FireWxNet: A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments*

Carl Hartung, Richard Han
Department of Computer Science
University of Colorado, Boulder
Boulder, CO USA
carl.hartung@colorado.edu,
richard.han@colorado.edu

Carl Seielstad, Saxon Holbrook
Department of Forestry
University of Montana
Missoula, MT USA
carl@ntsg.umt.edu,
saxon@ntsg.umt.edu

ABSTRACT

In this paper we present FireWxNet, a multi-tiered portable wireless system for monitoring weather conditions in rugged wildland fire environments. FireWxNet provides the fire fighting community the ability to safely and easily measure and view fire and weather conditions over a wide range of locations and elevations within forest fires. This previously unattainable information allows fire behavior analysts to better predict fire behavior, heightening safety considerations. Our system uses a tiered structure beginning with directional radios to stretch deployment capabilities into the wilderness far beyond current infrastructures. At the end point of our system we designed and integrated a multi-hop sensor network to provide environmental data. We also integrated web-enabled surveillance cameras to provide visual data. This paper describes a week long full system deployment utilizing 3 sensor networks and 2 web-cams in the Selway-Salmon Complex Fires of 2005. We perform an analysis of system performance and present observations and lessons gained from our deployment.

Categories and Subject Descriptors

J.2 [**Computer Applications**]: Physical Sciences and Engineering—*Earth and atmospheric sciences*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communications*

General Terms

Design, Experimentation, Performance, Reliability

*This work was supported by the National Science Foundation (NSF) project: *SENSORS: Collaborative Research: Self-configuring In-Situ Wireless Sensor Networks For Prescribed Fire Management* (grant number 0330466)

MobiSys'06, June 19–22, 2006, Uppsala, Sweden.

Keywords

Wireless Sensor Networks, Fire, Forest Fires, Applications, Deployments

1. INTRODUCTION AND MOTIVATION

Wildland firefighting has long been a dangerous, though necessary task during the summer months across the globe. Last year over 77,000 fires burned nearly 7 million acres in the U.S. alone costing over $890 million in suppression costs[13]. With tens of thousands of firefighters dispatched to fight these fires each year, safety is the number one priority.

Fire behavior can change rapidly due to a variety of environmental conditions such as temperature, relative humidity, and wind. Moreover, these environmental conditions can differ significantly between topographical features such as elevation and aspect. Hence, the ability to accurately monitor these environmental conditions over a wide area becomes of paramount importance.

One of the primary factors influencing safety in wildland firefighting is the ability to accurately predict the fire's behavior. Such predictions are usually based on a combination of current observations, spot weather forecasts provided by the National Weather Service, and recorded weather observations from the previous few days. While such predictions can give a general picture of expected fire behavior for a region, actual fire behavior can vary tremendously over relatively small changes in elevation due to varying weather conditions. For example, it is generally expected that as elevation increases air temperature decreases. However, well known phenomena such as *thermal belts* and *temperature inversions* can cause bands of warmer air to exist at higher elevations. These conditions, which normally occur in the evenings and overnight, are caused when cold air near the surface of the mountains moves down into the valleys forcing the warmer air to rise. As the warmer air rises it gets trapped by continuously moving air above the ridge tops. The ability to detect thermal belts and inversions is of great importance to the fire community. Since temperatures in these regions stay warmer with lower relative humidities than surrounding areas, fires tend to stay more active.

There are a variety of methods that the fire community currently uses to measure weather conditions. Perhaps the

Figure 1: A fire fighter takes wind measurements using a belt-weather kit.

most common way of measuring weather on a fire is the use of a belt-weather kit as seen in Figure 1. Generally, one firefighter is selected per squad to carry such a kit, take measurements every hour or so, and report the data back to base camp. It typically takes between 5 and 10 minutes to collect accurate results and report them. The incident commanders back at base camp use this information to help determine where to position units and when to pull them away from a fire. However, there are several drawbacks to this method of measurement. First, it only provides data for areas where squads are located. Squads are highly mobile, and their locations are generally determined by fire intensities and suppression goals. Therefore, it is unlikely they will cover wide ranges of area or elevation. Also, this simple yet important task can be easily forgotten when battling a fire.

The United States Forest Service (USFS) also maintains a network of around 2,200 permanent Remote Automated Weather Stations (RAWS)[18] to help track weather conditions for a given area. In addition to temperature, wind speed and direction, and relative humidity, RAWS stations also measure precipitation, barometric pressure, fuel moisture and temperature, and soil moisture. While the data collected from the RAWS is extremely useful, they tend to be sparsely located. For example, there was only one RAWS located in the vicinity of the Selway-Salmon Complex Fires where we deployed our system. Furthermore, the positioning of such stations is not always representative of the surrounding area. The RAWS station in our fire complex was placed at the Hell's Half Acre peak located at an elevation of almost 2,500m. Many of the fires were burning much lower in the valleys closer to 1,500m. Also, RAWS anemometers, which measure wind speed and direction, are placed a minimum of 6.1m from the ground. Most forest fires tend to burn along the ground, only occasionally torching a tree, and 6.1m may not be representative of wind speeds nearer to the ground. Another problem with RAWS stations is that their large size prevents them from being portable. Even if they were smaller, there is a fairly lengthy process to register them with the USFS, and their data interfaces do not support mobility. The RAWS relay their data to a Geostationary Operational Environmental Satellite (GOES). The satellite then transmits the data to the Weather Management Information System (WIMS) where the data is made available online.

Working together with fire fighting teams and fire researchers, we set out to build a system that could report weather conditions over a variety of elevations from anywhere within the fire environment. Together, we formulated a set of requirements that drove our design:

- **Weather Data**
  Our system needed to be able to report temperature, relative humidity, and wind speed and direction. These elements have the greatest immediate influence on fire behavior. The data needed to reported about once every half-hour to an hour, 24 hours a day.

- **Visual Data**
  We needed to have 'eyes' on the fire 24 hours a day. Generally, personnel leave the fire before dark, and the overnight fire behavior is guessed based on weather conditions. Also, seeing a fire is the only way to know how it is actually behaving; therefore, we wanted to give Incident Command a readily accessible view of current conditions.

- **Elevational Gradient in Rugged Terrain**
  Our system had to be able to provide data over a wide range of elevations in potentially extremely rugged mountainous and forested terrain. Our goal was to be able to cover at least one kilometer of elevation.

- **Long Range Remote Monitoring**
  With no available electricity and no communication infrastructure, our system needed the ability to transmit data upwards of 150 Km in order to relay information from the deployment areas to Incident Command.

- **Power Efficient**
  Since all of our nodes were in rough terrain and many were only reachable in a timely manner by helicopter, it was very important that our network function for long periods of time. In examining the length of time fires typically burn, we determined that 3 weeks per set of batteries would be sufficient. Further, all sensor nodes needed to run with AA batteries to comply with informal fire standards. Currently, almost all electronic equipment used on wildland fires employ AA batteries.

- **Simple and Robust**
  Generally, our network would be set up by people with minimal experience using sensor nodes or embedded computers. Hence, we needed our system to be simple to deploy and use. Further, since our network was deployed in a fire environment, it needed to continue to function in the presence of node failures.

- **Low Cost**
  Though we attempted to deploy in safer areas to minimize harm to our equipment, losing nodes is always a possibility in an environment surrounded by fire. As such, we needed hardware that could be easily and

cheaply replaced. A loose goal was for the overall cost of the hardware and software to not exceed $20,000.

- **Portable**
  Fires usually only last between 2-8 weeks, and they rarely strike in the same area more than once until many years have passed. Thus, our system needed to be portable so we could deploy at multiple fires in different locations throughout the fire season.

We successfully converted our requirements into a fully implemented system that we deployed with the help of the Northern Rockies #1 FUMT (Fire Use Management Team) in the Bitterroot National Forest in Idaho (USA) in 2005. There were over 50 lightning strike fires in that area that burned over 35,000 acres and became known as the Selway-Salmon Complex Fires. The system we deployed consisted of 5 long range links, 3 sensor networks consisting of a total of 13 nodes, 5 wireless access points, and 2 web cameras. The access points and web cameras were deployed for just over 3 weeks, while the sensor networks were deployed for just over a week. We referred to our system as the 'Fire Weather Net'. Generally, people abbreviated 'Weather Net' as 'WxNet', which is how our system received its name.

Every member of our deployment team received wildland fire training and achieved at least a minimal certification as a FireFighter Type 2 (FFT2). With this certification each member also received a *Red Card* which authorizes the individual to work in a wildland fire environment. Achieving this certification required a week-long course in fire safety and organization, as well as a physical test where each member had to carry a 45 pound (20 Kg) pack 3 miles (4.82 Km) in under 45 minutes.

The remainder of the paper is organized as follows: Section 2 talks about prior related work in this area. In section 3 we talk about design and implementation of FireWxNet. Section 4 presents results from our deployment as well as observations and lessons learned from our deployment. Section 5 discusses future directions we intend to pursue with our system. Finally, section 6 concludes the paper.

2. RELATED WORK

Applications using wireless sensor networks have become increasingly common. Over the past few years, the capabilities of these deployments has evolved tremendously from the initial single-hop 'sense and send' deployments to scalable multi-hop deployments. This section discusses applications related to our work and describes how our contributions further this area of research.

From the onset of sensor network technology, monitoring environmental conditions or habitat monitoring has been at the forefront of the application space. In one of the first successful demonstrations of a sensor network deployment, researchers at the University of California, Berkeley deployed a sensor network at Great Duck Island off the coast of Maine[10]. They placed their sensors in burrows and used heat to detect the presence of nesting birds, providing invaluable data to biological researchers. Additionally, their work provided helpful observations about many deployment aspects such as performance, routing, and topology construction.

Similarly, researchers for the Center for Embedded Networked Sensing deployed a sensor network into the James Reserve Forest in California with purposes from soil temperature monitoring to tracking wildlife[3]. Their work extended sensor network research by using multi-hop routing and multiple, heterogeneous nodes. Other habitat monitoring deployments that have been used for monitoring specific species include a system to monitor Cane Toad populations[8], and a system for tracking the movements of Zebras[9].

Figure 2: Some deployment locations could only be reached by helicopter.

On a smaller scale, a sensor network was recently deployed on a single redwood tree using 33 nodes to cover roughly 50m[17]. With this unique deployment researchers were able to map the differences in the microclimate over a single tree.

Deployments in rugged terrain and under extremely harsh conditions have just begun to be developed. A group of researchers from Harvard recently deployed a sensor network on an active volcano in South America to monitor seismic activity and similar conditions related to volcanic eruptions[19].

Most relevant to our project, researchers from the University of California, Berkeley demonstrated the feasibility of sensor networking technology in a fire environment with their FireBug application[5]. They deployed a 10 node network in a field and successfully measured important environmental conditions such as relative humidity and temperature as a flame front passed during a prescribed burn. Though our nodes are designed for the encounter, we stayed away from attempting to track or measure flame fronts. The fire community currently utilizes high-tech airborne infrared sensors to track flame fronts and intensities over very large scale areas.

In contrast to many of these application deployments, our sensor network is distinguished by its especially rugged and unique deployment environment, its emphasis on robust design, and its relatively sparse deployment. The task of deploying our in-situ network was particularly severe, given the rugged mountainous and forested terrain over which FireWxNet was spread. Our network covers a unique topology which has not been studied before, ranging from sub-

stantial and sharp elevational differences to a fairly wide coverage area spanning about 160 square Kilometers. The network had to be capable of providing both wide area communication coverage and fine-grained local weather sensing coverage. The large elevational differences between our nodes resulted in very different radio propagation models from typical flat-ground, short distance, or in-building deployments. In many cases, the terrain was so inaccessible that nodes had to be brought to the peak of a mountain by helicopter, as in Figure 2, and then deployed down the steep slope of the mountain. An important design point was therefore robustness in all aspects of the system, ranging from robust physical equipment to robust network routing protocols. The helicopter itself further imposed weight and volume limits, i.e. helicopters cannot carry much weight at high altitudes. Only a limited number of sensor nodes with housings could be ferried to the top of the mountain. Once at the deployment locations, firefighters needed to navigate steep terrain which made it impossible to carry more than two or three sensor packages (nodes, sensors, and shelters) at a time. As a result, our weather sensor networks sought to maximize the information return for each placed sensor node, i.e. our networks were strategically and sparsely deployed to cover as much meaningful terrain with as few nodes as possible. We could not fall back upon dense sensor deployments to provide fault tolerance and redundancy both for the sensed data and for relaying of that sensed data.

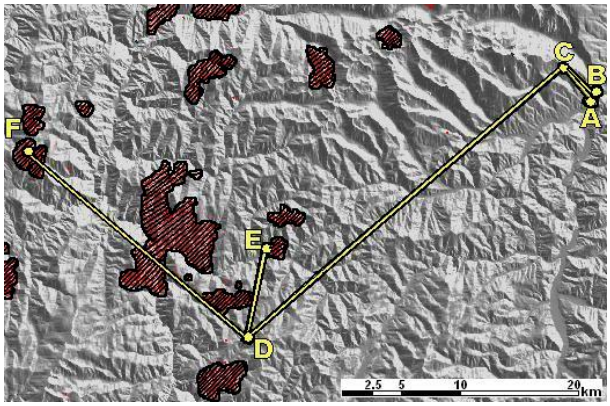


Figure 3: Our network setup relative to a few of the major fires in the area (shown as darkened areas). Locations and equipment were as follows:
A) Incident Command: Backhaul Link, WAP, Satellite
B) USFS District Office: Backhaul Link, WAP
C) Boulder Peak: Backhaul Links(3), WAP
D) Hell's Half Acre: Backhaul Links(3), WAP, Webcam, Sensor Net
E) Kit Carson: Backhaul Link, WAP, Sensor Net
F) Spot Mountain: Backhaul Link, WAP, Webcam, Sensor Net

3. SYSTEM DESIGN AND IMPLEMENTATION

We developed a tiered system of wireless technologies because such an integrated architecture was well suited to our design requirements for long range communication coverage, local area weather sensing coverage, portability, low cost, and robustness. Deploying a wireless weather network hundreds of kilometers into the wilderness proved quite a challenge. Our system needed to relay data from many points of interest to our base camp (Incident Command) through an area with no internet connectivity or even electricity. At the base of our system was the satellite uplink we used for internet access. The satellite dish provided internet access for our entire system as well as all of the administration at Incident Command. The dish was then connected to our *backhaul* network tier, a series of radios with directional antennas that created wireless links from 3-50 Km long. Finally, at the end of each set of radios we connected the *weather network*. The weather network consisted of multiple sensor nodes with wireless links up to 400m as well as a steerable webcam. Figure 4 shows a typical example of the topology for a deployment site in our system.

3.1 Network Setup

For our deployment, we used five long distance wireless links in our backhaul, three sensor networks, and two webcameras. The cameras were set up at Hells Half Acre and Spot Mountain. The sensor networks set up at Hells Half Acre, Kit Carson, and Spot Mountain consisted of six nodes, five nodes, and two nodes respectively. Figure 3 gives an overview of our entire topology in relation to the fires, superimposed on a topographic map of the area. The Hells Half Acre (D), Kit Carson (E), and Spot Mountain (F) locations were strategically chosen to maximize the value of the weather information retrieved due to their proximity to multiple fires. The Boulder Peak (C) location was chosen since it was the highest peak with a good line-of-sight to base camp. The nodes were placed sufficiently near to the fires of interest, but not so near as to imperil the firefighters or the equipment for the duration of the observation.

3.2 Backhaul Network Tier

3.2.1 Hardware

Since base camp was set up near a small ranger station, we had access to electricity, but the nearest internet connection was nearly 65 Km away. To solve this problem, we used a small portable satellite transceiver for our internet connection. The satellite, manufactured by Skycasters[15], provided the camp with speeds of 128 kbps up and 512 kbps down. After initial configuration, the satellite required almost no setup as it automatically searched for the best satellite link and oriented itself appropriately. The satellite connected to our backhaul simply using standard ethernet through an ethernet switch.

For the main links in our backhaul we used two different types of radios made by TrangoBroadband Wireless: The Trango Access5830 and the Trango M900S Access Point / Subscriber Module Radios (AP/SU). Our primary radios were the Access5830s. These radios were strictly point-to-point directional radios that used polarized directional antennas to achieve a range of roughly 50 kilometers. They operated at 10 megabits per second in the frequency range of 900Mhz-930Mhz. For shorter links we used the M900S AP/SU radios. These radios formed a point-to-multipoint setup where the subscriber modules all communicated with a single access point. To increase the communication distance of these radios we attached an external Yagi antenna

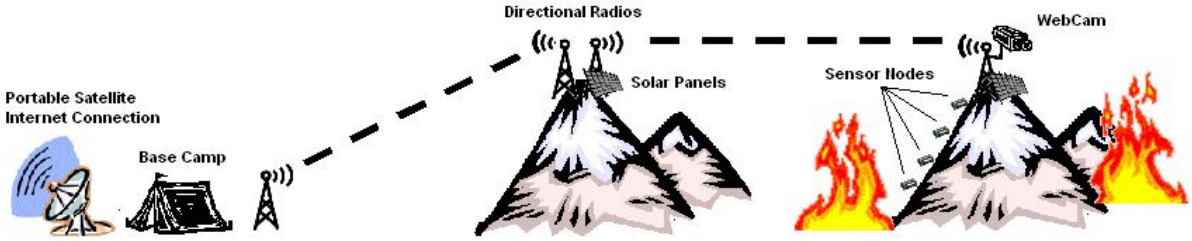


Figure 4: System Overview: Radios with directional antennas were used at each peak and at the basecamp to relay data from our sensor network and webcam.

Figure 5: The Backhaul link set up on Boulder Peak. To the near side is an Access5830 with a directional antenna. On the far side is a M900S connected to a Yagi antenna. Below are two of the solar panels we used to provide power.

which extended the range to just over 32 kilometers. These radios also operated in the 900Mhz-930Mhz range, but were slightly slower at 3 megabits per second. All the Trango radios used the standard 802.11 and TCP/IP protocols for communication, and were manually given IP addresses prior to deployment.

Though the 900Mhz spectrum gave us increased range in our wireless hops as compared to 2.4Ghz, it did present a small problem. Our sensor nodes also operated in the 900Mhz range, and the initial configuration values of each were close enough to interfere with each other. In fact, our entire base camp was so flooded with signal that the CSMA protocol implemented on the sensor nodes would back off indefinitely. We fixed this issue by configuring the Trango radios to communicate using the 924Mhz frequency.

We mounted the radios and antennas to a pole 1-2 meters off the ground which was then mounted to a secure base. The radios were all powered with a power-over-ethernet setup using a 24V power supply. Once set up, the Access5830 radios required a single user at each end in order to fine-tune the antenna direction. Since the M900S radios were many-to-one, only one user at the SU was required to fine tune the connection. All of the radios provided both a web-based and a command-line telnet interface to help the user align the antennas to ensure maximum signal strength. Figure 5 shows an example setup. Once aligned, the radios functioned exactly as a wired ethernet link. At each hop radios were connected to the next hop via an ethernet switch.

The ethernet switches at each hop were Linksys WRTG45 4-port Wireless Access Point (WAP) switches. This meant that every radio hop in our network also provided standard 802.11 WiFi internet access to any units in the area. We frequently took advantage of this feature since we could monitor and manage all of our sensor nets and web cameras from anywhere in proximity to our network.

3.2.2 Power

With no access to electricity between the Incident Command and any hop in our backhaul, we decided to use solar panels and large batteries to power the various equipment. At each hop in our backhaul we set up two solar panels, a 24V and a 12V, and four 12V batteries. During the day the solar panels produced enough energy to both run the system and charge the batteries, while at night the system ran solely from the batteries. Even with the rapidly decreasing daylight during the fall in northern Idaho we were able to keep all of the radios fully powered and connected for the length of our deployment. Further, all of the switches, access points, and web cameras were powered by the batteries and solar arrays as well.

To protect the equipment from moisture, animals, and other hazards we placed the switches, access points, and base stations inside a large plastic waterproof briefcase made by Pelican. We drilled holes in the back of the case to run wires, and then sealed the holes with electrical tape. Since the cases were rather thick black plastic, we worried that with the temperatures reaching upwards of 33^{o}C, the equipment would overheat. Especially worrisome was the lack of airflow inside the cases since none of the equipment inside contained even a single fan. However, we found this to not be a problem even when the cases spent most of the day in direct sunlight.

3.3 Weather Network Hardware

The weather networks consisted of a number of sensor nodes, a webcam, and a small embedded computer running linux. We tested our system using two different webcams. The first was a Sony SNC-RZ30N and the second was a Panasonic KX-HCM280. The basic functionality described below was the same for both cameras. We set up each inside a protective case with a clear plastic dome to protect the equipment from the elements while still allowing clear viewing as shown in Figure 7(c). Once the webcam was mounted, it only needed to be powered on and connected to an ethernet switch. The cameras required only minimal con-

figuration such as setting the camera's IP addresses prior to deployment. The webcams ran their own web servers which allowed users to connect to it from any web browser. The camera controls were also accessible through the web interface and allowed the camera to rotate a full 360 degrees and tilt up to 180 degrees. The Panasonic provided a 21x optical and 2x digital zoom, and a picture resolution of 640x480. The Sony had a 25x optical and 300x digital zoom and produced pictures with a resolution of 736x480. Both cameras provided an infra-red night vision feature and could deliver video at up to 30 frames per second. Additionally, they each required a 12V power source and used the batteries and solar arrays.

3.3.1 Base Station

Our base station provided the very important link between our sensor network and our backhaul. Figure 6 shows how the base station bridged the gap. The device we chose for our base station was the Soekris net4801 [16]. We chose this board for its small form factor, minimal power consumption, and numerous readily available peripherals. The Soekris operates with a 233Mhz NSC SC1100 processor and has 32 Mbyte SDRAM soldered on board. For onboard peripherals it contains three 10/100 Ethernet Ports, two Serial ports, and one USB port. The Soekris also boasts a CompactFLASH socket and a PCI slot. Another benefit of this system is its wide range of operating temperatures. According to the specifications, the Soekris can operate at temperatures between 0-60^{o}C. Additionally, the Soekris can run at a wide range of power levels from 6-28V. During our deployment we ran the boards at 12V.

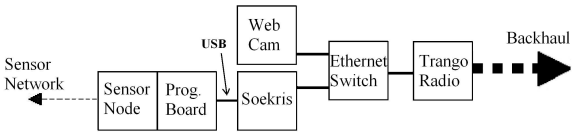


Figure 6: A block diagram of our weather network. The webcam, Soekris, and trango radio all connected to the switch via wired ethernet. The sensor node connected to the Soekris via a USB-to-serial cable.

We ran a stripped down version of Gentoo Linux from a 512Mb CompactFLASH card for our operating system on the Soekris boards. The Soekris connected to the backbone through standard, wired ethernet. However, with the PCI slot we could have attached a wireless ethernet card and used that to connect to the backbone. The wireless ethernet option would have given us more freedom in the placement of our base station. This was not necessary in our deployment since our base stations were out of harms way, however it is conceivable that in future deployments we would want to place the base stations closer to the fire and at the same time place the more expensive Trango radios and solar panels in a safer location where they would not be exposed to fire. We connected the Soekris to the sensor network through the USB port. We attached a node to a MIB510 programming board and connected the programming board to the Soekris through a USB-to-serial converter.

3.3.2 Sensor Nodes

For the nodes in the senor network we chose the Mica2 platfrom made by Xbow. The Mica2s utilize AA batteries for their power source. We cannot overstate how much this simple fact helped our system gain quick acceptance by the fire community as nearly all electronic devices used by fire crews operate with AA batteries. The Mica2 is controlled by an Atmel ATMega128 8-bit processor running at 7.37Mhz. For communications, the Mica2 uses the Chipcon CC1000 radio operating at 900Mhz. To allow for different sensing packages, the Mica2 contains an external 52-pin connector.

We developed our own sensor package using the Xbow MTS101 Basic Sensor Board and soldering the necessary sensors to it. The MTS101 comes with a built in temperature sensor, the TSI 44006, which when calibrated is accurate to within 0.2^{o}C. For our relative humidity sensor we chose the Humirel 1520 RH sensor due to its superior accuracy at low relative humidity levels. Fire activity can change rapidly with even small changes at low RH, so a high degree of accuracy when the relative humidity was below 30% was very important. We encountered one issue with the Humirel 1520 in that it required a 5V power source, but the two AA batteries on our board only produced a maximum of 3V. To solve this issue we developed a small power regulator board which took a 10mA current as input and output a 5V power source. Finally, for our anemometer we chose the Davis Standard Anemometer. The anemometer provided wind direction accurate to within 7 degrees, and wind speed to within 5% of the reported value. While knowing the location of the nodes was very important, we decided not to outfit our sensor package with GPS units. Even small GPS units tend to use an enormous amount of energy, and would significantly decrease the life of our system. Since the nodes are immobile, we decided to just carry a hand held GPS unit and record the locations of the nodes at the time of deployment.

To protect the sensor nodes from outdoor elements but still allow them to accurately record weather data, we built 20 custom enclosures as shown in figures 7(a) and 7(b). The enclosures needed to protect the node from moisture and heat, but still expose the sensors to the outdoor conditions. To accomplish this we designed the enclosures to have three ventilated sides, an open bottom, and a combination vented ceiling and roof. The sides were made with downsloping slits much like blinds in order to allow ample airflow through the apparatus while still protecting the node from falling rain. The node was attached to the ceiling of the box with heavy duty velcro. The ceiling had three 4cm diameter holes drilled in it to prevent the unit from trapping rising hot air. The roof was made slightly larger than the ceiling and separated by a .6cm inch spacer to allow air to escape. We painted the units white in order to protect the equipment from solar radiation heating. The box was then affixed at 1.5m from the ground to a 4cm diameter wooden dowel which was mounted in a metal tripod. The temperature and relative humidity sensors were located inside the box attached to the sensor node. The anemometer was mounted at the top of the dowel at a height of 1.8m. The height from the ground not only helped to alleviate the communications burden caused by Fresnel Zones (discussed in section 3.4.1), but also provided the nodes protection from heat damage caused by the fire.

(a) An enclosure we built for our sensor nodes. The open bottom, vented ceiling, and slotted walls allowed the sensors to accurately record conditions.

(b) Nodes were attached to the ceiling with industrial velcro. All sensors faced down towards the open bottom.

(c) One of the webcams used in our deployment

Figure 7: Weather Network Hardware

3.4 Weather Network Software

Constructing a reliable, self-healing, multi-hop network for an actual deployment presents many design implications and challenges that no simulator can fully emulate. Problems such as interference and asymmetric links are not only hard to simulate, but also vary greatly from deployment to deployment as we show in section 4. In order for our system to function we needed to design a robust mechanism which would ensure, with high probability, that our data would reach our base stations even in the varying presence of interference and asynchronous links. Rather than implement a protocol with guaranteed delivery, we developed a best-effort converge-cast protocol similar to [11, 14]. In place of reliability mechanisms we chose to send messages multiple times, in effect creating a forward error correction mechanism. In this way we reduced the need for every packet to reach the base station during a certain time period to only needing a single packet per node.

Rather than creating a single monolithic application, we built our sensor network on the MANTIS operating system[2]. MANTIS is a multi-threaded, embedded operating system closely resembling Unix. We chose this operating system for several reasons: First, it provides easy to use interfaces to all of the features of the nodes such as communications and energy-efficient scheduling. Second, MANTIS can be used on multiple platforms and has already been ported to the Mica2, MicaZ and TelosB nodes. Finally, all applications for the MANTIS operating system are written in the standard C programming language.

3.4.1 Deployment Issues

The fairly sparse nature of our deployment, our desire to utilize radio links as long as possible, and considerations about the topology of the area we deployed in led to some interesting deployment challenges. Our deployments were fairly linear and some had very little, if any, overlap of communication between nodes that were not adjacent to each other. This meant that we needed to be sure bi-directional links existed between nodes to ensure that data from nodes further down in the chain would reach our base station. Large changes in elevation between nodes and dense forest and/or underbrush further complicated our deployment. Due to the large change in elevation we found that the range of the radios was much greater than if the nodes were both placed on level ground. Most other deployments have used dense clusters of nodes and placed them at distances of less than 30m to ensure connectivity. In our deployment, for example, our average distance between nodes was 138m with our longest link nearly 393m. We were able to achieve such large distances by exploiting a phenomena called Fresnel Zones[20], which is basically what causes multipath interference. Fresnel Zones are a measurement of the phase difference between the reflections of radio waves between transmitter and receiver. Being out of phase can cause a canceling effect and significantly weaken the ability to receive the signal. With Fresnel Zones, the Earth is the primary cause for such reflections because the ground itself acts as a major obstacle. Therefore, the further the nodes are located from the ground, the less interference encountered. Most of the nodes in our deployment sent their radio signals from peaks to valleys and along steep hillsides where there was far less ground to cause interference. In informal testing we were able to establish a radio link of .71Km (.44 miles) between

two mica2 nodes placed on peaks with a deep valley between them. Finally, due to sometimes dense vegetation between the nodes the range of the radio was significantly shortened. We found, however, that we were usually able to find a good line-of-sight by moving just a few feet in either direction.

3.4.2 Deployment Mechanism

Taking into account the above considerations, we developed a simple yet effective deployment mechanism which enabled us to verify some connectivity between our nodes. When powered on, the nodes would start by sending LOCATE packets at the rate of 1 packet per second. The nodes would also listen for LOCATE packets and respond with a similar FOUND packet. Each of the LOCATE and FOUND packets were the size of the largest data packet sent in the network. This was because we found that smaller packets tend to transmit further distances with less packet loss than larger packets. Every time a LOCATE packet was not answered within 2 seconds by a FOUND packet the red LED would blink. Conversely, every time a FOUND packet destined for the appropriate node was received the green LED would blink. The nodes would stay in this state in order to allow for other nodes to be placed.

Once all of the nodes were placed, the base station was turned on and it began broadcasting control packets. All of the other nodes would forward the control packets using a standard flooding protocol. Upon receiving a control packet all LEDs would turn off to save power and the nodes would begin their duty cycle as explained in section 3.4.4. We found this deployment mechanism to be extremely simple and very efficient. Fire fighters deploying the nodes needed only to examine the LEDs to determine connectivity. We found quite often that a movement of a few feet to the right or left significantly increased or decreased connectivity. One downfall to our mechanism was that a user would not receive verification when adding a node to a currently deployed network. An easy workaround was simply to restart one or more of the nearest nodes which were known to be connected. They would then be back in their STARTUP phase and would send and receive the appropriate packets to aid in placing the new node. Then, when the rest of the network woke up during a duty cycle, the nodes would all rejoin the network.

3.4.3 Routing

Once the base station was powered on, it began sending out control packets (or beacons) for one minute at the rate of one every four seconds. These beacons served multiple purposes in our network including route discovery, fault tolerance, and time synchronization. Multiple beacons were sent during awake periods since our network did not use any guaranteed delivery mechanisms. The beacons were propagated through the network by a simple directed flooding algorithm where nodes retransmit control packets when the distance to base (DTB) of the originating node is less (closer to base) than its own. This can lead to nodes receiving and/or sending multiple copies of the same packet. However, due to the relatively long period of the packets, we found that more packets were better than fewer packets, i.e. we erred on the side of more redundancy to provide fault-tolerant dispersion throughout the network. When nodes sent data packets to the base station they used the same protocol in reverse. Data packets were forwarded only if the sending node's DTB was greater than the receiver. This again can create duplication at an exponential rate, but due to the linear nature of our networks only 6.9% of packets were ever duplicated. Furthermore, due to the sparse nature of our networks we found duplication actually helped to ensure packets were transmitted to the base.

3.4.4 Duty Cycling and Time Synchronization

In order to save power, our entire network ran on a duty cycle. While a duty cycle based on a wake-on-radio approach[6] would be ideal, current technology only allows such schemes to work in extremely short ranges (under 3m). Thus, we implemented our network with a 15 minute period where the nodes would sleep for 14 minutes, wake up and send packets for 1 minute, and then fall asleep again. This resulted in a 6.67% duty cycle. Our technique differed from most current duty cycling techniques where the nodes wake up much more frequently, perform one task, and fall asleep as in [7]. Ours woke up far less frequently, performed many tasks, and then fell asleep. However, since sleeping nodes would not forward packets, we needed to ensure that our entire network ran on the same 15 minute duty cycle. To accomplish this we developed a loose, relative time synchronization mechanism which performed like a simplified version of [12]. To save energy and minimize network traffic, we re-used the control beacons to serve the purpose of time synchronization as well. Each beacon contained a sequence number which the nodes used to determine when the next sleep cycle would begin using the simple equation (60 - 4 * SeqNo). The sequence number counted from 0 to 15 and then reset at the next interval. This was only calculated for the first beacon received during an awake period. Once nodes awoke, they waited to send any data until they heard a beacon. This allowed the beacons to propagate relatively unobstructed by interference, and saved power in the nodes by not sending packets when other nodes were not listening. This mechanism kept our network time synchronized with the base station at all times. Drift within the network was not a problem because the nodes would resynchronize with the base every 15 minutes. Thus, only the drift of the base station had to be monitored relative to real time. We countered drift in the base station by using the Network Time Protocol (NTP) daemon included with Gentoo.

3.4.5 Fault Tolerance

The last purpose of the beacons was to provide fault tolerance. During an awake period, the nodes in the network would listen for beacons from nodes with a DTB less than their own. If the nodes did not receive a beacon in 10 seconds (2 and 1/2 beacon cycles), they would reset their own DTB and listen for any beacon. Upon hearing a new beacon the node would reconnect to the network with a new DTB. This mechanism was useful as it allowed nodes to be reset, have their batteries replaced, create routes around failed nodes, or be moved (which actually happened in our deployment) and still continue to function in the network. During a personnel change, the new fire fighters switched two of our nodes so that their IDs would better reflect their placement on the hill. Our network recovered perfectly and there were no gaps in the data, the only difference being that their hop counts were swapped. However, we did notice some interesting phenomena due to changing asymmetric links and/or interference. This is discussed in section 4.

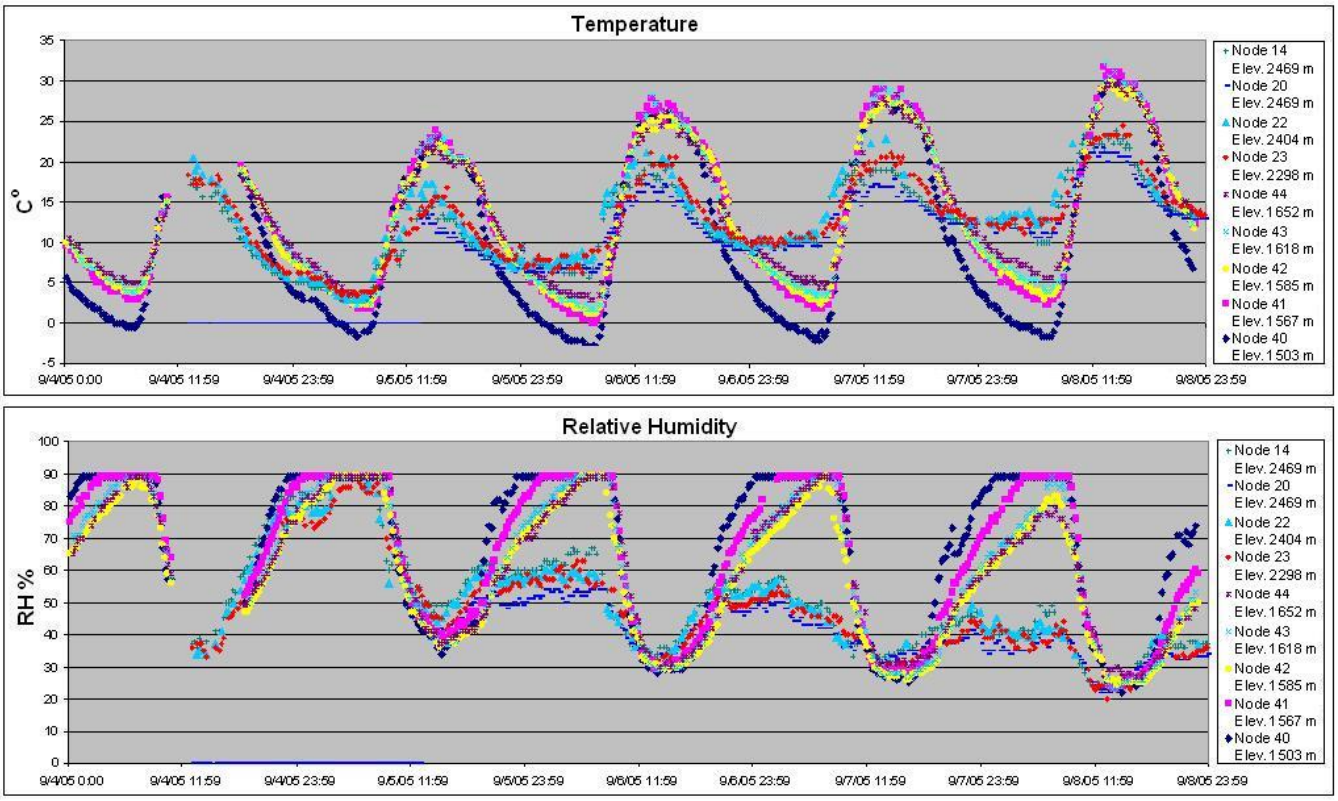


Figure 8: Temperature and Relative Humidity graphs for the length of our deployment.

3.5 Gathering the Data

In order to transmit all the data from the network to base camp, we used a number of freely available tools. The data was gathered at the Soekris and written into a tab delineated text file. A cron-job ran in the background which ftp-pushed that text file to a computer at Incident Command every 15 minutes just after each round of the network duty cycle. Alternatively, users could ssh or sftp into the Soekris and manually retrieve the logs at any time. Once at a local machine the data was generally imported into a spreadsheet or database program to be analyzed. In the future we intend to write all of our data directly to a database for easier retrieval and analysis.

4. EVALUATION

The success of our system was quickly demonstrated on the second morning of our deployment. Both visual and weather data we collected were used to locate an inversion which was announced by the fire behavior analyst in the morning briefing. They continued to use our data and announce safety precautions for the length of our deployment. In addition to those immediately useful results, we collected over 80,000 data packets containing measurements from 5 different sensors as well as 7 separate pieces of information on routing topology. This gave us a total of nearly 1 million individual data points to analyze. We present an evaluation of the scientific results, as well as an evaluation of our network performance over the course of the deployment.

A few notes about the data being analyzed:

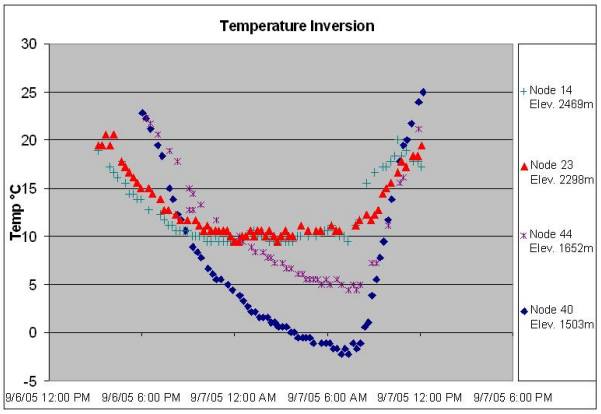


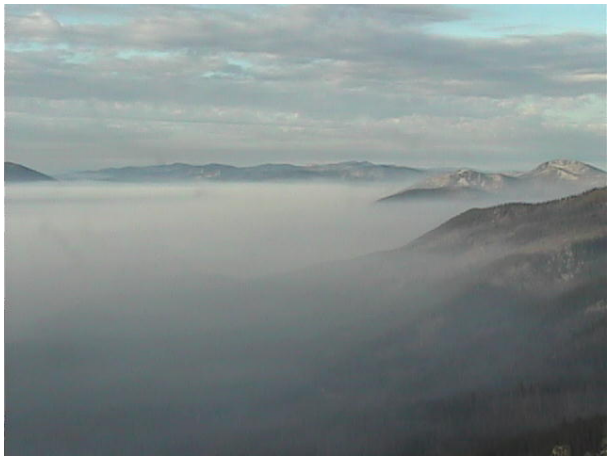
Figure 9: Close-up of a temperature inversion

- The weather network at Hell's Half Acre (Nodes 14, 20, 22, 23) was deployed from 9/4 at 13:03 until 9/9 at 15:13. The network recorded no data on 9/4 from 14:49 to 15:34 due to a problem with the base station. Due to a faulty sensor package, node 20 reported only 0 for all weather data from initial deployment until 9/5 at 13:00 when the sensor package was replaced.
- The weather network at Kit Carson (Nodes 40, 41, 42, 43, 44) was deployed from 9/2 at 19:15 until 9/10 at 19:19. The network was down between 14:10 and 15:53 on 9/3. During this time we reorganized the network for better coverage. The network recorded no data on 9/4 between 10:47 and 18:37 due to a problem with our base station. The network reported no data between 9/8 at 22:55 and 9/10 at 17:03 due to the batteries dying on the node connected to the base station.
- The weather data we present is between 9/4 and 9/9 when both weather networks were fully functional. The network analysis covers the entire time each network was active.
- Though our routing algorithm had the possibility of duplicating packets, only 6.9% of all packets were duplicated. Duplicates were removed for all routing and weather analysis.
- Our relative humidity sensor peaked at 90%, so all values above that threshold were reported as 90%. Low relative humidities (below 30%) were more important to our study.
- Though we collected wind data, we omit it from our discussion as its average and variation were negligible during our deployment.
- We ignore packet statistics from the Spot Mountain deployment as it was a single-hop deployment. Since there were no crews working in that area, we only placed one node at that location to supplement our camera with the localized weather data.

4.1 Finding a Temperature Inversion

Figure 8 shows the temperature and relative humidity changing over the length of our deployment. As expected, the relative humidity decreased while temperature increased throughout the day, and the converse is true in the evening once the sun began to set. One thing to take note of in both graphs is the large difference in oscillation of both temperature and relative humidity between the different elevations. The lower elevation sensor nodes reported very large changes in temperature throughout the day, upwards of 30^{o}C, whereas the upper elevation nodes reported much smaller changes in the range of 10^{o}C per day.

The general expectation of temperature is that it decreases as elevation increases. This trend can be observed each day during our deployment. However, other than the evening of 9/5, our data indicates that a fairly significant inversion set in each night around roughly 20:00 and did not lift until 11:00-12:00 the next day. A closer view of one such inversion can be seen in figure 9. The inversion begins when the temperature at the lower elevation nodes drops below that of the upper elevation nodes. As shown in the graph, once the inversion set in the temperature at the upper elevations stayed relatively constant, while the temperature in the lower elevations droped significantly. For example, the temperature on the Hell's Half nodes stay in the range of 10^{o}C whereas the nodes at Kit Carson drop as low as -3^{o}C. Also of note, there is almost no difference in temperature between the nodes at 2298m and 2469m. However, the lower nodes show just how quickly the temperature changes in the inversion. At Kit Carson, the node at 1500m reported temperatures 9^{o}C less than the node located only 150m above it. These large variances in temperature have a great effect on fire behavior. The higher temperatures at the higher elevations mean that fires will continue to burn throughout the night as shown in figure 10(b). However, the much colder temperatures at the lower elevations means the fire will settle down and simply smolder overnight until the temperature increases again the next day.



(a) Visual verification of the morning's temperature inversion from one of our webcams.

(b) Also captured by one of our webcams: the warmer air above the inversion allows this fire to burn actively throughout the night.

Figure 10: Effects of a temperature inversion captured by one of our web cameras.

On mornings with inversions, we were also able to quickly identify the inversions with our web cameras. Figure 10(a) shows a good example of the smoke from the fires getting caught just under the inversion. With a topographical map and the provided images we could quickly identify peaks in the images and estimate the level of the inversion. Correlating the visual image with the data from our weather network allowed us to more closely pinpoint the inversion layer.

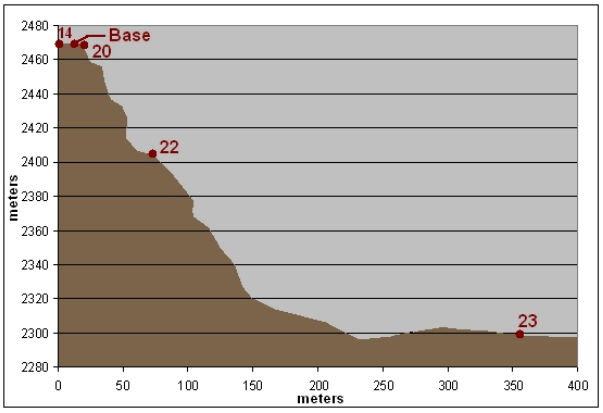
4.2 Network Performance

The two weather sensor networks were deployed as shown in Figures 11(a) and 11(b). The vertical vs. horizontal distances between our sensor nodes are shown. Though each of our deployments performed similarly in packet reception rate, each of our deployments performed extremely different in terms of building and maintaining their network topology. Our network at Hell's Half Acre maintained a steady topology which never changed during the life of the deployment. The topology of our Kit Carson deployment, however, changed significantly and often during our deployment.

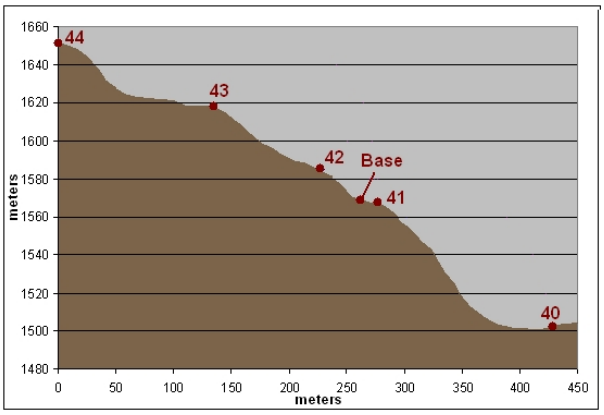
We designed our network to send 60 data packets within a 1 minute time interval every 15 minutes. However, we estimate that we were only able to send 50 data packets during that time period due to some timing limitations. For example, the RH sensor required 200ms of settling time once it was activated before it could produce an accurate reading. This caused the delay between sending packets to be 1.2 seconds rather than 1 second. Additionally, we used a CSMA MAC protocol which would back-off in the presence of interference. This would further reduce our total possible packets sent.

We received 87276 packets out of an expected 218,200 resulting in a 40% overall yield from our network. Figure 12(b) shows the day by day yield per node in our network. This performance is on par with other sensor network deployments and correlates to previous studies in multi-hop routing in sensor networks [22]. However, since each node transmitted the same sensor data for all of its packets during an awake period, we only needed to receive 1 of those packets to consider a send successful. Using that metric and counting only unique packets per node per 15 minutes, our unique yield for our deployment was a relatively high 78%. This redundancy over provisioned in favor of fault tolerance, but it also provided us with invaluable data about the effectiveness of our routing scheme.

In terms of overall performances, each of our networks performed similarly to one another. The Hell's Half deployment resulted in a 47% overall yield and a 74% unique yield. The Kit Carson deployment resulted in a 36% overall and an 80% unique yield. The Hell's Half deployment numbers are artificially lowered by Node 22, which we purposely placed in a location where our deployment mechanism was alternating between red and green LEDs, i.e. this node was only intermittently connected. We placed Node 22 as such to test the difference in performance between it and the well-connected nodes that were only flashing green LEDs. Figures 12(a) and 12(b) show how each node performed over the course of our deployment. Some trends to notice are that nodes that were further from the base station (both physically and by hop count) performed worse than those that were closer. This result is expected since with each hop in the network the chances for dropping a packet increase. Also, especially with nodes further from the base, nodes generally performed worse over time. We hypothesize that these results are related to decreasing battery life since nodes are not as 'loud' and therefore are more easily interfered with over time.



(a) Network topology at Hell's Half Acre

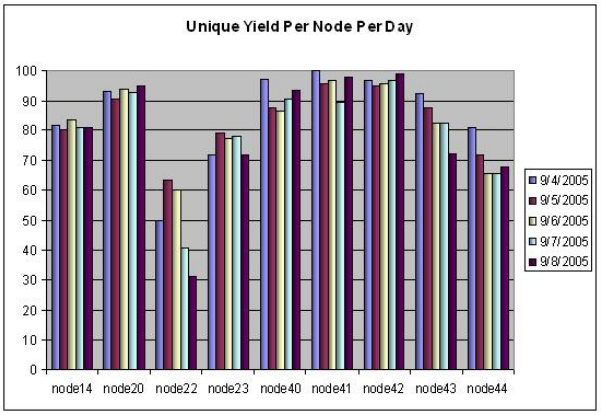


(b) Network topology at Kit Carson

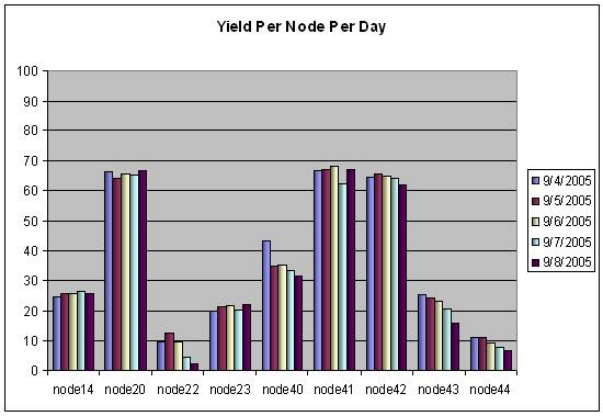
Figure 11: Elevation vs distance profiles of both sensor networks

4.2.1 *Hell's Half Deployment*

When we were deploying, we expected that Node 23 would form a link with Node 22, however Node 20 and Node 23 maintained communication throughout the deployment of our network even though they were located nearly 400m from one another. As such, Nodes 22, 23, and 14 all maintained a distance of 2 hops from the base station while Node 20 always maintained a single hop. However, about 20% of the time Node 20 reported itself as 'lost', in that it had not heard a control beacon in the specified time period. This further reduced our packet counts since 'lost' nodes would not forward data packets in order to prevent loops within the network. Because of its close proximity to the base station, we assume that its inability to receive control packets was caused by interference from other nodes.



(a) Unique yield per node per day



(b) Total yield per node per day

Figure 12: Packet yields from our deployment

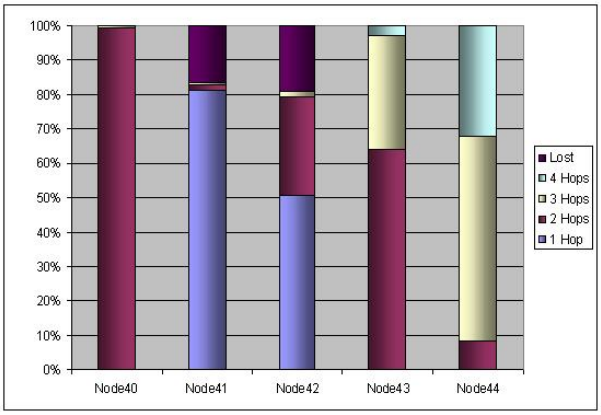


Figure 13: Variability in network topology

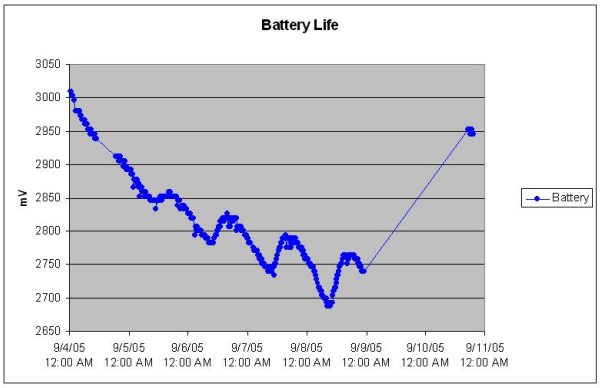


Figure 14: Battery Life

4.2.2 Kit Carson Deployment

As shown by the graph in figure 13, the topology of the Kit Carson deployment changed frequently over the course of the deployment. We attempted to correlate these changes over time, however most of the changes only lasted for a few seconds. This seemed to indicate that there were rapidly changing, random asymmetric links in our network quite often. Due to the rapid variability, it appears these asymmetries lasted only on the order of seconds or tens of seconds, and were most likely caused by interference between the nodes. These results are similar to other studies on radio irregularities[23]. As such, in both networks our routing algorithm was a little overaggressive in attempting to reconnect to the network. Algorithms that function on link or route quality such as [4, 21] would be more appropriate for such situations where interference causes short duration asymmetries, and are an aspect of our future work.

4.3 Battery Performance

Figure 14 shows the performance of Node 40 during our deployment. The jump near the end of the graph shows when the batteries were replaced. Though we had implemented our duty-cycling mechanism, we had improperly set the sleep-mode on the processor to idle rather than deep-sleep. Hence, during our deployment our batteries lasted a maximum of 5 days. We have since corrected the issue and a deployment within our lab lasted over 5 weeks, well beyond our stated goal of 3 weeks. However, our mistake exacerbated a different issue we had not considered for our deployment: temperature has a significant effect on the battery's available power. As shown in the graph, each day as the temperature dropped the performance of the battery dropped significantly. As the temperature rose, the available power in the battery rose as well. In freezing temperatures, nodes could appear to 'die' only to begin reporting once they warmed up again. This never occurred during our deployment, as nodes physically connected to our base station always depleted their batteries and stopped recording first. This was caused because the nodes used an additional 14mA when they were connected to a programming board regardless of sleep-state. The programming boards, however, offer an external power connection which we plan to use in future deployments to alleviate the programming board's extra power requirement. This oscillation correlated with temperature could create problems in our and other deploy-

ments as battery life will effectively be significantly reduced. The network could only be able to report for portions of the day when the temperature is above a certain temperature threshold related to the remaining battery level.

5. DISCUSSION AND FUTURE WORK

We learned a great deal through our collaborative effort with the fire fighters and fire researchers. While many aspects of a system may work well in simulation or small-scale tests, actual deployment of our system taught us the most. Many assumptions that hold true in simulation or devised tests disappear during actual deployments. In short, the biggest lesson we learned is that there is no substitute for the practical experience of actually deploying a full system. In this section, we review the lessons we learned and discuss future directions we intend to take this work.

With our system, we accomplished the following:

- Demonstrated that a sparsely deployed heterogeneous wireless system can provide meaningful scientific data to the fire community. This data consisted of environmental measurements as well as visual images.
- Deployed over a wide range of elevations in rugged terrain.
- Implemented effective multi-hop routing, power saving duty-cycling, and fault tolerant robustness in our sensor networks.
- Developed our system for roughly $22,000, slightly more than our cost estimate.

With overall packet loss rates averaging over 50% for multi-hop sensor networks, routing protocols must be wisely chosen to improve success. In our case, we chose to resend the packets multiple times which improved our success rate to nearly 80%, but at the cost of some efficiency. We are currently experimenting with protocols containing small reliability mechanisms such, as hop-to-hop ACKs, hoping that in future deployments we can cut down on the number of packets we send. This should, in turn, save energy and help our system last even longer. Additionally, fewer packets will also hopefully reduce the amount of interference within our networks.

Even with high loss rates, we found we were fairly overwhelmed with the amount of raw data coming through our system in real-time. We wrote a handful of fairly simple python scripts to transform the raw data into meaningful and usable information. However, we believe there is still much to be explored in terms of creating applications to represent such wide arrays of data to end-users. For example, we hypothesize that our system would benefit most from a web application that could display not only current, but also historical conditions (including images) to a large number of distributed users. Piping the data directly to a database would be a good start, but how to present the data remains an open question.

Quite surprising in our deployment was the ability of our sensor nodes to create wireless links over 10 times further than any current deployment. These long range links were made possible by the large elevational differences between our nodes, which greatly reduced interference caused by the ground. With such variety in our links, a simple visual deployment mechanism was necessary to help ensure bi-directional links existed between the sparsely deployed nodes in our network. More sophisticated deployment mechanisms may help to further increase overall yields in sensor networks. We intend to leverage this knowledge in future deployments to help determine node placement and build routing protocols which can take advantage of similar topographical features.

With sensor networking technology still rapidly evolving, we intend to migrate our application to new mote platforms which have better radios, better sleep capability, and more functionality such as the TelosB. We hope such improvements, especially with the radios, will also further improve our packet yields.

To help improve the remote management capabilities of our network, we would like to incorporate remote code updates into our system. The Mantis OS already supports this feature, and we hope to exploit it in future deployments.

In the field, we used belt weather kits to calibrate our sensors. This proved to be effective, but a rather long process even for a relatively small number of nodes. In the future, we intend to use a wind-tunnel to calibrate all of our sensors prior to deployment.

Though our system was deployed in an uncontrolled fire environment, our system could easily be used for site monitoring prior to and during prescribed burns. For future deployments we intend to include more sensors such as a tip-bucket to measure precipitation and a solar radiation sensor to measure the sun's affects on fuels. After adding such sensors it will be possible to deploy our system for an even greater variety of studies.

The portability of our system was tested only on one set of fires. Our entire system, including backhaul network and weather nodes, was deployed and then retrieved successfully. We were planning to redeploy FireWxNet to other fires later in the fire season, but a season-ending event (the first snow of the season) intervened. We hope to redeploy next spring.

Fire behavior analysts already have and are continuously developing a multitude of fire models[1] they use to predict fire behavior for a wide variety of situations. We hope that with future deployments of our system we can continue to work closely with fire analysts to incorporate our data into such models to help improve their accuracy.

6. CONCLUSIONS

Just as long-range wireless technology has been praised as a means of bringing communications to remote areas, short range sensor networks have been lauded as a means of gathering large amounts of data from small areas. We blended these two ideals into an actual real-world deployment that combines the best of both technologies. In so doing, we built a system that successfully presented an elevational gradient of environmental conditions in wildland fire environments. This previously unattainable information will help fire behavior analysts make better predictions about fire conditions and create a 'more aware' environment in the fire community, which will in turn help make fighting forest fires safer in the future.

7. ACKNOWLEDGEMENTS

Special thanks to the Northern Rockies #1 FUMT (Fire Use Management Team) and the members of the Fire Intelligence Module from the National Center for Landscape

Fire Analysis at the University of Montana for inviting us to the fire and helping us deploy our system. Also thank you to Erich Nahum for early conceptual discussions on fire sensor networks.

8. REFERENCES

[1] M. Behar. Rendering inferno. In *Wired Magazine, Issue 12.10*, October 2004.

[2] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. In *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on Wireless Sensor Networks*, 2005.

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM Sigcomm Workshop on Data Communications*, San Jose, Costa Rica, April 2001.

[4] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM Mobicom*, September 2004.

[5] D. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. In *SPIE Symposium on Smart Structures & Materials*, San Diego, CA, March 2005.

[6] L. Gu and J. Stankovic. Radio-triggered wake-up capability for sensor networks. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS04)*, 2004.

[7] T. He, S. Krishnamurthy, J. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu. Energy-efficient surveillance system using wireless sensor networks. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys '04)*, June 2004.

[8] W. Hu, V. Tran, N. Bulusu, C. tung Chou, S. Jha, and A. Taylor. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *Information Processing in Sensor Networks (IPSN 2005)*, Los Angeles, CA, April 2005.

[9] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradoffs and early experiences with zebranet. In *Arthicetrual Support for Programming Languages and Operating Systems (ASPLOS 2002)*, October 2002.

[10] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *1st ACM International Workshop on Wireless Sensor Networks and Applicatoins (WSNA 2002)*, Atlanta, GA, September 2002.

[11] M. Maroti. The directed flood routing framework. In *ACM/IFIP/USENIX 5th International Middleware Conference*, October 2004.

[12] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Second ACM Conference on Embedded Networked Sensor Systems (Sensys)*, November 2004.

[13] National Interagency Fire Center. http://www.nifc.gov/stats/wildlandfirestats.html.

[14] G. Simon, G. Balogh, G. Pap, M. Maroti, B. Kusy, J. Sallai, A. Ledeczi, A. Nadas, and K. Frampton. Sensor network-based countersniper system. In *Second ACM Conference on Embedded Networked Sensor Systems (Sensys)*, 2004.

[15] Skycasters. Satellite Internet. http://www.skycasters.com.

[16] Soekris Engineering. http://www.soekris.com.

[17] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Third International Conference on Embedded Networked Sensor Systems (Sensys 2005)*, San Diego, CA, November 2005.

[18] United States Forest Service RAWS. http://www.fs.fed.us/raws/.

[19] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Second European Workshop on Wireless Sensor Networks (EWSN '05)*, January 2005.

[20] Wireless - Fresnel Zones and their Effect. http://www.zytrax.com/tech/wireless/fresnel.htm.

[21] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *First ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Los Angeles, CA, November 2003.

[22] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *First ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Los Angeles, CA, November 2003.

[23] G. Zhou, T. He, and J. Stankovic. Impact of radio irregularity on wireless sensor networks. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.