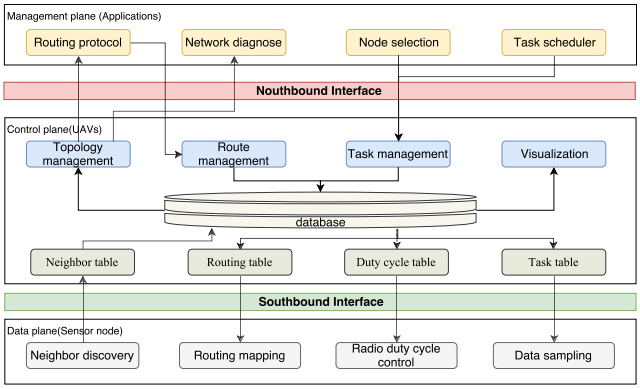


Figure 1: Malak architecture. Arrows are data flows

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.

3.3 SDN Planes of Malak

The **data plane** runs on sensor nodes contains four components, including neighbor discovery, routing mapping, radio duty cycle control and data sampling. We abstract the function running on sensor nodes to southbound interface. Our clearly abstraction of data plane functions make the SDN system work efficiently.

In the **control plane** of Malak, 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. Our system is capable to run application together. We design and implement five applications on Malak. The detail design of our applications is described in section 4.

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. The detail implementation of our system is introduced in section 5.

3.4 The Workflow of Malak

Given a deployment area, we show how the routing app works as example with six steps. First, each sensor node invokes the neighbor discovery component, computes a list of its local neighbors, and uploads the list to a UAV's neighbor table component in its own communication range. Second, a UAV stores the neighbor list sent from its nearby nodes into its database. Third, a set of UAVs fly and evenly distributed within the area, in order to cover all sensor nodes. In our evaluation, because each UAV can cover 150 m, and each sensor node can cover 20 m, we find it easy for UAVs to cover all sensors. Fourth, UAVs' topology management components compute a global routing topology, and upload the topology to the routing protocol module on the management plane. Fifth, the routing protocol computes a globally optimized routing topology (Fig 6), and distributes the topology to the route management component in UAVs. Sixth, each UAV generates a global routing table for itself and distributes the routing mapping table for each sensor nearby.

3.5 Nouthbound Data Flow

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

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

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 flow

the network.

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 flow

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 explaination of data structure will discuss in section 4.

3.6 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.

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

4. SDN 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 UAVs based SDN controllers. The central controllers help 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. Though SDN in wired networks is widely studied, there are only a few works on wireless sensor networks due to the following challenges. First, the wireless communication is unstable. The poor reliability of connections between controllers and sensor nodes makes the performance of SDN system worse, especially when the controllers communicate with nodes through multi-hop wireless approaches. Second, the sensor network is energy sensitive. The transmission energy cost between controls and nodes needs to be as less as possible. Last but not least, achieving the robustness of SDN routing in WSN is challenging because the controllers 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 mobile controllers which communicate with nodes via one-hop transmission. In this way, it improves the reliability of connection between nodes and controllers. 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, one kind of routing protocol is based on clusters. In this way, nodes in the network will choose their nearby header periodically. The nodes around the cluster header (CH) send their data to the header, and each cluster header forwards it to one another until the root node receives the data.

This kind of cluster method releases the energy hole problem [51] of WSNs. However, in the traditional WSNs, sensor nodes only have local information and are limited to computing capability which makes them hard to achieve global optimal routing.

We improve this cluster method by using our SDN controllers to update the routing protocol between cluster headers. At first, a UAV flies around the sensor field and get topology data from nodes. Then it calculates to decide the node to be the cluster header at a 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 energy limitation, UAV cannot keep flying above the sensing field. It only flies to sensing field to gather topology when the network is deployed and updates nodes' routing table 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.

As shown in Figure 4, after a node gets routing table from a 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 broadcasts its next hop IP and the substitute node IP. After the substitute node received the broadcast packet, it saves the source of next hop IP and keeps 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 algo-

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

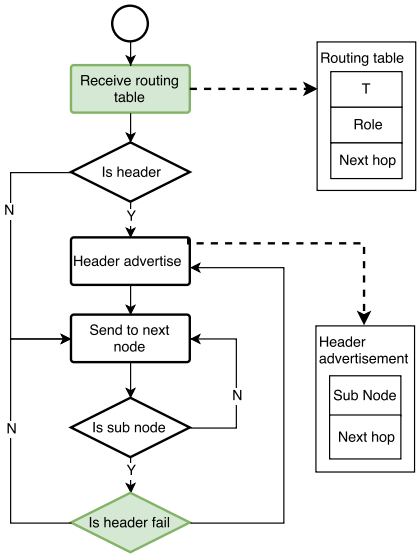


Figure 4: Routing local reparation

rithm

```
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 mechanism in Malak, and demonstrate the code in List 1. Our routing application in Malak achieves 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 [19], 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 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

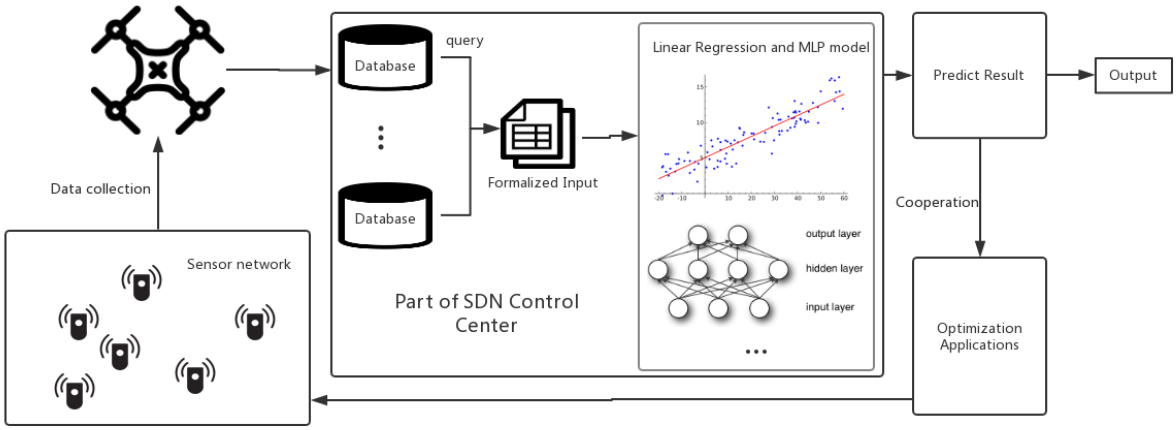


Figure 5: AI prediction

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.

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) [20], 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

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

Spatially regularized streaming sensor selection (SRSSS). To realize more intelligent and effective sensor selection, we introduce a state-of-the-art AI algorithm named spatially regularized streaming sensor selection (SRSSS) [26].

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:

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

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 [26]. 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

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 $Energy(u_i)+\phi \geq \eta \| Storage(u_i)+\varphi \geq \xi$ **then**
    Set T to task queue;
  end if
end for
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 Mutitasks

```
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();
}
```

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.

Table 2: Task Buffer

Task ID	Node set	Sensing rate	Sensing range	Sensing duration
001	$\{u_2, u_6, ..., u_{99}\}$	0.2	4	2046
...	...	...	...	...
101	$\{u_3, u_4, ..., u_{67}\}$	0.2	5	4096
...	...	...	...	...

5. IMPLEMENTATION

5.1 Data Plane

We modify contiki-ng, a system for next generation IoT (Internet of Things), to abstract sensor nodes functions. 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 frist 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 start data sampling to make sure that they cost minimal energy to get the same quality data.

5.2 Control Plane

We run our SDN control plane on 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 kid of sensors which is suitable for large scale wireless sensor networks.

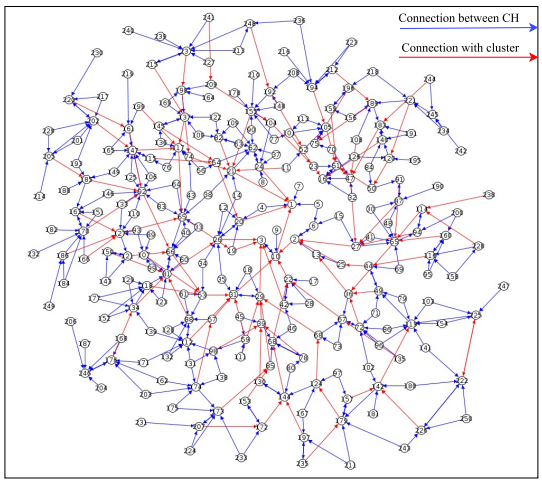


Figure 6: Network Run-time Topology Computed by Malak. The blue lines represent the connection between cluster members with cluster headers, and the red lines donate the route between cluster headers deployed by UAV.

According to our UAV gathered network run-time topology, we draw Figure 6, the network run-time topology computed by Malak.

6. EVALUATION

6.1 Evaluation Setup

In this section, we evaluate the performance of Malak by real-scene experiments. We deploy 100 TI SensorTag sensors randomly in a 250 $\times$ 250 square meters area. The operating system of sensors is contiki-ng []. We use different metrics for various applications and compare our works with other state-of-the-art works such as RPL and LEACH. Although Alder implements a sensor system using UAVs, it doesn't provide a complex routing protocol, and thus we do not compare with it on throughput or latency.

When measuring 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 8000 seconds.

6.2 Evaluation Metrics

Packet received ratio is defined as the proportion of the total data packets received by data center and the total data packets sent by all nodes, we adopt the model in [9] and it can be formulated as

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

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 use our basic Malak routing protocol without any intelligent approaches only. We evaluate routing performance by throughput, packet received ratio and end-to-end latency. Meanwhile, We also evaluate the scalability of our network through network lifetime. We do not compare Malak with Alder on throughput and latency because Alder doesn't implement a complete routing protocol.

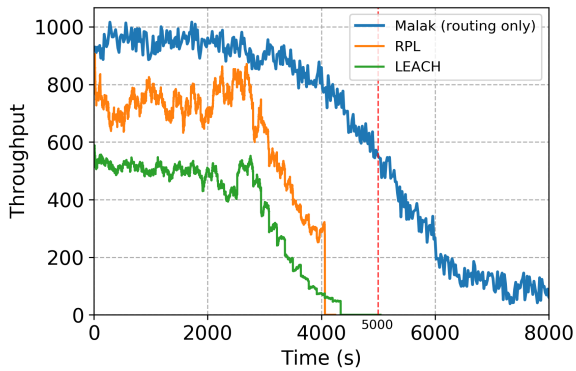


Figure 7: End-to-end routing throughput. The Malak's integral of the throughput is 1.2 times higher than the the PRL's in within 0 to 5000 seconds, and is 2 times higher within 0 to 8000 seconds.

Figure 7 compares network throughput by deploying various routing algorithms. As our Malak uses OSPF, which finds the optimal path from sensors to data cen-

ter, to calculate the route table, its throughput exceeds RPL and LEACH. This implies there are less collisions in Malak. Besides, in Malak, sensors' lifetime increases a lot more than others, since all computation-intensive tasks are done by UAV and sensors do not need to send packets to negotiate route path, which decreases the energy consumption with no doubt.

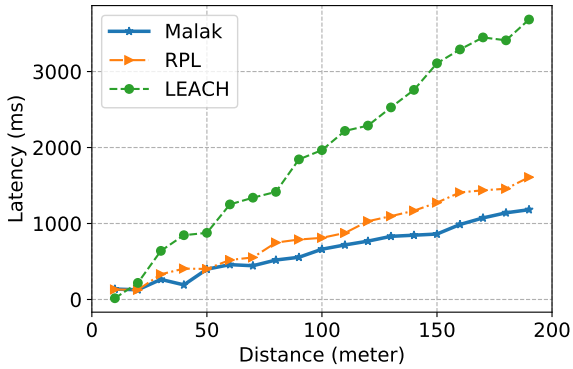


Figure 8: End-to-end average latency of different routing algorithms and distance.

When measuring the latency of the network, our Malak routing method performs best due to the use of the global topology information to find the optimal route path. As shown in Figure 8, 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.

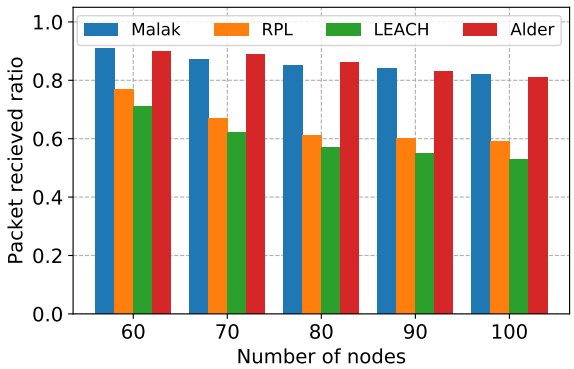


Figure 9: Per-node packet received ratio improved by Malak. This figure implies the Malak diminishes collisions of nodes much more by reducing the routing negotiation, which decreases the radio transmission.

Figure 9 shows two remarkable advantages of Malak: (1) The packet received ratio exceeds state-of-the-art algorithms RPL[48] and LEACH[22] 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.

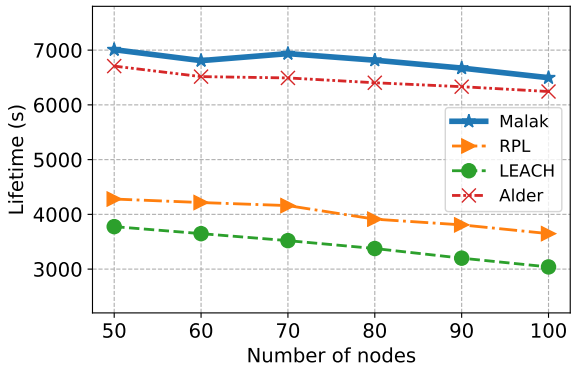


Figure 10: Network lifetime using various routing algorithm Malak surpasses RPL and LEACH in different network sizes.

Our Malak system is capable in large scale wireless sensor networks as Figure 10 shows. Its lifetime excels RPL and LEACH in both small and large network sizes.

6.4 AI Applications

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, and it can further improves the reliability of network. And the packet received ratio grows by 10%∼15% and achieves almost 100% as Figure 11 shows.

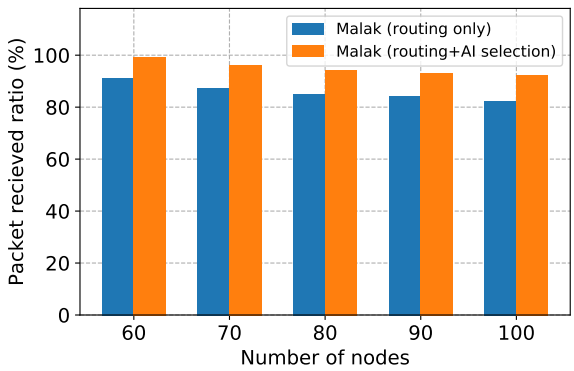
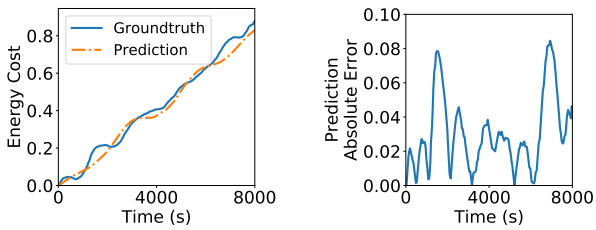


Figure 11: Packet received ratio improved by intelligent node selection

By deploying MLP on UAVs, Malak can predicts the energy consumption precisely as Figure 12 shows, and the absolute error can be restricted in 0.1, which enable us to predict and fix energy failure in advance.



(a) Energy prediction (b) Prediction error

Figure 12: Energy prediction We use MLP to predict energy consumption and the prediction result is consistent with the truth-ground. The absolute error of energy prediction never exceeds 0.1 with normalized energy (i.e. the maximum energy of a sensor is 1).

6.5 Resilience

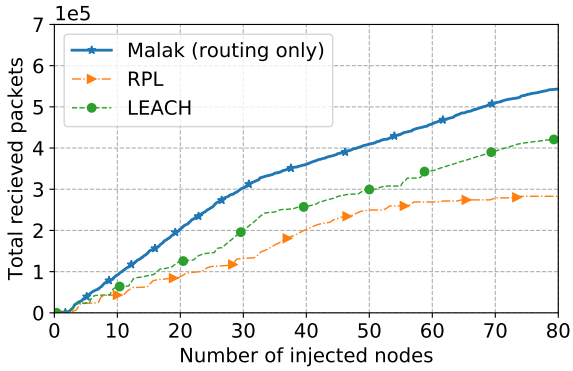


Figure 13: Fault tolerance RPL and Malak are compared on resilience, and Malak perform slightly better than RPL.

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 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.

We compare our algorithm with DID[19], but as it is not open-source, we implement DID by closely following the algorithm proposed in [19] and evaluate it in our settings. The main reason Malak performs better than DID is that our UAVs can fly to a the fault area which decrease the number of potential candidate of injected nodes. Besides, our 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 15.

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%.

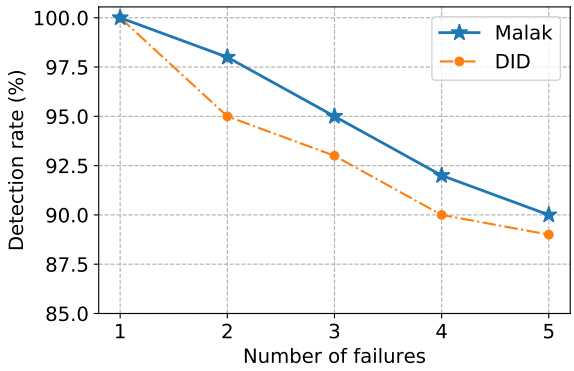


Figure 14: Comparation of Malak and DID on failure detection rate. Malak surpasses DID by using UAVs to lessen dectecting area.

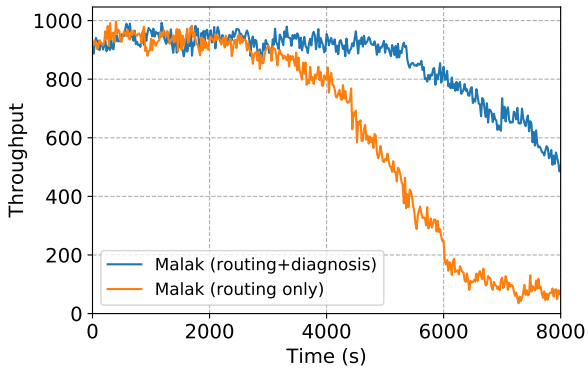


Figure 15: The performance improvement of automatic network repair. The automatic repair mechenism increases the lifetime 1.5 times compared with the basic Malak routing algorithm.

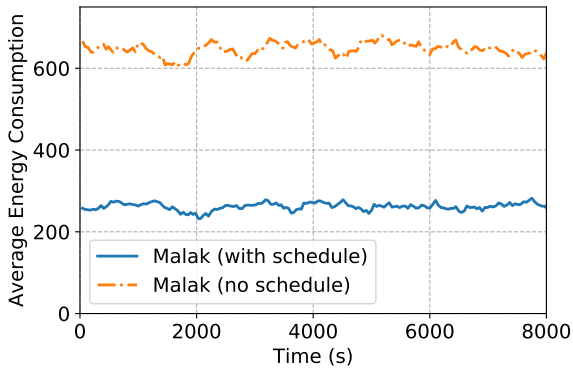


Figure 16: Average energy consumption with and without multi-task schedule When with multi-task schedule, sensor consumes half of the energy comparing to when without multi-task schedule.

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 improves the throughput by 1.2 times and prolongs the average lifetime of sensor nodes by 81.2%. By running the energy prediction application, Malak further increases the lifetime by 83.9%. All source code and results of Malak are ready to open source.

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), 2237âĂŞ2249.

[3] Alicherry, M., Bhatia, R., and Li, L. E. Joint channel assignment and routing for throughput optimization in multiradio wireless mesh networks. In *International Conference on Mobile Computing and NETWORKING* (2005), pp. 58–72.

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

[5] Atzori, L., Iera, A., and Morabito, G. The internet of things: A survey. *Computer Networks 54*, 15 (2010), 2787–2805.

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

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

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

[9] 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).

[10] 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.

[11] 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.

[12] 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.

[13] Deering, S., and Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification*. RFC Editor, 1998.

[14] Dongda, L., Yuexuan, W., Zhanoquan, G., Tong, S., Tianhao, W., Yongqin, F., Heming, C., Mingli, S., and Francis, C. Adler: A resilient, high-performance and energy-efficient uav-enabled sensor system.

[15] Farahani, S. *ZigBee Wireless Networks and Transceivers*. Newnes, 2008.

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

[17] 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.

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

[19] 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.

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

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

[22] Kaur, S. P. WsnâĂŹs leach protocol: A survey. *Wireless Communication 8*, 3 (2016), 113–116.

[23] 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.

[24] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. Software-defined networking: A

comprehensive survey. *Proceedings of the IEEE 103*, 1 (2015), 14–76.

[25] 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.

[26] 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).

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

[28] 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.

[29] 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.

[30] 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.

[31] Miao, G., Himayat, N., Li, Y., and Swami, A. Cross-layer optimization for energy-efficient wireless communications: a survey. *Wireless Communications & Mobile Computing 9*, 4 (2010), 529–542.

[32] 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.

[33] 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.

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

[35] 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.

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

[37] Poole, D. L., and Mackworth, A. K. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, 2010.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

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

[43] Somasundara, A. A., Ramamoorthy, A., and Srivastava, M. B. Mobile element scheduling with dynamic deadlines. *IEEE Transactions on Mobile Computing 6*, 4 (2007), 395–410.

[44] 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.

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

[46] 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.

[47] 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.

[48] Winter, T. Rpl: Ipv6 routing protocol for low-power and lossy networks.

[49] 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.

[50] 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.

[51] Zeng, Z. W., Chen, Z. G., and Liu, A. F. Energy-hole avoidance for wsn based on adjust transmission power: Energy-hole avoidance for wsn based on adjust transmission power. *Chinese Journal of Computers 33*, 33 (2010), 12–22.

[52] 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.