Fire Away

Authors: Karen Abruzzo, Ankita Bhanjois, Arden Diakhate-Palme
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A system capable of signalling to emergency services the location of a wildfire. Currently, it takes time and effort for wildland firefighters to constantly monitor the possible wildfire locations. This project intends to create a low power and accurate system that can detect the location of wildfires, by using a wireless sensor network. If a fire was set at a specific node in the network, that node's data will be transmitted across the network in order for wildland firefighters to then be deployed.**

***Index Terms*—Fire Detection, MAC Protocol, Node, Routing, Scheduling, Spanning Tree Protocol, Web Application, Wireless Sensor Network**

1 INTRODUCTION

In 2022, California has had approximately 6000 recorded wildfires, spanning almost 300 thousand acres. A fire itself takes around 1 hour from the ignition of the fire to become a fully developed fire. This was the inspiration behind our project. We wanted to create a system for wildfire detection to be used by wildland firefighters. The current solution for detecting forest fires is if someone calls 911 or via a system of lookout towers. However, with our solution, firefighters would be able to pinpoint the location of conflagrations in a timely manner. The system is designed for their convenience and safety. Therefore, the goal of the system that we are creating for our project is to have (1) accuracy and (2) low power.

In order to do this, we will be creating a wireless sensor network of eight nodes, in which each node will collect data regarding the temperature in its vicinity to indicate if a fire was ignited. The sensor data collected will be transmitted across the network and be collected by a gateway router, which will display the node where the conflagration was detected on a web application. The web application will provide an alert to the fire department to deploy their personnel.

Due to budgeting and time restrictions that come with completing this project in one semester, the project is of course scaled down. However, the network protocol that is being used is designed to be scaled up. The node architecture is scalable because there is a variable window for sensor data collection. However, a mere temperature sensor mounted on a tree would not be able to detect forest fire, since it is only detecting the ignition of a matchstick. At scale, the web application would be hosted in the cloud to circumvent connectivity issues (i.e. in case the network was to go down). The main part of our project is how to use a wireless sensor network for fire detection and the protocols that make the system useful.

2 USE-CASE REQUIREMENTS

Our use case is rapid wildfire detection for the fire department. Since it could be catastrophic if a fire is not detected, we want to detect a fire with 90-95% accuracy. This is to allow for some slack in case a node goes offline. Our spanning tree protocol would need to adjust itself to find an appropriate path to the border gateway. It takes 1 hour from the ignition of a fire to become an open fire. Hence, it is critical that the fire is detected during this period, so that wildland firefighters can put out the fire before it becomes fully-developed. Therefore, our web application needs to show active fires within 15 minutes of conflagration. The nodes also need to be low power to make our system as low-maintenance as possible. The advantage of using a sensor network is that people would not have to be stationed in the forest in lookout tours to keep an eye out for fires. If park rangers need to frequently visit the nodes to replace the battery, then it defeats the purpose of the system. In order to meet the needs of the use case, the nodes need to be able to operate for one month on a 3000mAh battery.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

A block diagram visualizing our system is provided in Fig. 1, but for clarity we have also provided a full-page block diagram visualizing our system in Fig. 11 on page 11. This block diagram has been broken down into individual components in the upcoming sections. Our system is broken down into three major subsystems: Networking, Node Architecture, and Web Application. The first subsystem is the Network that we are constructing. We are using a MAC protocol to reduce power consumption by using a spanning tree protocol, a link-state advertisement, and RX/TX timeslot scheduling. This would help transmit data between the nodes about the sensor data being collected and adjust itself in the case of collisions or node failure. The second subsystem is the Node Architecture which consist of our micro-controller, transceiver, and sensors for fire detection. This system needs to maintained low power. The third subsystem is the web application. This application will display the nodes where the fire is located and alert the firefighters of that destination. The application and router for subsystem 1 will be hosted on a

Raspberry Pi.

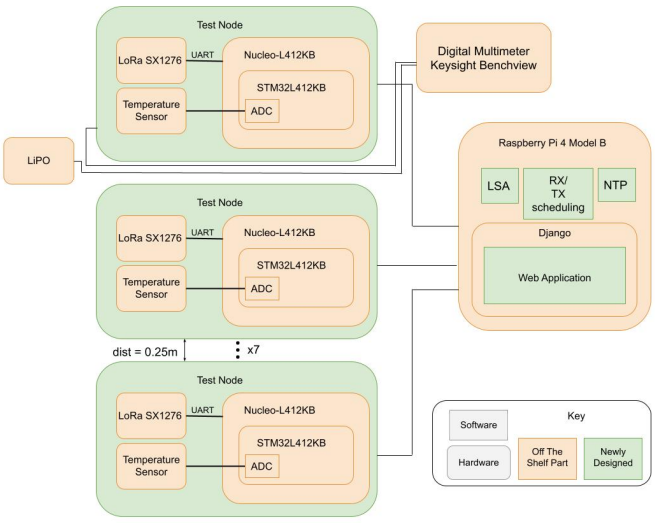
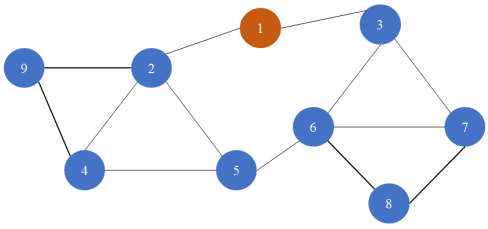
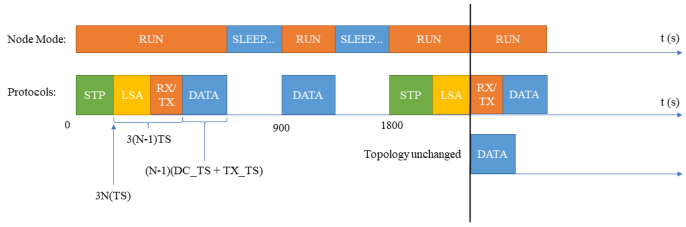


Figure 1: Overall Block Diagram

3.1 Networking



(a)



(b)

Figure 2: (a) Example Mesh Topology (b) Protocol Stack and Operation Overview

Our main objective for this section is to maximize the nodes' sleep time and minimize the amount of time they spend transmitting and receiving data. This is an example of the initial mesh topology. A network operator would hike across the forest and fasten nodes on trees above RF-obstructing foliage, ensuring that nodes adhere to the mesh topology rules (described in system implementation). Figure 2(a) provides an example of the resulting topology after the network administrator has setup the nodes, covering a region in the forest.

3.2 Node

This subsystem is how the nodes are constructed as we want to create a system that is low power. Our STM32l4 micro-controller allows for low-power. The LoRA transceiver also has low power and has a range that can be used to scale up to a real world scenario. A temperature sensor is used for detecting fire. The node is powered by a LiPO battery.

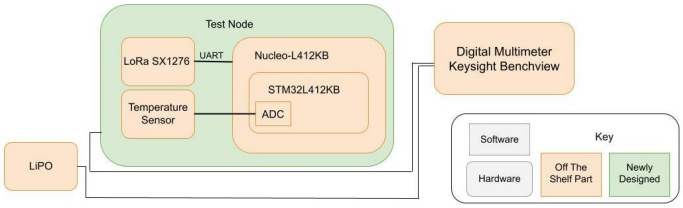


Figure 3: Node Architecture

The image shown above is the test node. This node will be used to show the power consumption. All other nodes used in this system do not have a battery as shown in Fig. 11.

3.3 Web Application

This subsystem is how the web application is going to be configured. The web application will be written on Django and will consist of two main components: the visual interface and the push notification system. The interface will take in the inputs from the sensor and display where the fires are located. The push notification system will also take the sensor data and alert the firefighters of the node with the fire.

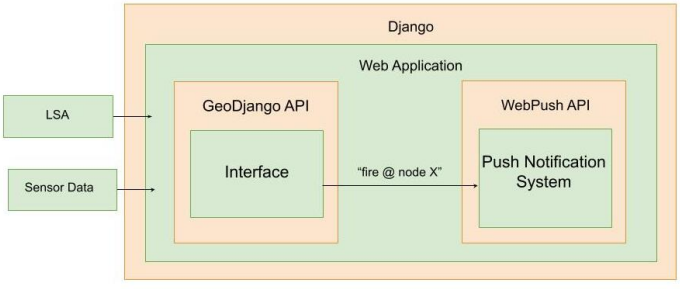


Figure 4: Web Application Block Diagram

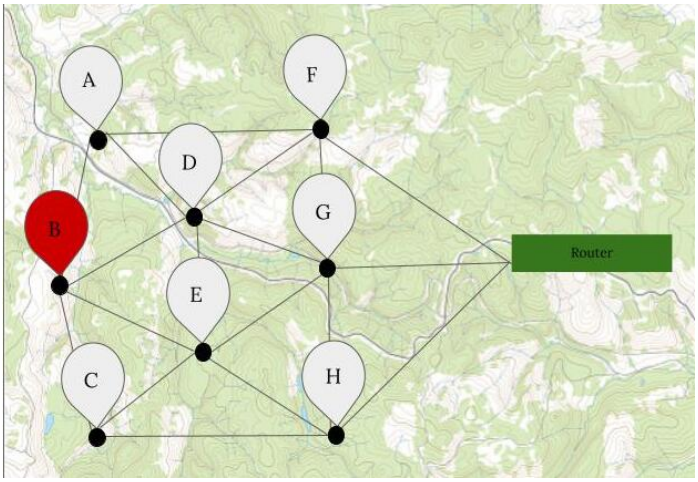


Figure 5: Web Application Sample Interface

Fig. 5 shows what the interface would look like for the web application. Note that the red node B refers to the node where the fire is located.

4 DESIGN REQUIREMENTS

We would like our network to be up for a minimum of 1 month, in order to make maintenance cost reasonable. Hence, each node should be able to last one month on its battery life. Furthermore, if a node drops offline due to an external event (such as a branch hitting its antenna), we would want the network operator to be informed of this. If a node drops offline it could also mean that a fire has engulfed the node within the 15 minute window (maybe the node was at the epicenter). So, we would want to inform the fire department that at worst, a conflagration has occurred at this node's location.

In order to minimize the cost of a single node, we should minimize battery cost. Transceiver power consumption, and the MCU will be the primary drain on the battery life. Since propagation delay will differ based on the specific configuration of the LoRa transceivers, we are considering the default settings for power and total runtime of the networking protocol stack (see Fig. 6).

Ideally, we would want 15 minutes updates without node failure detection, and a 30 minute timespan to detect node or root-link failure. We have decided to reduce the mesh network to a tree topology, and will detect node failure at any point in the protocol once the tree has been formed.

Furthermore, we want the MCU to be in the lowest power mode for the longest period of time. When it is not transmitting or performing control-plane functions. We are able to achieve such low power modes, by placing the node's transceiver in low-power mode, and then transitioning the MCU to a low-power mode.

The power consumption of a given node is contingent on the efficiency of the routing protocol. If the protocol does not allow nodes to go into low-power since the node has to be able to route packets, we will be losing power unnecessarily.

Our nodes will be placed 0.25 meters apart from each other. This will ensure easier testing as we can keep the node on a singular table. This distance was determined knowing that the LoRA transceiver can have a range of 250m and we wanted to scale that down. [2]

5 DESIGN TRADE STUDIES

5.1 Network Architecture

In order to enable nodes to sleep for a maximum period of time, we considered using the LoRaWAN MAC protocol. After having read through the Link Layer specification, class A operation of network nodes seemed to be meet our use-case requirements best. According to the specification document, "Class A operation is the lowest-power end-device system for applications that require only downlink communication from the server shortly after the end-device has sent an uplink transmission." [5] The two principal downsides to employing the LoRa-WAN protocol are as follows[5]:

1. **Star-of-stars topology**: The need for intermediary gateways which communicate with the Network server (border gateway) over IP connections

2. **Single-hop** RF communication from nodes to gateways, not applicable for multi-hop sensor networks, gateways would need to have a longer battery life

3. **ALOHA MAC protocol**: Less precise control over node sleep schedule, and higher collision rate: throughput $S = \frac{1}{e}$ for pure ALOHA and $S = \frac{1}{2e}$ for slotted ALOHA.

A multi-hop wireless sensor network would reduce material cost since higher-capacity expensive batteries would be required for intermediate LoRa-WAN gateways. Furthermore, power grows exponentially in a multi-hop WSN, so we will require technicians to service nodes closer to the border gateway more frequently than nodes deeper in the forest. However, nodes closer to the gateway are assumed to be more accessible.
When designing our own protocol for low-power medium access control, we considered two main strategies.

1. **Asynchronous B-MAC**: Nodes periodically wake up to check their children nodes In this protocol, a node that wants to send a packet sends a preamble for the same time duration as the receiving node's sleep period. Hence, nodes must periodically return to a "receive-only" state to check if their children have a packet to send. For a multi-hop network this approach imposes a prohibitive current draw for all nodes, since the sampling period is once every 15 minutes.

2. **Synchronous MAC**: Nodes use TDMA to avoid packet collisions, which allows them to go to a lower power-consumption sleep state when they are not scheduled to transmit or receive. The tradeoff is that this type of MAC protocol comes at a high traffic control cost, and requires periodic beacons to correct nodes' clock drift. A global notion of time is required so that nodes wake up and transmit in their actual timeslots, and not an adjacent one. Since each node each node's MCU will have to use its own clock to determine when it is time for them to transmit, periodic clock synchronization is necessary to ensure a global notion of time across nodes.

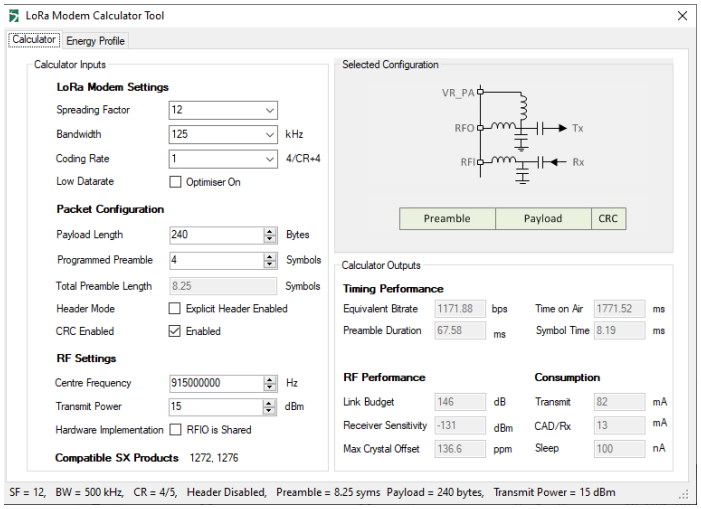


Figure 6: Modem Attributes

We have calculated the worst-case power consumption and times as follows based on the chosen synchronous MAC routing algorithm. Here, t_{TS} is the duration of a TS in milliseconds, and t_{DC} is the duration of a data-collection timeslot (DC_TS) in milliseconds:

1. **STP**: The protocol will convergence in 3 iterations, and in the worst-case for a single iteration, each sensor node will have to transmit and receive at most $N-1$ times. If we also consider timeslots for the gateway to send STP messages we obtain the following. Furthermore, a node will transmit at most $N-1$ in a star-topology.
   total-duration: $3N(t_{TS})$
   power consumption: $3(Q_{TX}+Q_{RX})(N-1)$

2. **LSA**: A spanning tree of N nodes will have at most $N-1$ links (if the tree is a line-topology). Since we are transmitting both LS_REQ and LS_RESP, we obtain the following maximum time for this phase. Furthermore, a node will transmit a packet and receive a packet in the worst-case $N-1$ times, if it is the center of a star topology.
   total-duration: $2(N-1)t_{TS}$
   power consumption: $(N-1)(Q_{TX}+Q_{RX})$

3. **RX/TX**: This is the same worst-case time duration as the **LSA** phase, except that we are not waiting from ACK's from children nodes. Furthermore, a star topology would maximize a node's transmissions, but ACKs are not required for leaf-nodes, so each node would only have to receive a schedule once.
   total-duration: $(N-1)t_{TS}$
   power consumption: $(N-1)(Q_{TX})+Q_{RX}$

4. **DATA**: As observed in Fig #, for a WSN of N nodes, we have that each node will necessarily need to take sensor readings and transmit them upstream to the GW router. A minimal total duration was obtained via the RX/TX scheduling algorithm. Furthermore, in a star topology, a node would have to receive $N-1$ times, but each node is only transmitting once in this phase.
   total-duration: $(t_{DC}+t_{TS})(N-1)$
   power consumption: $(N-1)(Q_{RX})+Q_{TX}$

Considering the ToA based on the LoRa modem parameters, a timeslot buffer of 0.5 seconds, and $t_{TX_TS} = t_{RX_TS} = t_{TS}$ =ToA+0.5s = 2.271s
t_{ToA} = The ToA for modem RX and TX functions [6].
$t_{DC} = 0.5t_{TS}$
where I_{RUN} = 3.79mA, I_{SLEEP} = 0.95uA [[9]]
I_{RX} = 13mA, Q_{TX} = 82mA, I_{lora_low} = 100nA (see fig.) [6]
I_{TEMP} = 50uA. [7]
The average power consumption for this protocol is as follows:

$$I_{RUN}\frac{(t_{TS}(9N-6))}{1800}+I_{SLEEP}\frac{(1800-t_{TS}(9N-6))}{1800}$$
$$+I_{RX}\frac{(6N-5)(t_{ToA})}{1800}+I_{TX}\frac{(5N-3)t_{ToA}}{1800}+$$
$$I_{lora\_low}\frac{(1800-t_{TS}(9N-6))}{1800}+I_{TEMP}$$

Placing in the corresponding values we obtain that the protocol run at a 15min sampling period and 30min link/node failure detection period consumes on average 3.86mA. Running this WSN for a period of 1 month would consume $(3.86)(720) = 2779.0mAh$. Hence, a battery with $3000mAh$ of capacity would make a node last one month.

5.2 Low Power Networking VS BMS

One of our ideas for making the system low power was to have a battery management system and a solar panel. The idea was we could power the node and charge the battery when it was sunny, and when it was not sunny we could use the battery to power the node. If the node got low on power, we could change how often we transmit data or put the node into a really low power mode to preserve power until the battery could be recharged. However, we decided not to go with this idea for a few reasons. None of us are familiar with a BMS, and due to the time constraints of the class we were worried about figuring out how to get it to work in time. There was also concern that since the sensors are meant to be placed on trees in a forest, the trees

would block too much of the sun and that they would not be able to recharge the battery quickly enough to avoid the battery completely running out of power.

5.3 Node Architecture

For the sensor nodes, we decided to use the STM32L4 as our micro-controller as it is one of ST's ultra-low-power microcontrollers. It has the lowest current in its lowest power mode with RAM retention and the RTC enabled compared to the other ultra-low-power STM32s.[8] It also has a larger variety of low power modes than most of the other ultra-low-power STMs, which allows for more flexibility when entering a low power state.

In addition to the STM32L4, we decided to use LoRa trancievers because they are relatively low power for the range they offer. LoRa's Tx power of 20 mW makes it lower power than Wifi and 3G/4G, which have Tx powers of 80 mW and 5000 mW respectively. Bluetooth is lower power at 2.5 mW, however it only has a range of 10m. This would not meet the requirements of our use case, where nodes would need to be spread out over a forest. LoRa, however, has a range of 250m in a forest environment. This would indeed fulfill our use case requirements and help when this node architecture is scaled up in a real world scenario.

5.4 Web Application

Regarding the web application, we wanted to create a way to visualize the location of the fires. We are using a Raspberry Pi to host the web application. We chose to use an RPi for convenience as the RPi will be used as both the border gateway for the network and as the host for the web application. For the webapp itself, we will be using Django as our framework software. Django uses Python which is easier to use in terms of implementation. Django is also well-documented and there are plenty of tutorials that can be of assistance. Django also has two particular libraries, GeoDjango and Webpush, that are beneficial for the design of the web application, making it a favorable option. It is important to note why we are using the RPi to host the web server instead on the cloud. The web application that is being created is simply being used as a visualization of the nodes and the location of the fire. It also includes a push notification system. The web application itself is not to scale if this project would be scaled up. A web application would need to be reconfigured and redesigned for it to be used on a larger scale. For the purposes of our project, it is simply a visualization tool.

6 SYSTEM IMPLEMENTATION

6.1 Networking

Our MAC protocol seeks to reduce power consumption, and to be resilient in the face of link or node failures. If a node drops offline or the single link connecting a node to the rest of the WSN fails, a field technician should be informed of that node's failure. Since traffic intensity increases exponentially as we approach the tree's root, we will be using data aggregation to take statistics of sensor data over a period of time, and then send that data up the tree during that node's transmission timeslot.

1. **TDM STP**: On node startup, it will run the spanning tree protocol until it has heard a message from the true root (which is the gateway router with address #1). This will take 3 iterations given the following constraints on the initial mesh topology:
In order to ensure that no packet collisions occur, a node will wait for $a(TS)$ before transmitting, where for a topology of N nodes, $a \in [0, N-1]$ is the node's address.

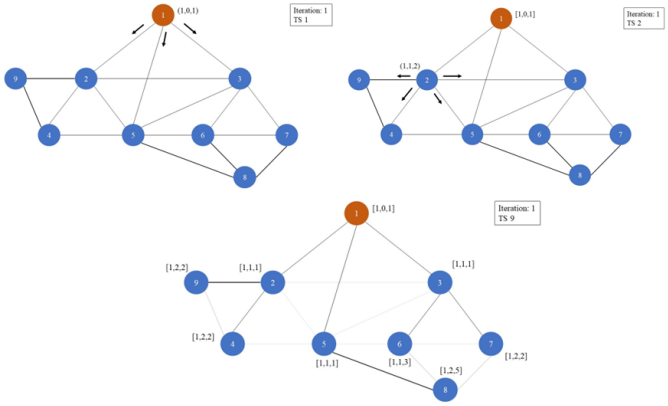
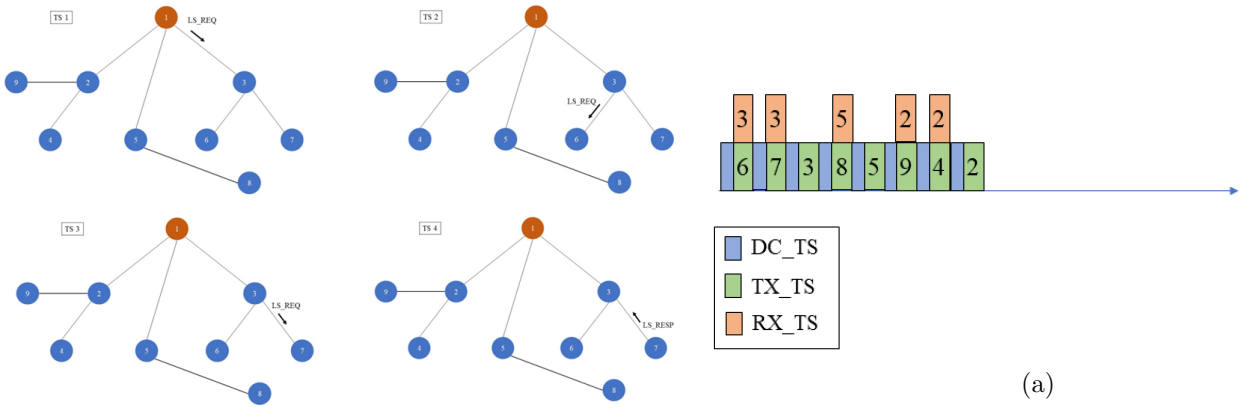


Figure 7: Spanning Tree Protocol with time-division

2. **LSA**: Once the spanning tree protocol has converged (in three maximum iterations given the constraints for the initial topology). Propagate link-state requests down the spanning tree. Each node gathers link-state responses from their children, and then passes that information back up to their parent.

   In this phase, each node begins in RUN mode with active listening.

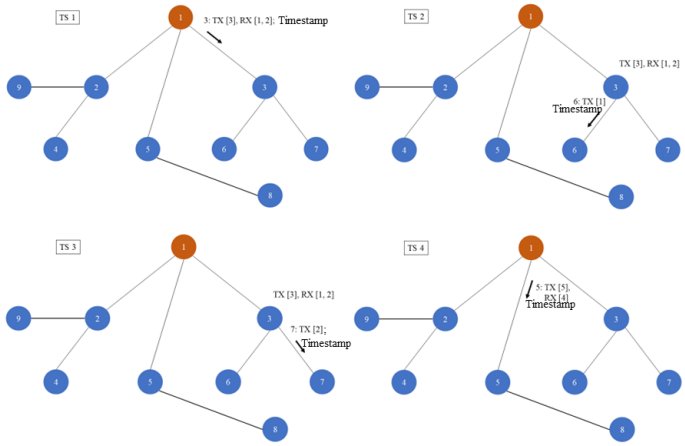
   - If the node has children,
     - on receiving an LS request, forward that request to each child one by one starting with the child with the lowest address.
     - once it receives LS responses from each of its children, store these responses locally (it now knows the topology of all its children), and aggregate the responses and send them to your singular parent node.
   - If the node has **no** children,
     - on receiving an LS request, respond with your neighbors (i.e your ID) to your parent.



(a)

Figure 8: Link State Protocol Overview

3. **RX/TX Scheduling**: Once the LSA section of the algorithm has run, the gateway router knows the overall topology of the sensor network, hence, it can schedule transmission and receive slots for each node to ensure a power-efficient TDM schedule which allows all nodes to propagate data up the spanning tree back to the root. Having a global view of the tree, the gateway router schedules TX and RX data slots as follows:



(b)

Figure 9: (a) Example schedule created by GW router (b) Overview of RX/TX Schedule Distribution Algorithm

- nodes which have children have RX slots when their children are transmitting, and TX slots once all their children have transmitted.

- nodes which have no children only have TX slots.

It is similarly crucial that the transmission schedule accounts for a sampling time DC_TS during which the node can poll its temperature sensor.

This schedule is then encoded in a packet, and eventually sent to each child. This requires all nodes to be in the RUN mode with active listening. Schedule distribution proceeds as follows:

- on receiving RX/TX timeslots,
  - Store RX/TX timeslots received in memory.
  - set data-sampling interval DC_TS to the received value
  - If the node has children,
    - allot TX_TS's for every RX_TS received, such that the node's children transmit when the node is listening. Assign RX/TX slots to children according to the downstream topology (learned in the LSA phase).
    - forward assigned TX and RX timeslots to children.
  - If the node has **no** children, the node can enter SLEEP_MODE, after setting its DATA_TIMER.

- on receiving timestamp from parent, calculate an offset based on the packet's total_hop_ct, the timestamp, and the current time. Use this offset to set the SCS_TIMER.

Timestamps from the border gateway are also piggybacked onto schedule packets, such that a nodes can adjust for their clock-drift [[4]]. A node calculates its offset via the following formula:

$$T_i(L) = T_n(L) - \delta$$

Here, $T_i(L)$ is an node n's estimate of when its parent (node i) sent the timestamp. δ is a constant based on the LoRa PHY layer propagation time. $T_n(L)$ is node n's local time when it received the timestamp.

$$T_s(L) = T_i(L) - (h)t_{TS}$$

Here, $T_s(L)$ is an estimate of when the GW router sent the timestamp. h is the hop count, and t_{TS} is the duration of the one timeslot.

$$Offset = T_s(R) - T_s(L)$$

Here, $T_s(R)$ is the beacon timestamp value in the received packet.
Then the offset is used to adjust local time in the calculation of DATA_TIMER. The amount of time to transition between RUN and SLEEP mode is also included in this timer calculation. Hence, all nodes' timers will expire simultaneously, and all nodes will transition to the Data sampling phase. The node waits in RUN_MODE until DATA_TIMER expires.

Note that due to the chosen MCU's minuscule drift rate, we will not have to re-adjust for time every 30minute interval.

4. **Data Sampling** A node now knows its RX and TX schedule, how long it must spend on data collection, and the topology of its downstream child nodes.

- Once the node's DATA_TIMER expires, the node will transitioning out of SLEEP_MODE into RUN_MODE.
- If the node has children, the node will
  - Spend its RX_TS's waiting to receive data from each of its children
  - Aggregate of the received statistics (min, max, avg) of all its children. (it will include its own sensor data in this aggregate once it becomes available)
- The node will spend DC_TS seconds polling the temperature sensor. Taking statistics (min, max, avg) of the sensor readings and storing them in FLASH.
- Transmit its data statistics or aggregate of statistics (if it has children) to its parent node.

6.2 Node Architecture

We will be implementing a sensor network by creating a system of 8 wireless sensor nodes. The sensors will use an STM32L4 microcontroller, as it is designed to be low power and has multiple low power modes. A LoRa transceiver, which the STM32 will communicate with using UART, will be used to send data between the nodes and the gateway. The nodes will have a TMP36 analog temperature sensor in order to detect fire, which the STM32 will get data from using ADC.

The STM32 has multiple different low power modes. The ones we plan to use are low-power run mode, STOP2, and shutdown. When the node needs to be "awake" the STM32 will operate in low-power running mode. When the node does not need to be awake, but does need to remember the schedule for when to awaken for transmitting/receiving, the STM32 will go into STOP2 mode. This is the lowest power mode where all memory and registers are maintained. When the node does not need to be awake and no longer needs the schedule, the STM32 will go into shutdown mode, as this is the lowest power mode.

6.3 Web Application

To create our web application, we will be using Django as web framework. Upon receiving data from the sensors (i.e. the fire was sensed at node X), the web application would take this data as input. Using the GeoDjango library[3], the web application will consist of a topographical map of a forest, which would be overlayed with markers. The library lets us choose from different maps and for the purposes of our project, one of a forest will be chosen. The markers are the pins on the maps which represent the individual nodes in our network system. When the sensor data is received, the marker that is associated with the node where the fire is located will change color (or show some other visual indication of a change). This inputted sensor data would also be sent to the push notification system being created, using the Webpush library[10]. This would result in a pop-up window with a message similar to: "Fire located at node X." These web application updates and notifications should be instant after receiving the data from the sensors. Essentially, the web application is the visualization of what is happening in the "forest."

7 TEST & VALIDATION

7.1 Node Testing

To test if the nodes is low power, we will measure the current over a 30 minute period using a multimeter and Keysight Benchview. Using this data, we will calculate how much power the node used over the 30 minutes, and estimate how much power the node would use in a month. The node will pass the test if we estimate it would use less than 3000mAH of power in a one month period (720 hours).

We will be verifying that the power consumption for each node is less than the worst-case value for each phase of the network protocol. Q_{TX} and Q_{RX} from section 5.1 are based on the following statistics for the default calibration of the LoRa modems (see Fig. 6).

7.2 Tests for Network

We will test three distinct network topologies, as illustrated in Fig 10. We will gather message logs on a subset of nodes for all phases of each protocol and ensure that each protocol takes as long or shorter than the worst-case time. Actual values for the TS and DC_TS times are taken from the LoRa modem calculation tool's ToA value (assuming default transceiver configuration).

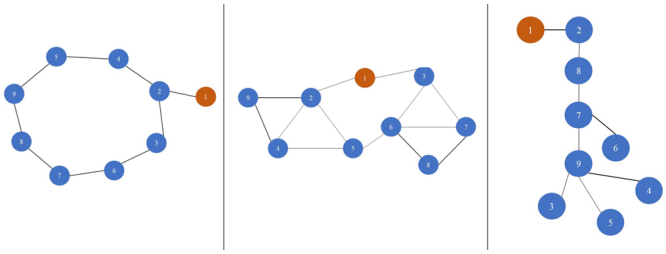


Figure 10: Link State Protocol Overview

7.3 Tests for Web Application

Testing the web application is mainly integrated with testing the network as it needs to show which node the fire is occurring. The web application should be updated once the border gateway receives the data regarding the location of the fire. Design testing can be done without the other systems of the project; however, to test if the data is being received the network must be completed and functioning. As the network is being built iteratively, once there is a working network with a couple nodes, the connection to the web application is tested to see if the data is displayed correctly. As more nodes are added, this connection is constantly checked to make sure the web application is reacting appropriately. Verify that the topology after the reduction to a tree (after STP and LSA phases) is reflected properly on the web app, (active links highlighted in UI).

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule is shown in Fig. 12 on page 12.

8.2 Team Member Responsibilities

Karen will be focusing on creating the nodes, putting the nodes into different power modes to optimize power usage, and getting the nodes to communicate with each other.

Ankita will be focusing on creating the web application. Once that is completed, she will assist Arden with the network. She will be helping manage the scheduling protocol that the network will use.

Arden will be focusing on developing each phase of the networking protocol, checking that they are inter-operable and verifying and validating the protocol in Networking tests.

8.3 Bill of Materials and Budget

Refer to Table 1 on page 10 for the bill of materials.

8.4 Risk Mitigation Plans

One of the risks our project faces is that the timestamps received by the nodes from the border gateway will not be enough to synchronize the clocks to account for clock drift, which will cause nodes to miss their TX/RX slot. If this becomes a problem, we have a GPS we could use as a backup in order to get the exact time and synchronize the test node.

If the ignition of a match does not trigger the temperature sensor such that the difference between the maximum and minimum temperature readings or the average reading does not meet a pre-determined threshold (used for fire detection), we will simulate the detection of a fire by encoding detection in 1 bit in node's DATA packet.

9 RELATED WORK

There are other projects that are documented about creating a wireless sensor networks. For example, there is study completed by researchers in the University of the Coast in Barranquilla where they also created a wireless sensor network for fire detection [11]. This project focuses more on the sensing aspect of the fire detection system and how to determine if a fire is at a location. There are also projects that use machine learning to take the aggregate data to predict the future locations of fires [1]. In comparison to those projects, due to our use case as well as the constraints of budgeting and time, our project focuses on the network as we want to create a accurate and low power system for fire detection. Low power leads to lower maintenance by firefighters and that was one of our goals. We also are focusing our project mainly on the network and how to adjust that architecture to make our project most efficient. More than determining the fire, we are focusing on whether the location of the fire can be relayed correctly.

10 SUMMARY

In our project, we hope to create a low power and accurate system for wildfire detection. This would consist

of a wireless sensor network of 8 nodes that transmit data regarding the locations of the fire. This data would be displayed on a web application for wildland firefighters. This is intended to be low power, leading to low maintenance as well as accurate. The system would help pinpoint the fire's location. We are expecting the integration of the individual parts to be a challenge, as the time that the network will take to interpret and send data might adjust the network's design. We are hoping to keep the time constraints and ensure low power. We are determined to make this product for the convenience and safety of wildland firefighters.

Glossary of Acronyms

- MQTT – Message Queuing Telemetry Transport
- OBD – On-Board Diagnostics
- RPi – Raspberry Pi
- TS - the timeslot duration in seconds.
- TDMA - Time Division Multiple Access
- MAC - Medium Access Control
- LSA - Link State Advertisement
- LS - Link State
- LoRa-WAN - Cloud-based MAC protocol designed for the LORA-PHY (physical) layer
- ToA - Time-on-air: time from when a signal is transmitted by a sender until the receiver receives it.
- DC_TS - Data Collection Timeslot (seconds)
- TX_TS - Transmission Timeslot (seconds)
- RX_TS - Reception Timeslot (seconds)

References

[1] U. Dampage, L. Bandaranayake, and R. et al. Wanasinghe. "Forest fire detection system using wireless sensor networks and machine learning". In: *Sci Rep* 12 (2022), p. 46.

[2] Ana Elisa Ferreira et al. "A study of the LoRa signal propagation in forest, urban, and suburban environments". In: *Annals of Telecommunications - annales des télécommunications* (July 2020). DOI: ff10.1007/s12243-020-00789-wff.ffhal-02907283f.

[3] *GeoDjango*. URL: https://docs.djangoproject.com/en/4.1/ref/contrib/gis/.

[4] Dimitrios Koutsonikolas et al. "TDM MAC protocol design and implementation for wireless mesh networks". In: Jan. 2008, p. 28. DOI: 10.1145/1544012.1544040.

[5] *LoRa*. URL: https://lora.readthedocs.io/en/latest/#:~:text=When%20a%20signal%20is%20send,device%2C%20or%20system%20is%20operated.

[6] *LoRa Calculator*. URL: https://www.semtech.com/search/results_documents/lora%20calculator*.

[7] *Low Voltage Temperature Sensors*. D00337-0-5/15. Rev. H. Analog Devices.

[8] *STM32 ultra low power mcus*. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32-ultra-low-power-mcus.html.

[9] *Ultra-low-power Arm Cortex-M4 32-bit MCU+FPU, 100DMIPS, up to 128KB Flash, 40KB SRAM, analog, ext. SMPS*. DS12469. Rev. 8. STMicroelectronics. Nov. 2020.

[10] Richard Umoffia. *How to send web push notifications from Django Applications*. 2018. URL: https://www.digitalocean.com/community/tutorials/how-to-send-web-push-notifications-from-django-applications.

[11] Noel Varela et al. "Wireless sensor network for forest fire detection". In: *Procedia Computer Science* 175 (2020). The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology, pp. 435–440. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2020.07.061. URL: https://www.sciencedirect.com/science/article/pii/S1877050920317427.

Table 1: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Temperature Sensor	TMP36	Adafruit	8	$2.75	$22.00
Nucleo Board	NUCLEO-L412KB	Digi-Key	10	$11.00	$110.00
LoRA Transceiver	RYLR896	REYAX	6	$19.50	$117.00
Breadboard	3.25” x 2.2”	Adafruit	9	$4.95	$44.55
Raspberry Pi 4	B	CanaKit	1	$0.00	$0.00
Li-ion Battery	3.7V/2000mAh	Adafruit	1	$12.50	$12.50
Li-ion Regulator	L4931-3.3 TO-92	Adafruit	1	$1.50	$1.50
Li-ion Charger	v1.2	Adafruit	1	$5.95	$5.95
Li-ion Adapter	JST-PH	Adafruit	1	$2.50	$2.50
					$316.00

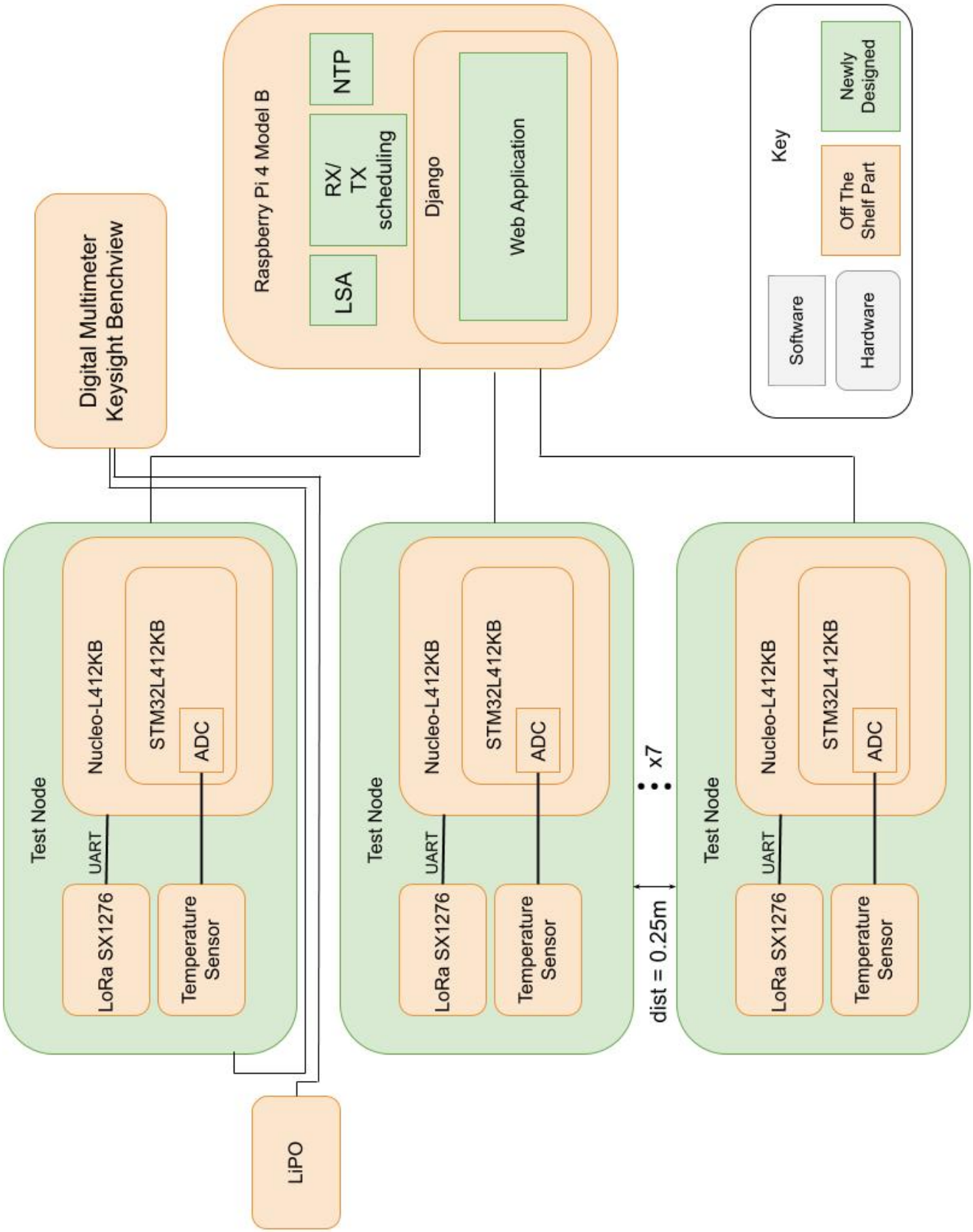


Figure 11: Full-page version block diagram.

TASKS	Week 3	Week 4	Week 5	Week 6	Week 7	Fall Break	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
	9/12	9/17	9/26	10/02	10/10	10/17	10/24	10/31	11/07	11/14	11/21	11/28	12/04
Class Assignments													
Project Proposal Presentation		DUE											
Design Presentation				DUE									
Design Document					DUE								
Ethics Assignment							DUE						
Interim Demo									DUE				
Final Presentation													DUE
Overall Project Work													
Research Equipment/Materials													
Order Equipment/Materials													
Individual Parts Development													
Integrating Nodes + Network													
Integrating Network + WebApp													
Whole Project Integration													
Slack Time													
Web Application													
Create a dummy web server to test Django/RPi													
Interface with nodes and router shown													
Animation during conflagration													
Notification system													
Node Architecture/Embedded System													
Put node into low power mode on a schedule													
Nodes listen, take measurement, transmit data													
Nodes accept beacn and syncs clocks													
Nodes are synced and listen/transmit/sleep on time													
Routing Network System													
Connecting singular node to border gateway													
Connecting star topology of nodes to border gateway													
Connecting all nodes to border gateway													
Spanning Tree Protocol Implementation													
Data Scheduling													
Data Mode for all nodes													
Integration of protocol phases and low power nodes													
Testing of protocol stack and debugging													

Figure 12: Gantt Chart