Design, Implementation, and Deployment of TempMesh: A Wireless Mesh Network to Aggregate River-Temperature Data

1st Scott Burman
Department of Land, Air and Water Resources
University of California
Davis, USA
sgburman@ucdavis.edu

2nd Jingya Gao
Department of Computer Science
University of California
Davis, USA
aligao@ucdavis.edu

3rd Dipak Ghosal
Department of Computer Science
University of California
Davis, USA
dghosal@ucdavis.edu

4th Greg Pasternack
Department of Land, Air and Water Resources
University of California
Davis, USA
gpast@ucdacvis.edu

5th Nann Fangue
Department of Wildlife, Fish, and Conservation Biology
University of California
Davis, USA
nafangue@ucdavis.edu

***Abstract*—As part of a restoration project designed to enhance Chinook salmon populations in the lower Yuba River, we sought to characterize river temperatures at micro-habitats scales relevant to juveniles. Given that temperatures vary more across the channel than they do longitudinally, temperature had to be assayed both along the length of a study reach as well as across the channel. These temperatures needed to be measured at least hourly in order to capture the totality of the diel fluctuations. It was necessary to sample for the full duration of juvenile residence in the lower Yuba River – a six-month period over the winter. Though it was desirable to design a system that could sample for at least a year to capture the totality of seasonal fluctuations. In this paper, we present the design, implementation and deployment of a wireless sensor network that was installed to monitor temperature data on the Lower Yuba River. The mesh network was implemented using nodes that contained a MSP430 chipset with a radio operating at 433 MHz. We describe the network architecture which included a network storage function to address intermittent link failures. We describe the event-based software architecture of the mesh network mode specifically in relation to energy optimization. We also give details of the deployment and provide measurement data.**

***Index Terms*—Monitor River-Temperature, Wireless Sensor Network, Battery Power Optimization, Network Storage, Timestamp Alignment**

I. Introduction

River managers and experts working on the Accord Management Team for the lower Yuba River in California's Central Valley sought to enhance Chinook salmon populations in 2014. They targeted the juvenile life-stage as particularly suitable for restoration efforts in the river. Recent work had determined that salmonids were adapted to the temperatures in their home rivers [1] [2]. There was hope that restoration work could be done to adapt the river system to better address the energetic needs of juvenile Chinook salmon, which have historically been abundant on the Yuba River [3].

Anadromous fish, like salmon, swim from oceans into rivers where they spawn. Their progeny hatch, grow and mature into smolts in these rivers, then move downstream and eventually out into the ocean. Salmonids expend considerable energy during river life-stages Prior to spawning, they must swim for miles against currents, generate gametes, and spawn. Then, the juveniles must hatch, grow in size many orders of magnitude, avoid predators, eat foods nearly as large as themselves, and finally smoltify. Yet, these river life-stages occupy only about a year of their total lifespan. Indeed, most of their time is spent in the ocean.

Anadromous fish must be adapted to the oceans where they live, which are relatively homogeneous in temperature. At the same time, they must retain (and pass on to their progeny) the ability to expend considerable energy and thrive in rivers with variable temperatures. In our study system, this is even more difficult as the fish must first traverse the relatively warm Feather River before entering the much colder Yuba River.

In addition to the evolutionary energetic impacts of river temperature – which might be suitably characterized by seasonal temperature data for a basin – we endeavored to place river temperature into an ecological context by assessing temperatures in micro-habitats. Micro-habitat river temperatures are complex and dynamic. River temperatures generally increase moving downstream from headwaters to mouth [4], and are related-to, but not forced by air-temperatures [5]. Yet, these trends vary seasonally and daily [6]. Solar warming is a strong predictor of river temperature [7], and as a result, vegetative shading can have intense local effects on micro-habitat temperature [8]. Finally, flows through sediments, which are shaded from the sun and are not subject to convective warming from air can stay relatively constant in temperature over vast distances [9]. Integrating these various inputs makes temperature predictions in rivers exceedingly complicated.

Determining river temperatures from models was found to be feasible, but computationally difficult, particularly at fine-scales, physics-based models were less effective than were interpolative models [10] [11]. Further, sensing methodologies like FLIR [12] and fiber-optic sensing [13] were not feasible in remote and treacherous environments found on may rivers as they require continuous maintenance and large power sources.

A. Goals and Objectives

In order to meaningfully sample the temperature experienced by juvenile salmonids, we needed to sample river temperatures at least once per hour (we opted for 4/hr) for a 3 km reach of the multi-threaded lower Yuba River. In order to capture the full juvenile life-stage, this sampling needed to span at least six months. Due to the regular winter flooding that coincides with the juvenile Chinook salmon residency and renders the river inaccessible, sensors needed to record and report data remotely. Data-logging temperature sensors were impractical for this reason. Monitoring needed to be highly-resolved both in space and time. Specifically, it was required that the temperature be monitored both longitudinally (along the river spaced a hundreds of meters apart) and across the channel (spaced meters apart).

B. Design Considerations

There were many factors that impacted the design of the sensors and the wireless mesh network. With respect to the sensors, one important consideration was their ability to survive diverse and changing flows and river conditions. To that end, sensors were installed in thick cables, which were tethered to anchors buried in the riverbed. While sensors were designed to withstand high flows and velocities, it was infeasible to design them to survive trees and other large debris carried by the river during floods.

Multiple environmental factors played in the design of both the wireless mesh network and the node level architecture. First, there were no reliable power sources on the river, where the sensor network was deployed. Consequently, the sensors and the wireless mesh network nodes needed to be battery-powered. Second, the geography and topography of the river and surrounding floodplains constrained the node placement. Third, the river reach was highly-trafficked by hikers, beavers, and bears, this required that nodes be small and well hidden. As a result of these environmental and human factors nodes were widely spaced and hard to access. First, due to larger distances between the network nodes, the link quality could become impaired particularly during heavy downpours and windy conditions. This required a network layer function to store data in intermediate nodes until network links were re-established. Second, once deployed, changing batteries at many of the nodes was a long and expensive operation. Hence, it was very important to ensure that the software architecture of the nodes were optimized to minimize power use.

II. Technology

A. Sensor Technology

We used off-the-shelf temperature sensors: Campbell Scientific CS225-L. These sensors were ideal as they are designed to be robust to field conditions (containing a steel wire-rope core and thick rubber jacket). These sensors are also versatile as they use SDI-12, each sensor is individually addressed, and are digital, so strings of sensors can be of any length without issue.

B. Wireless Mesh Network

We opted to connect these off-the-shelf sensors to wireless mesh network constructed of inexpensive, low-power devices with radios, capable of maintaining communications over the necessary ranges (100s of meters). We opted use 433 MHz, as it strikes a reasonable balance between penetration and distance, even at low power and with low-cost antennae. In applications such as ours, ranges of 500m are common. At this frequency, transmissions can penetrate vegetation, yet reflect well off of solid surfaces. In a river, the steep embankments leading up to floodplains and levees can create something of an echo chamber to enhance transmission ranges.

We chose to use the Wizzimote [14] – an IoT device built on top of TI CC430F5137 Microcontroller (based upon the MSP430, and with integrated 433MHz radio) – as the backbone of our WSN. As a System-on-Chip (SOC), CC430F5137 provides integrated peripherals for a variety of battery operated wireless applications. The operating modes take into account three different needs: ultra-low power, speed and data throughput, and minimization of individual peripheral current consumption [15]. MSP430-based SOCs are capable of multiple low-power modes (LPM0: 80uA, LPM2: 6.5uA, LPM3: 2uA, LPM4: 1uA), which preserve energy by shutting down respective clocks. MSP430s can later return to active mode (AM: RX: 15mA, TX: 30mA, depending on transmit strength) through enabled interrupts in less than 6us [15]. While transitioning between modes, the state of execution is saved on the stack and is restored; state and memory are also maintained for data preservation.

III. Protocol and Network Architecture

The network is comprised of three node types (Fig. 1):

1) **Sensor Nodes** *(orange)*: These units interfaced with a temperature sensor string, and were connected to the wireless mesh network. They were the sources of the temperature data. An Arduino periodically pulls, formats, and passes data to the Wizzimote through a serial interface. The Wizzimote transmits the packet to a relay node in the wireless mesh network. In the deployed network, there were as many as seven sensor nodes, each located at the edge of the river, with five sensors attached, spanning the width of the channel.
2) **Relay Nodes** *(blue)*: These nodes formed the backbone of the wireless mesh network. They received data packets from sensors, then passed them through upstream relay nodes. Packets were transmitted until they reached the gateway. They periodically received and transmitted packets, with the ability to buffer packets during periods when the forwarding channel was impaired.
3) **Gateway Node** *(maroon)*: This node was the aggregation point. The data, received from one or more relays, was sent from the Wizzimote to a Raspberry Pi over a serial interface. The Raspberry Pi timestamped the data

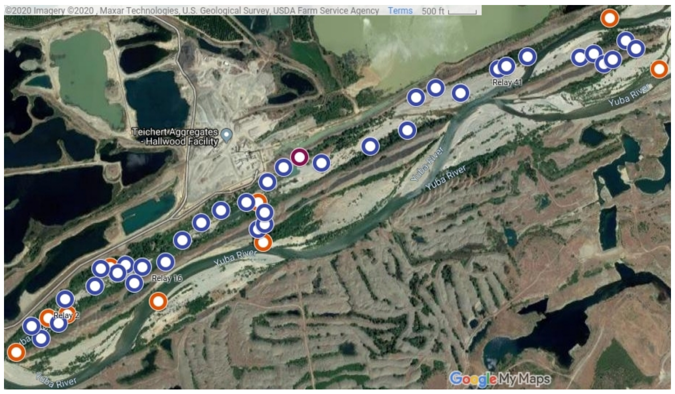
©2020 Imagery ©2020 , Maxar Technologies, U.S. Geological Survey, USDA Farm Service Agency Terms 500 ft
Teichert Aggregates Hallwood Facility
Relay 41
Relay 16
Relay 2
Yuba River
Google My Maps

Fig. 1. Map of the network deployed in November 2018 through May 2019 on the lower Yuba River. The orange nodes are the sensor nodes, the blue nodes are the relay nodes, and the maroon node is the gateway node. This map was generated using Google My Maps.

packets, stored them on a memory card, and uploaded the data to an off-site server using a cellular connection. The gateway was solar powered.

A. Medium Access Control

Synchronization between nodes was achieved using an advertisement-based approach as in [16]. To ensure that two nodes could synchronize when needed, we set each node's sleeping period to be 900 ticks (1 tick was roughly 1 ms) and the advertisement period to be 1000 ticks. This ensured that the receiving node woke up during the transmitting node's advertisement. During the 1000 ticks, the transmit node continuously sent packets containing the remaining time of advertisement (in ms). The data were transmitted immediately after the advertisement period. The transmitting node had no knowledge of whether or not a receiver heard the advertisement. The transmitter reattempted this process (in between sleeping) until the receiving node acknowledged receiving the data packet.

The use of advertisements to synchronize nodes significantly increased the chance of collision at merge points, where multiple nodes transmitted to one receiving node. As failed transmissions were repeated, collisions increased the transit time for a packet to reach the gateway node from a sensor. To avoid collisions, we implemented cross-listening – listening on the transmitting channel for other nodes before advertising. This is similar to the Carrier Sense Multiple Access (CSMA) used in IEEE 802.11 protocols.

B. Routing and Channel Assignment

To save energy and simplify the installation, the network used static routing. As the network topology was uncertain when designed, developing the system with static routing provided better control over the exact traversal path of the packets. Static routing was achieved by assigning fixed channels (frequency bands) between nodes. As there were only a limited number of channels, channels were reused in links that were far apart.

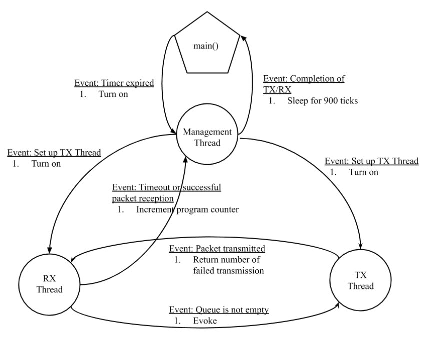
main()
Event: Timer expired
1. Turn on
Event: Completion of TX/RX
1. Sleep for 900 ticks
Management Thread
Event: Set up TX Thread
1. Turn on
Event: Set up TX Thread
1. Turn on
Event: Timeout or successful packet reception
1. Increment program counter
Event: Packet transmitted
1. Return number of failed transmission
RX Thread
TX Thread
Event: Queue is not empty
1. Evoke

Fig. 2. Finite State Machine (FSM) showing the overall organization of the pseudo-threads and the events that trigger transitions to the different threads.

C. Network Storage

Through testing in the field, we found that weather, distance, position, direction, obstacles, etc, can all intermittently impair the link quality. In order to mitigate these impediments, we implemented a module in the nodes that minimized repeated re-transmissions while preserving a maximum number of packets. This was achieved by exponentially backing-off of the transmitter in times of unsuccessful transmission. In particular, the base transmission interval of 4 seconds, on unsuccessful transmissions, was binary exponentially increased to 8 seconds, 16 seconds, 32 seconds, and so on. While the node slowed its transmissions, received packets were stored until its queue filled, thereby storing as many packets as possible in the network. Once the link quality was restored (detected by nodes via a successfully acknowledged transmission), the transmission interval was reset to 4 seconds. The details of this algorithm are beyond the scope of the present paper.

IV. Node Architecture

The Wizzimote library implemented Protothreads, a system of stack-less threads that provided a blocking context on top of an event-driven system [17]. Using macros to save the relative processing state of each function, Protothreads provided sequential flow of control without using complex state machines or full multi-threading. The Wizzimote radio library contained one radio buffer that was shared between transmitting and receiving. This affected our decision to have threads scheduled sequentially (transmit after receive), so as to prevent a conflict over radio resources.

A. Thread Architecture

In each relay node, there were three threads: the management thread, the transmit (TX) thread, and the receive (RX) thread (Figure 2).

1) Management Thread: The management thread initialized both transmit and receive threads. The main() and management thread coexist because all KAL processes must exit before the node could transition into sleep mode, which was achieved by killing the management thread. The main() function also controlled two timers. The main timer put the node to intermittent sleep (See "Energy Efficiency" below) and the built-in watchdog timer of MSP430, which detected inactivity and performed a hardware restart if necessary [15].

2) RX Thread: The RX thread was started by the management thread, and listened for incoming packets. The receive thread was instantiated before the transmit thread in order to maximize queue usage. Even when the forward channel was broken, the node continued to receive until the queue was full.

3) TX Thread: The TX thread was called after the RX thread finished its cycle. When the queue was non-empty,the TX thread advertised and transmitted (backing off exponentially when transmission were unsuccessful). The end of TX thread returned to the management thread, which triggered the next sleep, receive, transmit cycle.

B. Energy Efficiency

In order maximize battery-life and minimize in-field maintenance, nodes were put to sleep when not transmitting or receiving. Each relay node was pulled to low power mode 3 (LPM3), which disabled the CPU (MCLK) and the high frequency clock (SMCLK), leaving only the 32kHz low frequency crystal clock ACLK active [15]. A timer of 900 ticks was initialized in main() to interrupt and return to active mode. After closing the radio layer and the serial, the node was put into low power mode 3 (LPM3) and global interrupts were enabled. The node woke up when the main timer timed out. This process was repeated after every cycle of receive and transmit.

V. Deployment and Evaluation

A. Power Use

In order to determine the amount of power used by relay nodes, we conducted an experiment that logged their current draw. We did this by setting up three nodes: a dummy-sensor node that generated numerous packets at regular intervals, a relay which transmitted the data, and a gateway node that received the data from the relay. We setup the dummy-sensor node to transmit at a regular interval of 4, 8, 16, or 32 seconds. The relay node was connected to a 3.600 VDC power source, through a logging bench-top multimeter capable of 500 samples per second, with 200,000 sample stoage (yielding 6 min 40 sec of continuous measurement). We measured voltages both above and below this logging meter to ensure that burden voltage was not too large, and that supply voltage to the relay node stayed above 3.500 VDC. These voltage/burden monitoring multimeters were put into their high-speed max/min modes, which measured at least once every 1ms.

Each of the relay node's states were reflected consistently in the data [Figs. 3]. Low-power sleep states used 2-3 uA, active mode use about 5 mA, and receive and transmit modes used 20-30 mA [Figs. 3]. These data were consistent with the power-use metrics found in [15].

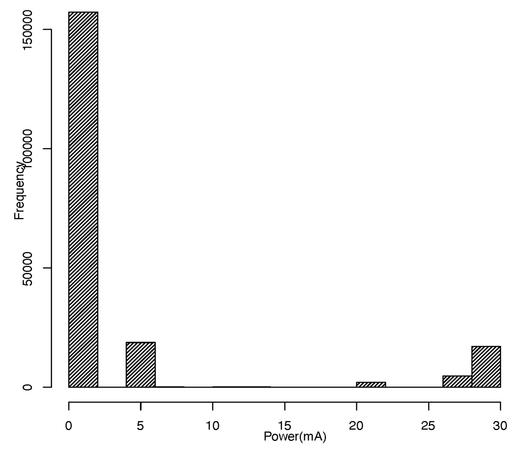

Fig. 3. Histogram of the power used in mA when relay node receives a data packet every eight seconds. Each of the peaks in this histogram corresponds to a state for the Wizzimote. The peak around 0 mA corresponds to the low-power sleep state, 5 mA corresponds to active mode, and the peaks between 20 and 30 mA correspond to transmit and receive modes.

B. Network Storage

To test the efficacy of the network storage algorithm, we recreated a simple network that simulated link instability and observed the network recovery and loss. This amounted to a linear network, in which a sensor node connected to a gateway node through two relay nodes. The sensor node transmitted data every 30 seconds and the relay nodes had a packet queue of size 10 packets. The link between the second relay and the gateway was shut down for 10, 100, 600, and 1000 seconds, each of these shutdowns was followed by 60 minutes of potential recovery time. The network recovery time was defined as the duration from the link failure to the point where the gateway received its first packet after the network recovered. There was an idle time period, during which the network link had recovered, but the transmitting node had not yet reattempted its transmissions. We consider the link to be dysfunctional during this idle time period because the network health was dependent on the overall traversal of packets in the network rather than the individual link health.

Figure 4 shows that as the link remains down for longer periods of time, the delays in between transmissions increased exponentially, and so did the recovery times of each node. The size of queue also influenced the back-off rate of the network. When the queues of connected relay filled, the overall recovery time of the network increased as a accumulation of multiple relay back-offs.

The rate of packet creation was significantly greater than in a realistic deployment. This was done to ensure that packet queues saturated while the link was broken, which in turn allowed us to test the back-propagation of packet storage. Once the queues of all available relays filled, the packet loss increased in proportion to the link down time. In the field implementation, the packet frequency was much smaller, and the number of relays was much larger. These two factors granted the network in the field deployment more storage, over

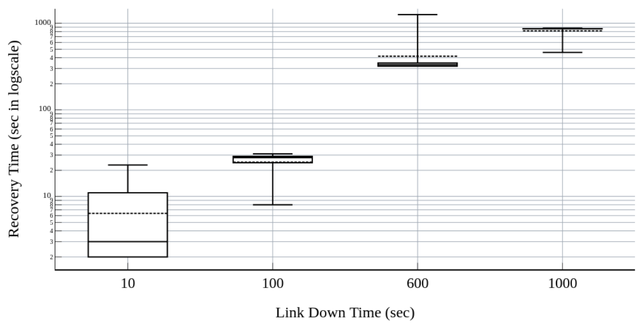

Fig. 4. Network recovery time in seconds (log scale) as a function of link failure duration. The mean recovery times are shown as the dashed line and are respectively 6.36 s, 24.83 s, 415.36 s, and 815.75 s.The median recovery times are respectively 3 s, 28 s, 331 s, 865.5 s.

a longer period of time, which minimized the probability of packet loss.

C. *Timestamp Alignment*

Data were timestamped when they were received by the gateway node. As a result, the timestamps were recorded after any network-latency or delays that resulted from low-network fidelity. When network fidelity was good, the latency of the network yielded delays on the order of seconds to minutes, which were below the scale of fluctuations of river temperatures. But, when network fidelity was lower, or when data were stored in the network for extended periods of time due to node-failure, these delays were substantial. As such, we needed to detect and align incorrectly-time-stamped data packets. To do this, we first needed to find bad packets. Data packets were generated every 15 min (+/- 1 min due to temperature and crystal variation). For each packet of from a sensor, we found the 10 packets that arrived before and after that packet. Using the timestamps we computed an expected timestamp for the packet in question, then averaged these 20 estimates. If the recorded timestamp was within 10 minutes of that estimate, we kept the timestamp, else we marked it as bad. This yielded 29,168 packets which were properly timestamped, and 16,542 that were improperly timestamped. Once identified, we sought to align the timestamps of these packets. To this end, we created a window around the bad packet, and expanded the window (from 10 to 48 hours) until it contained 15 properly timestamped packets. We then used these neighboring packets to estimate the timestamp for the improperly stamped packet. 13,728 of the improperly timestamped packets had at least 15 neighboring properly timestamped packets in this window with which to estimate the correct timestamp. We assessed the quality of these estimates by considering their range. We kept those estimates which had a range of less than 30 minutes. Finally, we took the mean of all estimates and saved it as a new timestamp. Using this methodology, we aligned the timestamps of 6,395 incorrectly timestamped packets. The remaining either did not have 15 good timestamps within 48 hours, or the estimates did not fall into the 30 minute range.

D. *Temperature Data*

This network persisted from November 2018 through May 2019 – when restoration work on the river rendered the network impossible to maintain. While this period was shorter than intended, it did cover the majority of the juvenile life-stage of the Chinook salmon present in the river. The network was particularly fruitful as it continued to upload data during flood events (Figure 5: vertical red bars). Few (if any) previous studies have recorded temperatures of large rivers during major flood events. This is because flood-events are powerful, and tend to damage, bury, and dislodge installed data-logging temperature sensors, rendering them (and their data) unrecoverable. While many of our sensors were damaged during these large floods, because data were transmitted in near-real-time, we were able to recover data recorded prior to their destruction. As we continue to analyze these data, we anticipate that fluvial temperatures homogenize during flood events, but hope to find more complex and nuanced dynamics at play during major floods.

VI. Related Work

Wireless sensor networks (WSNs) have been used in monitoring applications in similar harsh environments such as early detection of forest fires [18], high resolution spatio-temporal monitoring in underwater environments [19], and tracking applications such as animal telemetry [20]. Techniques to optimize energy in wireless sensor networks has been extensively studied [21]. In [22] a Sparse Topology and Energy Management (STEM) algorithm was proposed to efficiently wake up nodes from a deep sleep state without the need for an ultra low-power radio. It allowed tradeoff between energy efficiency and the latency that is incurred to wake up the node. The tradeoff between data fidelity and energy efficiency has been investigated in [23]. There has also been a number of studies on energy efficient routing schemes [24]. In our work we have used an advertisement based node synchronization method [16]. An alternative approach is a randomized algorithm based on the birthday paradox which has been proposed for wireless sensor networks [25]. A comparative analysis of the two approaches (advertisement-based and randomized) is beyond the scope of this paper.

VII. Conclusions

In this paper we have presented a detailed design, implementation, and deployment of a wireless mesh network to collect river-temperature data at a fine spatial and temporal scale (Figure 5), which has not been done before. The overarching goal is use the temperature data along with other data including anadromous fish habitat and other flow features of the river to develop models and study the impact of river-temperature profile on the survival and growth of juvenile anadromous fish species. In this practice paper, we discussed the challenges in deploying the mesh network and discussed how network

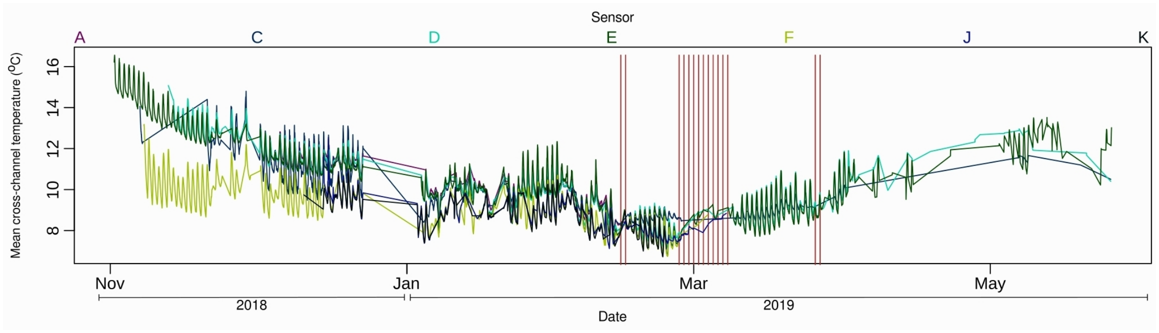

Fig. 5. Temperature data shown above are cross-channel means: they are the average of the five individual sensors attached to each sensor-node. The vertical red lines are at midnight on days in which there was a flood of over twice (10,000 cfs) the bankfull flow (5,000 cfs).

storage was implemented to address intermittent link failures during inclement weather. We also discussed the design of the relay node and how a pseudo-thread architecture along with low power mode was leveraged to optimize energy usage.

VIII. Acknowledgements

This work was supported by a cooperative agreement with the Anadromous Fisheries Restoration Program of the US Fish and Wildlife Service. We wish to express our gratitude to all who contributed to the success of this project. Dr. Beth Campbell and Paul Cadrett from USFW championed the project. Loren Stearman, Duane Massa, and John Cleveland of the LYR/PSMFC crew provided field support. Chelsea Martinez led the development of field protocols and was instrumental in their execution. Richard Hambrick began the software development process. Darcy Bostic, Annelise Del Rio, Arielle Gervais, Sean Luis, Vanessa Lo, Seanna McLaughlin, Brianna Ordung, Lea Pollack, Kathy Russel, Rebecca Walker, Jason Wiener, Katie Woodworth, and Ken Zillig volunteered their time in the field.

References

[1] Hague MJ Hanson LM Gallagher ZS Jeffries KM Gale MK Patterson DA Hinch SG Farrell AP Eliason EJ, Clark TD. Differences in thermal tolerance among sockeye salmon populations. *Science*, 332:109–112, 2011.

[2] Verhille CE Cocherell DE English KK Farrell AP, Fangue NA. Thermal performance of wild juvenile oncorhynchus mykiss in the lower tuolumne river: A case for local adjustment to high river temperature. Technical report, Don Pedro Project, 2015.

[3] Wells HL Chamberlain WH. History of yuba county, california, with illustrations descriptive of its scenery, residences, public buildings, fine blocks and manufactories. Technical report, Thompson West, 1879.

[4] Ward JV. *Perspectives in Southern Hemisphere Limnology*, chapter Thermal characteristics of running waters, pages 31–46. Dordrecht: Springer Netherlands, 1985.

[5] Nobilis F Webb BW. Long-term perspective on the nature of the water temperature relationship: a case study. *Hydrologic Processes*, 11:137–147, 1997.

[6] Smith K. Water temperature variations within a major river stream. *Hydrologic Processes*, 6:155–169, 1975.

[7] Parrett KB Duke SD Danehy RJ, Colson CG. Patterns and sources of thermal heterogeneity in small mountain streams within a forested setting. *Freshwater Biology*, 208:287–302, 2005.

[8] Meehan WR Bjornn TC Hetrick NJ, Brusven MA. Changes in solar input, water temperature, periphyton accumulation, and allochthonous input and storage after canopy removal along two small salmon streams in southeast alaska. *Transactions of the American Fisheries Society*, 127:859–875, 1998.

[9] Caissie D. The thermal regime of rivers: a review. *Freshwater Biology*, 51:1389–1406, 2001.

[10] PA Conrads and EA Roehle. Comparing physics-based and neural network models for simulating salinity, temperature and dissolved in a complex, tidally affected river basin. In *South Carolina Environmental Conference, Myrtle Beach, Carolina, USA*, 1999.

[11] Satish Mysore G Caissie D, El-Jabi N. Modelling of maximum daily water temperatures in a small stream using air temperatures. *Journal of Hydrology*, 251:14–28, 2001.

[12] Packman AI Scott DT Cardenas MB, Harvey JW. Ground-based thermography of fluvial systems at low and high discharge reveals potential complex thermal heterogeneity driven by flow variation and bioroughness. *Hydrologic Processes*, 22:980–986, 2008.

[13] John S Selker, Luc Thevenaz, Hendrik Huwald, Alfred Mallet, Wim Luxemburg, Nick van de Giesen, Martin Stejskal, Josef Zeman, Martijn Westhoff, and Marc B Parlange. Distributed fiber-optic temperature sensing for hydrologic systems. *Water Resources Research*, 42(W12202), 2006.

[14] Yordan Tabakov. Wizzilab:connecting things/wizzimote. http://wizzilab.com/wiki/#!hardware/wizzimote.md#WizziMote.

[15] Texas Instruments. Cc430 family user's guide (rev. e). Technical report, Texas Instruments, 2013.

[16] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, 2006.

[17] Thiemo Voigt Adam Dunkels, Oliver Schmidt and Muneeb Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006), Boulder, Colorado, USA*, 2006.

[18] Mohamed Hefeeda and Majid Bagheri. Forest fire modeling and early detection using wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 7(3-4):169–224, 2009.

[19] J. Wang, D. Li, M. Zhou, and D. Ghosal. Data collection with multiple mobile actors in underwater sensor networks. In *2008 The 28th International Conference on Distributed Computing Systems Workshops*, pages 216–221, June 2008.

[20] Roland Kays, Margaret C Crofoot, Walter Jetz, and Martin Wikelski. Terrestrial animal tracking as an eye on life and planet. *Science*, 348(6240):aaa2478, 2015.

[21] Holger Karl and Andreas Willig. *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.

[22] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE transactions on mobile computing*, 1(1):70–80, 2002.

[23] Athanassios Boulis, Saurabh Ganeriwal, and Mani B Srivastava. Aggregation in sensor networks: an energy–accuracy trade-off. *Ad hoc networks*, 1(2-3):317–331, 2003.

[24] Arvind Kumar et al. *Energy Efficient Clustering Algorithm for Wireless Sensor Network*. PhD thesis, Lovely Professional University, 2017.

[25] Michael J McGlynn and Steven A Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 137–145. ACM, 2001.