Computer Communications 118 (2018) 217–233

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Solar-powered, wireless smart camera network: An IoT solution for outdoor video monitoring

Kevin Abas*, Katia Obraczka, Leland Miller

Engineering 2, Room 323, University of California Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, United States

ARTICLE INFO

Keywords:
Wireless camera networks
IoT
Outdoor video monitoring
Solar harvesting
Energy efficiency

MSC:
00-01
99-00

ABSTRACT

In this paper, we present SlugCam, a solar-powered, wireless smart camera network that can be used in a variety of outdoor applications including video surveillance of public spaces, habitat and environmental monitoring, wildfire prevention and detection, to name a few. SlugCam was designed such that it can be deployed and left unattended for extended periods without requiring regular maintenance, e.g., frequent battery replacement. The system is built with off-the-shelf components which not only keeps it modular and low cost, but also facilitates its prototyping, rapid duplication, and evolution. SlugCam's on-board processing capability allows computer vision software to run locally and autonomously. Energy efficiency in SlugCam is accomplished both in: (1) hardware by micro-managing low-power components; as well as in (2) software by having the system's operation duty cycles automatically adapt to the current state of the battery in order to balance the trade-off between application-level requirements and power awareness. For example, SlugCam's smart camera node changes its monitoring behavior based on how much battery charge remains. Additionally, using its computer vision software, the system only records and transmits information upon event detection which contributes both to the system's energy efficiency as well as its low network bandwidth requirements. SlugCam's networking functionality enables camera nodes to transfer video files, as well as collaborate on tasks such as visual processing, event detection, and object tracking. It allows node-to-node, as well as scoped- or full broadcast communication. For point-to-point communication, SlugCam uses on-demand power-aware multi-path routing to transfer video files efficiently. Another important contribution of SlugCam is to provide an open-source wireless camera network that can adapt to address the requirements of future outdoor video monitoring applications. SlugCam also includes a Web-based server where video data is stored as well as a Web-based user interface that allows end users to interact with the system, tag, query and retrieve video files, and manage SlugCam nodes remotely. In addition to a detailed description of SlugCam, this paper presents an extensive power characterization of the system's operation and showcases its deployment in a lab testbed and a real world scenario.

1. Introduction

Several recent technological trends have enabled the Internet of Things (IoT), including the availability of small form factor, low-cost, low-power devices with on-board computing, communication, and sensing capabilities, as well as the availability of ubiquitous wireless communication. IoT applications are extremely diverse and include medical and health care, home, office, and industrial automation, environmental and habitat monitoring, transportation, as well as the so-called environments such as smart cities, smart grids, etc.

In our work, we focus on wireless camera networks as IoT solutions for video-based monitoring and surveillance of outdoor spaces. Wireless camera networks find a variety of applications including video surveillance of public spaces, habitat and environmental monitoring, wildfire prevention and detection, to name a few.

Wireless camera networks exhibit unique characteristics and have to address the particular needs of their applications [1]. For example, cameras for outdoor deployment have power efficiency as one of their main design requirements. Besides utilizing efficient hardware, wireless camera networks must also address the trade-off between satisfying performance requirements imposed by their applications while maximizing their lifetime [2]. For instance, camera network applications have far higher storage and bandwidth requirements when compared to networks of scalar sensors (e.g., temperature, humidity, etc). Furthermore, having cameras "stay on" and record live footage in outdoor surveillance scenarios is prohibitively expensive and may raise serious privacy concerns. Additionally, it is critical that only relevant data gets stored and/or transmitted to the end user to avoid (1) consuming

* Corresponding author.
E-mail addresses: kabas@soe.ucsc.edu (K. Abas), katia@soe.ucsc.edu (K. Obraczka), leland.miller@ucsc.edu (L. Miller).

https://doi.org/10.1016/j.comcom.2018.01.007
Received 1 May 2017; Received in revised form 24 December 2017; Accepted 14 January 2018
Available online 31 January 2018

unnecessary network and energy resources and (2) overwhelming the end user with large amounts of information, most of which is likely to be irrelevant.

Leveraging the availability of nodes with higher resolution cameras and on-board processing and storage capabilities, so-called "smart" camera networks employ computer vision algorithms to filter relevant data to be locally stored and/or transmitted to the end user. On-board video processing also yields significant network bandwidth and energy savings by only transmitting relevant data.

Energy efficiency and independence is of course a critical concern in wireless camera networks for outdoor applications. The increasing availability and decreasing cost of solar-powered solutions have been transforming the wireless sensor networking landscape. It has been enabling a wider range of visual sensing applications, especially in remote areas where access to the power grid is either non existent or too costly [3]. Solar-powered systems have also become an attractive alternative to traditional battery operated systems which typically require frequent maintenance and battery replacement.

In this paper, we introduce SlugCam [1], a solar-powered, wireless smart camera network that can be used in a variety of outdoor applications. SlugCam's main contributions can be summarized as follows:

- SlugCam uses an "open system" approach and is built with off-the-shelf hardware components which not only keeps it modular and low cost, but also facilitates its prototyping, rapid duplication, and evolution.
- Consistent with its open system philosophy, SlugCam's open source, modular software facilitates evolution and adaptability to requirements of emerging visual sensor network applications.
- SlugCam includes a Web-based user interface and Web server. The Web server stores video files captured and transmitted by SlugCam nodes. The user interface allows users to query the server for relevant video files. Through the user interface, users can tag events in the corresponding video. It also allows users to specify to SlugCam nodes behaviors to be monitored and detected.
- SlugCam's energy efficiency is achieved both by fine-grained management of low-power hardware components, as well as by having the system's operation duty cycles automatically adapt to the current state of the battery in order to balance the trade-o between application-level requirements and power efficiency.
- SlugCam's on-board processing capability allows computer vision algorithms to run locally contributing to the system's autonomous operation, as well as energy- and bandwidth efficiency.
- SlugCam's power-aware networking functionality enables efficient point-to-point and multi-point data communication.

The remainder of this paper is organized as follows. In Section 7 we discuss related work in the field. Section 5 provides an overview of SlugCam, including a discussion on our routing protocol choice, while SlugCam's design and architecture are described in Section 4. Section 5 presents experimental results obtained from deploying SlugCam in SlugCam, our wireless visual sensor network testbed. Finally, in Section 6, we provide a brief description of the SlugCam wireless smart camera node used in our testbed and Section 7 concludes the paper and discusses directions for future work.

2. SlugCam overview

SlugCam has been designed with readily available, energy efficient off-the-shelf components to reduce cost and allow for rapid development. Each device has been chosen carefully to meet our power consumption requirements, as detailed in Section 5. Unlike existing smart cameras that run solely on battery as their energy source, we have designed SlugCam to run efficiently on solar power while leveraging a rechargeable battery when sunlight is not available. Another important design consideration was to ensure sufficient on-board processing capabilities which allows SlugCam to perform video processing tasks locally. SlugCam's ability to run computer vision algorithms on-board, enables it to be more selective of the video it records and transmits, which in turn contributes to power efficiency and avoids overwhelming the end user with information that is not relevant.

SlugCam's on-board camera is normally off and gets turned on by a passive infrared (PIR) sensor when motion is detected. Additionally, SlugCam is able to adapt its operation to the available energy remaining in its rechargeable battery. By combining low-cost and low-power hardware design with energy-aware duty cycle operation, SlugCam adequately balances energy efficiency and application-level requirements, an essential feature in wireless smart cameras [2]. This is showcased in our thorough power consumption characterization of SlugCam described in Section 6.

Developing an open-source platform that could be used and adapted to other applications was also an important design consideration. As a result, besides being a low-cost, "open-hardware" wireless camera network, SlugCam also includes an open-source web-based management software suite that allows SlugCam's wireless nodes to me managed remotely, including the ability to view recorded video data as described in detail in Section 4.

Below, we summarize SlugCam's main design goals:

- *Energy efficiency:* One of SlugCam's distinguishing features is that it includes two processing units: an MSP430 and a Raspberry Pi. The low-power MSP430 micro-controller [4] manages the amount of time that the more power-consuming Raspberry Pi stays on. This is accomplished by having the MSP430 control PIR motion sensing while the Pi controls the rest of the system. When no motion is detected, only the MSP430 is "on" and all other components of the system, including the Pi, are "off". As will be shown by our power characterization experiments in Section 6, this "smart trip-wiring" approach yields substantial power savings, which allows SlugCam to use a smaller size solar panel and battery, and yet run processing-intensive computer vision algorithms using local system resources. Furthermore, SlugCam nodes operate under power-aware duty cycles which allow them to adjust their operation according to current battery levels.
- *Outdoor, wireless, off-grid operation:* As previously mentioned, one of SlugCam's main goals is the ability to not rely on the power grid or wired communication infrastructure, and operate autonomously and unattended for extended periods of time. Equipped with solar-powered batteries and directional WiFi antennas, SlugCam is completely wireless and thus can be deployed in remote, hard to access areas where access to the power- or wired communication infrastructure is either not viable or too costly.
- *Flexibility and open-source by design:* One of our main goals in the SlugCam project is to provide a platform that other researchers and the community at large can use and extend. As such, our system is both flexible and easily reproducible. As previously pointed out, the current SlugCam node is equipped with two sensors, namely the camera and the PIR, but can be extended with other sensors in order to cater to different applications and their requirements. Another one of our main design decisions was choosing to keep both hardware and software designs as open as possible. Not only do we make our code available on a public on-line repository, but we have also used hardware and software that are both open source and very friendly to modification.

3. SlugCam hardware

SlugCam's node hardware architecture is illustrated in Fig. 1. At the core of SlugCam's node is the Raspberry Pi (or sometimes referred to as

[1] An earlier version of this work was reported in [1].

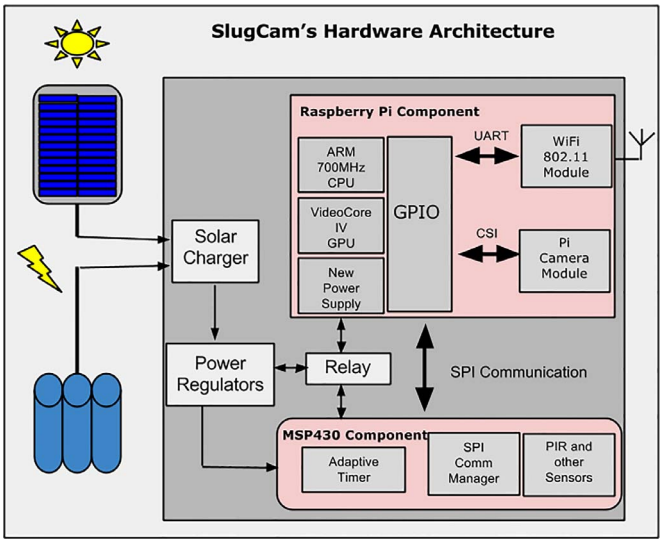

Fig. 1. SlugCam node's hardware architecture.

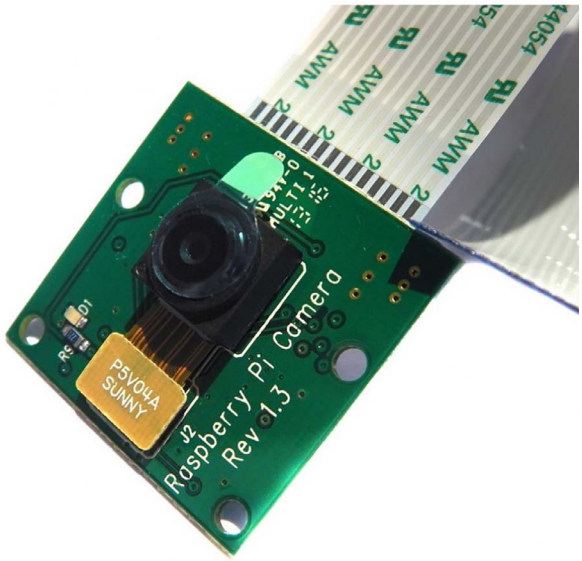

Fig. 2. Raspberry Pi compatible camera module.

"the Pi"), an open-hardware device [1]. SlugCam also includes a very low-power micro-controller module based on the MSP430 to further improve energy efficiency.

3.1. MSP430 Module

Its ability to run on 0.05mA on standby and nearly 5 times less current when sleeping makes the MSP430 quite attractive for systems where power-efficiency is of critical importance. In the case of SlugCam, by having the MSP430 control how long the Raspberry Pi module stays on, it is able to decrease SlugCam's overall power consumption substantially. The MSP430 accomplishes this using an on-board timer and an external mechanical relay acting as a switch. Before cutting power, the relay waits for a signal that the "Pi" has successfully shut down. To minimize time drifts due to large temperature variations common in outdoor deployments, a 32 Khz crystal clock is used. The Raspberry Pi can dynamically change how much time it stays on by communicating to the MSP430 over a Serial Peripheral Interface (SPI) connection. It can also adjust the MSP430's sensitivity to movement detection to minimize false positive events which negatively impact SlugCam's application-level performance and energy efficiency. For example, if a wild animal is causing the alarm, the MSP430 can start a timer that can vary in length for when it should again be sensitive to movement.

3.2. Raspberry Pi camera module

We used the Raspberry Pi camera module (shown in Fig. 2), which is capable of capturing 1920 × 1080 resolution color images with 30 fps maximum rate. It uses the Pis on board CSI Camera port which substantially reduces the system's overall power consumption, for example, when compared to a USB Webcam by not drawing any extra current from the USB hub.

3.3. WiFly module

Due to bandwidth and communication range requirements as well as video quality requirements, we decided to use WiFi as SlugCam's wireless communication technology. Zigbee (or IEEE 802.15.4) was also considered but its power efficiency versus performance trade-offs were not adequate to SlugCam's requirements. Additionally, manufacturers of WiFi embedded modules have been making large strides in reducing their power consumption. For example, we are currently using the RN-174 WiFly module from Microchip which consumes only 4μA in sleep state.

3.4. Daughter card and enclosure

SlugCam includes a custom PCB daughter-card to house its external devices and available I/O ports. The daughter-card plugs directly into the Raspberry Pi and improves robustness and reliability of the system's hardware; it also reduces SlugCam's form factor. Future revisions of SlugCams daughter-card may include additional sensors to address the needs of a wider range of outdoor monitoring applications. Fig. 7 shows SlugCam's weather-proof enclosure; note that the solar panel is detached to allow for optimal sunlight exposure.

3.5. Power module

Solar panel dimensioning

Unlike traditional battery powered smart cameras, SlugCam will rarely require battery replacement as it uses solar harvesting techniques and rechargeable batteries. When sizing SlugCam's solar panel and battery for our power requirements, one of our main goals was to keep both cost and form factor low. More specifically, for our planned SlugCam deployment, we used up solar radiation data for Santa Cruz, CA available from the National Renewable Energy Laboratory (NREL) Web site [5]. NREL's Web page shows monthly peak, average, and minimum levels of solar radiation at different orientation of PV cells relative to the sun. Since one of SlugCam's main design goals is unattended operation for long periods of time, we conducted worst case analysis; in other words, our solar panel dimensioning analysis ensures that even with extended periods of little sunlight, SlugCam could still operate, e.g., providing minimal service levels to its applications. Knowing that the worst case sun exposure is in the winter months, we chose the month of January with a PV cell orientation facing south as the basis for our analysis.

Solar panel dimensions account for the fact that the energy produced by a photo-voltaic (PV) cell, or its energy output, is directly proportional to the cell's "active area", the amount of light falling in that area, and the conversion efficiency of the cell, as described by the following expression.

$$\text{EnergyOutput} = \text{SolarEnergy} \times \text{PanelareaPV} \times \text{Efficiency}$$

Current PV cell technology differs in efficiency, construction material, and cost. Today the most known PV cells materials are crystalline silicon (e.g., mono-crystalline silicon and poly-crystalline silicon) and

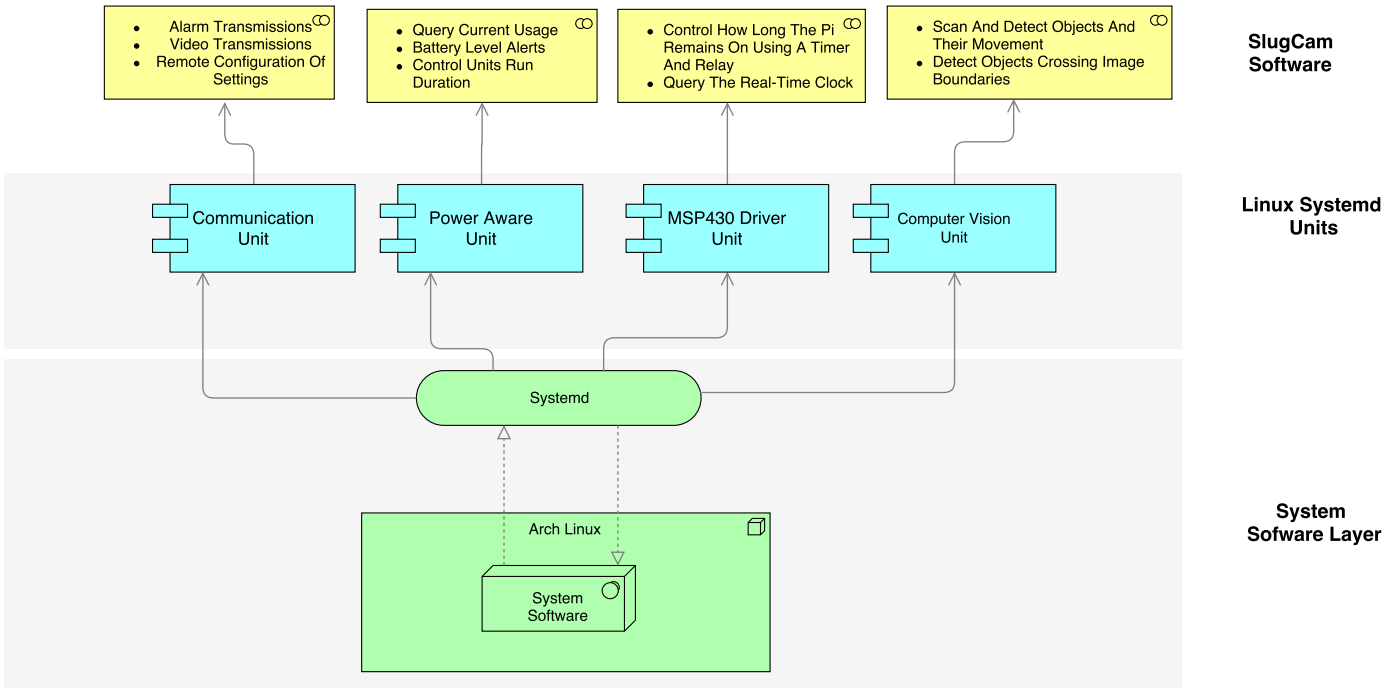

Fig. 3. Raspberry Pi manager's software architecture.

thin film (e.g., amorphous silicon, cadmium telluride, and copper indium gallium selenide) [6]. We found a reasonably priced mono-crystalline solar panel with 6V and 19% of efficiency. According to the data from our solar radiation research, Santa Cruz receives 2–3 kWh per day in the month of January. This represents the amount of energy accumulated over 1 m^2 in a 24 h period. As such, we picked a solar panel of size 220 mm by 175 mm which yields enough energy to power a SlugCam node for a whole day in Santa Cruz, our target deployment area.

Battery choice

Since PV cells can only absorb solar energy when there is enough sunlight exposure, batteries are needed to power the system when there is little to no sun. Lithium-ion (Li-ion) batteries are currently one of the most promising battery technologies: it is used for portable consumer products as well as for electric vehicle power-trains. Its main drawbacks are that they are usually more expensive than other types of batteries (e.g., Nickel and lead acid systems) and need protection circuits for safety.

A critical factor when choosing the battery is its capacity, i.e., how much energy it can store in ampere-hours (Ah). To estimate SlugCam's we use the expression below which is based on: (1) how much time the deployment location receives no solar radiation and (2) SlugCam's power consumption.

$$\text{BatteryCapacity} = \text{Consumption(Amps)} \times \text{NightUsage(Hrs)}$$

Considering that the Raspberry Pi consumes around 300–400 mA and night time is on average 8 h per day, we estimate that the Pi's nightly current per hour consumption is approximately 3200 mAh. We chose a 3.7 V 17600 mAh pack of Li-Ion batteries which will allow the system to last up to 4 days without sun at maximum current consumption levels.

In order to charge the battery adequately and safely, we use Microchip's MCP73871 charger [7]. It allows the SlugCam node to both charge and make use of the extra energy available during full sunlight exposure. The charger has other useful features such as temperature monitoring for the battery which prevents charging the battery under extreme weather conditions. It also provides status information for current battery conditions (e.g., when the battery has finished charging or when the battery has very low charge levels).

Fig. 4. SlugCam's background subtraction: occlusion detection.

On-board current sensing

The Pi also records current consumption which allows SlugCam to adapt its operation to available battery charge levels. Knowing its current usage and available battery level, the system can estimate how much time is available before it needs to enter a sleep state waiting for available solar power. The current sensor [8] also allows us to determine what generic power consuming state the system is in. The idling state is when both microprocessors are on and idling, but all auxiliary sensors and the WiFi module have been disabled or put to sleep. When we discuss SlugCam being in a high powered state, the system is using both the WiFi module and auxiliary sensors like the camera module actively. The low powered sleep state, refers to when only the PIR sensor is idling and the Pi is powered off, and the MSP430 has been put into its own sleep state waiting for a hardware interrupt from the PIR sensor to wake SlugCam up.

4. SlugCam software

SlugCam's main software components include the Raspberry Pi manager, the MSP430 firmware, the Web-based video server, and the Web-based user interface. Fig. 5 shows a three-node mesh SlugCam deployment.

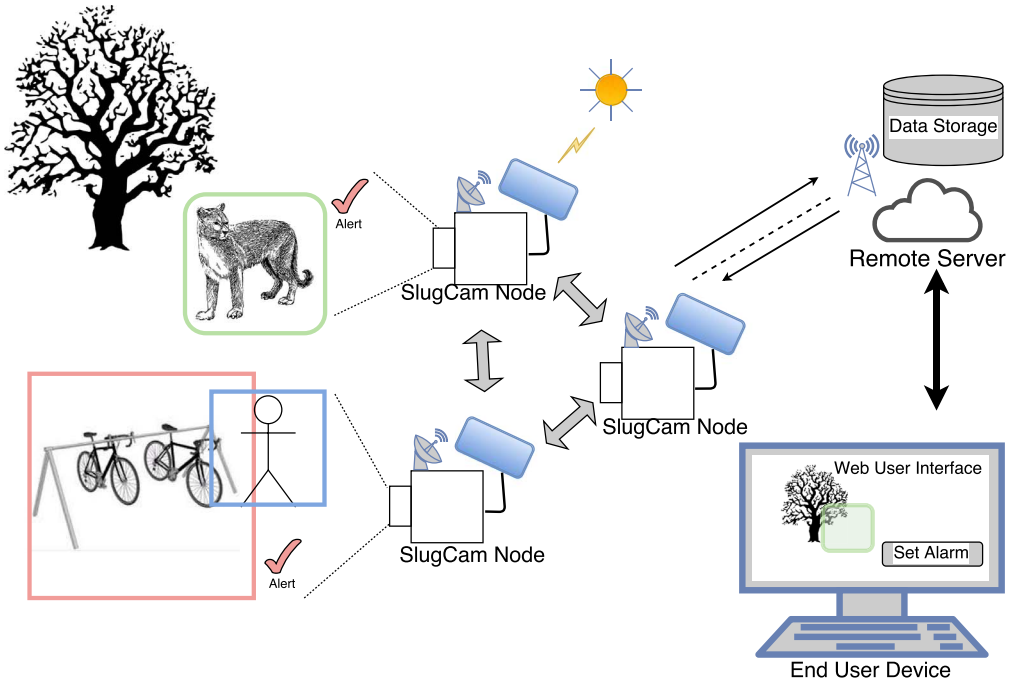

Fig. 5. Three-node mesh SlugCam deployment illustrating SlugCam's software components and their location.

4.1. Raspberry Pi manager

The Pi uses Arch Linux OS [9], which we found equally as powerful and capable as a custom Linux distribution. We use Arch Linux's built-in process (or "unit") and process management features, which enables all units on boot up. Process cooperation and communication is accomplished by using UNIX domain sockets [2].

As shown in Fig. 3, the Raspberry Pi manager is structured into 4 units, namely: the MSP430 unit, the computer vision unit, the communication unit, the communication unit, and the power aware unit. Each one of these units is described in detail below. All 4 components run on each SlugCam node independently as shown in Fig. 5

MSP430 unit

The MSP430 software module, which runs on a SlugCam node from Fig. 5, monitors battery levels as well as requests from other system units (and possibly remote users) to determine appropriate operational duty cycles over time. Since the Raspberry Pi does not have a hardware clock, the MSP430 also manages the real-time clock, this is different than other related systems which use a separate external clocks. [11].

Computer vision unit

SlugCams computer vision module works as follows: upon initialization, it takes pictures with 1280 × 720 resolution. It then and re-sizes them to 426 × 240 so they can be processed using the MoG method [12], which performs foreground extraction at an average of 3 frames per second (fps). Contours of the resulting foreground are established and then a bounded box is applied following the contours' limit (See Fig. 4). Since object classification is one of the most computationally complex tasks in computer vision systems [3], we chose to classify objects based on their dimensions in order to reduce processing and storage complexity. Then we use the objects location in the image provided we can start to track movements and intentions based on where the object goes to in the image.

Communication unit

The communication unit handles the networking stack as described in Section 5. The advantage of keeping it as a separate unit is that it acts as the gatekeeper to the wireless module. This is important as we are communicating over a serial connection, which means that would be difficult to coordinate multiple processes attempting to use the serial interface at the same time.

An important feature of our system is its robustness to arbitrarily frequent and long-lived disruptions in communication between its different components. To this end, we built all communication protocols following an "opportunistic" approach. For example, the video and message servers constantly wait for incoming connections from camera nodes and expect that those connections could be dropped at any time. Any messages that need to be sent to the camera must wait until that camera comes on-line; and as soon as the camera does come on-line we begin to send all outgoing messages. This opportunistic communication framework further increases camera autonomy at the cost of immediate communication ability. It allows for many interesting optimizations, such as reducing communication in low power situations and when only non-interesting data has been collected. Cameras can also be set to come alive on a timer to receive important control data from the network if needed in order to manage the communication latency inherent to this type of design.

Power aware unit

The power monitoring unit not only responds to requests of system battery levels, but if the battery reaches certain levels it will send the appropriate requests to other processes to alter duty cycle behavior or limit power hungry activities. Battery levels are approximated using the battery state signals given by the solar charger module and periodic monitoring of node energy consumption using the on-board current sensor explained in Section 6.

4.2. MSP430 firmware

The MPS430 firmware includes the ability to control the Raspberry Pi's power state, run a real-time clock, and receiving lengths of time over a serial SPI communication link. These time lengths, are period lengths before a system timer fires a interrupt service routine triggering the relay to cut power and put the system in a low powered sleep state.

[2] A complete functional description of UNIX domain sockets can be found in [10].

[3] Object classification typically uses training techniques to classify objects based on a previously stored database.

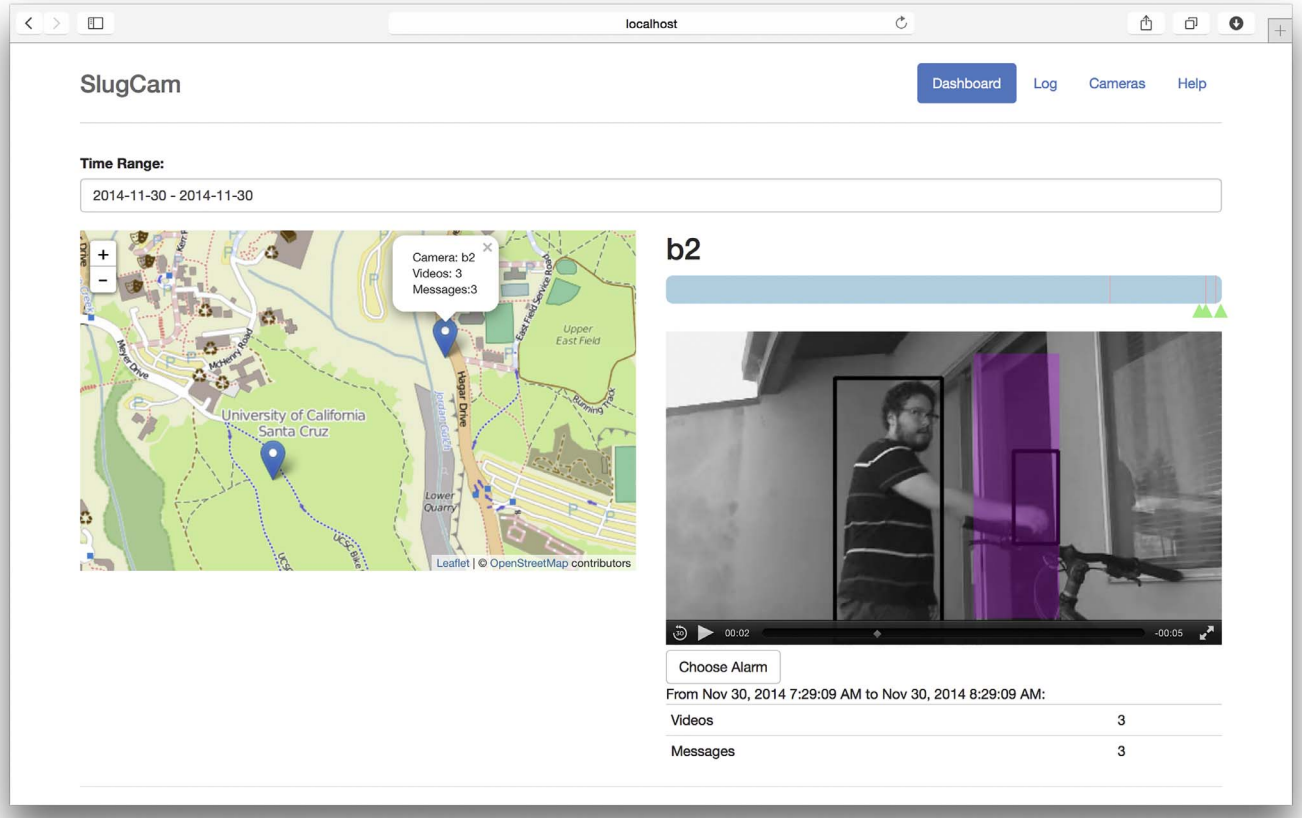

Fig. 6. Screen-shot of the dashboard of the client application featuring the activity bar, map, and alarm area selection.

The extremely low powered MSP430 makes it the perfect choice for remaining on to manage the systems power state.

4.3. Video server

The SlugCam video server ("Remote Server" in Fig. 5) has a Web-based front-end and a database back-end. The Web server responds to communication from the SlugCam user interface (described below) while the database back-end stores both raw video footage of each recording, as well as associated tags which mark relevant events in the video. The database server also stores SlugCam node management information such as battery life and scheduled events for the nodes. The video server also acts as the data sink for the SlugCam networking manager as described in Section 5.

4.4. User interface

SlugCam's Web-based user interface ("End User Device" in Fig. 5) allows end users to configure SlugCam nodes (e.g., scheduling video recordings, specifying events to be tracked/monitored, etc.), query the video server's database, check SlugCam node's battery status, etc.

Node management

Since SlugCam deployments may consist of potentially large numbers of nodes, the user interface displays all currently deployed nodes in a map of the area being monitored. This view is called "dashboard" as shown in Fig. 6. We also built what we call the activity bar which displays system activity over a period of time. This is an important aspect of our visual presentation as it allows users to correlate collected tags to the videos in which activity is happening in an intuitive way, but without hiding any information.

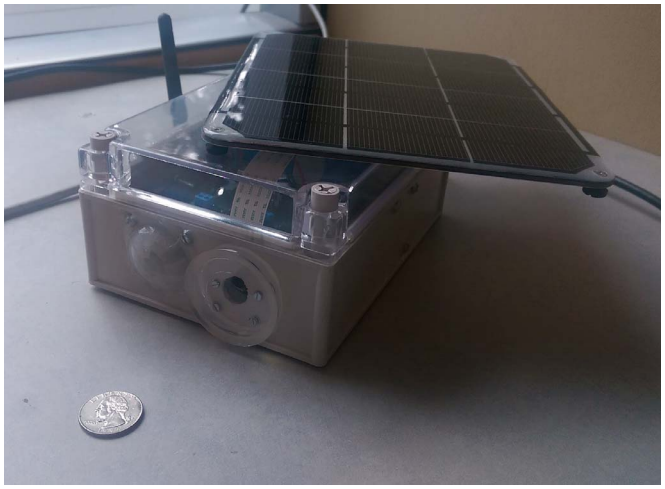

Fig. 7. The second iteration of the SlugCam prototype in its weatherproof enclosure and re-positional panel on top.

Setting monitoring alarms

Another important feature of SlugCam's user interface is the ability to set certain areas within the camera's field of view that will trigger alarms. When an alarm is triggered it could mean that the user is notified and the recorded video is instantly sent to the server, though we plan to allow users to define actions taken when alarms are triggered. The interface we have developed allows the user to select an area over the recorded video taking into account the current position of the camera. The alarm selection interface is currently available anywhere that video is displayed, and is demonstrated in our screenshot of the

dashboard in Fig. 6.

5. SlugCam networking

This section provides an overview of SlugCam's Communication Unit (see Fig. 3). For additional information on the design and implementation of SlugCam's networking functionality, the reader is referred to [13].

As described in Section 4, data communication in SlugCam is managed by the Communication Unit running on the Raspberry Pi. SlugCam's Communication Unit supports uni-cast, multi-cast, and broadcast communication. To this end, it provides two main routing modes, namely: flooding for multi-point communication, and uni-cast routing using the Dynamic Source Routing (DSR) protocol [14] for point-to-point exchanges. Nodes can choose one of these two routing methods for a given transmission. For instance, higher priority data can use flooding at the cost of increased resource consumption.

5.1. Multi-point communication

SlugCam's flooding mechanism is quite straightforward. Each packet originated at a node is provided with a unique ID and an optional time-to-live (TTL). In addition, each packet contains a list of destination nodes for the data (which can contain a broadcast address as well). When forwarding data, a node checks the packet ID against a table containing the IDs of previously forwarded packets from each originating node. If the packet has been seen, it will be dropped; otherwise, its ID is saved in the table. If the packet is not dropped, we then decrement its TTL and forward it if the TTL is still greater than zero. High priority data (e.g., alarms and emergency notifications) can use flooding to more quickly (flooding does not need to wait for routes to be discovered) and reliably reach their destinations (especially when multiple nodes must received the information) at the cost of higher resource consumption.

In addition to high priority data transmission, flooding can be used to support other system functions including updates and node synchronization.

5.2. Point-to-point communication

SlugCam uses the Dynamic Source Routing (DSR) protocol [14] to transfer video files and send status updates to the data sink (e.g., SlugCam nodes sending updates to SlugCam's central server on their battery level, etc.). As discussed in Section 7, there is a considerable body of work on routing protocols for MANETs. As such, choosing SlugCam's routing mechanism took into account a number of factors which we discuss in the remainder of this section.

Why reactive routing?

Traditionally, dynamic routing mechanisms compute routes reactively in response to routing updates. While proactive routing works well in networks where topology changes do not occur very often, they do not yield adequate performance when topologies change all the time. Reactive- or on-demand routing, on the other hand, have been proposed to accommodate networks whose topology dynamics would render proactive routing too expensive (in terms of the overhead they generate). In general, reactive routing protocols find routes on-demand, i.e., in response to a node's need to send data to another node. The main advantage of reactive routing is that the overhead of route discovery and maintenance does not occur unless and until routes are needed. Their main disadvantage is that there is additional delay in transmitting data if a route to the destination needs to be discovered prior to data forwarding.

SlugCam's requirements call for a hybrid routing approach: since every SlugCam node will need to communicate with SlugCam's central server at some point during a duty cycle (if for nothing else, they must send periodic status updates), it makes sense to pro-actively build routes to the central server (or the data sink). However, proactive routing incurs overhead to build routes that may never be used; thus, reactive routing is a suitable choice in this case.

Based on these considerations, we decided to use reactive routing and extend it to provide proactive routing functionality whenever needed. For example, since every node is required to send a status update to the data sink at the beginning of a duty cycle, a route to the data sink will be placed in every node's cache when the node starts. Since routes to the sink stay cached in most nodes, when another node comes on-line, its neighbors will most likely be able to provide the new node with a route to the sink. As a result, new nodes need not perform route discovery the first time they need to communicate with the sink.

Adopting reactive routing can lead to implementation challenges [15]. Below, we discuss some of these challenges in the context of our DSR implementation.

Advantages and disadvantages of DSR

DSR is a reactive source routing protocol. As such data sources discover routes to destinations on demand, i.e., when they have data to send to those destinations and routes to them do not exist in the node's route cache. Following the source routing mechanism, DSR packets carry route information in their header as illustrated by Algorithm 1. At each hop, the relay node gets information about the next hop from the packet header. Each node maintains a cache of known routes and, when data needs to be sent to a destination, the node checks its route cache for a route to that destination. If a route to the destination is not known, the data source performs route discovery by flooding a *route request* message. When a node receives a *route request* it checks its own route cache; if it knows a route to the destination, it will respond to the *route request* with a *route reply*; otherwise, it will append its own address to the list of visited nodes in the *route request* header and will forward the *route request*. When the *route request* reaches the intended destination, it will issue a *route reply* back to the source. The *route reply* will follow the reverse path back to the source and the corresponding forward path will be used to forward data from the source to the destination. Though this means that packet forwarding and routing functions are intermingled, the process of packet forwarding remains straightforward as shown in Algorithm 1.

An interesting feature of DSR is that its flooding of the *route request* allows multiple routes to be discovered. In our implementation of DSR, we take advantage of this feature to enable sources to discover and use multiple routes to destinations. Employing multi-path routing is key to achieve adequate throughput while balancing the load and power consumption among participating nodes.

Energy efficiency took a central role in SlugCam's design. Power awareness in routing protocols is discussed in detail in [16] and was integrated into our DSR implementation. In particular, one of the metrics used to select routes is based on the nodes' remaining battery level. The use of power-aware routing metrics coupled with the ability to route over multiple paths are part of SlugCam's energy efficient design.

5.3. SlugCam networking manager software architecture

SlugCam's networking manager runs as a daemon exposing a socket-based API through which other SlugCam software modules can easily access networking functions. One clear advantage of defining an API to access networking functionality is that it isolates other SlugCam software modules from any changes made to how networking functions are implemented (e.g., which routing protocol is used, etc). Another advantage is that the networking manager acts as the gatekeeper to the wireless module. This is important as we are communicating over a serial connection, which means that would be difficult to coordinate multiple processes attempting to use the serial interface at the same time.

In order to provide functional separation between the networking

```
Let p be the next received packet
for each received packet p do
    case p is an acknowledgment request:
      Send acknowledgment to requesting node;
     case p is an acknowledgment:
      Register the acknowledgment locally;
    case p is a route request:
      Append local address and advertised cost to request;
      Forward request to broadcast address;
     case p is a route reply:
      Add route to route cache;
      Send buffered packets waiting for this route;
     case p is a route error:
      Remove invalid route from route cache;
     case  p's destination is this node:
      Process p's payload;
     other:
      Add acknowledgment request to p;
      Send p to next address on route;
     endcase
end for
```

Algorithm 1. Packet inspection steps for forwarding a packet in SlugCam.

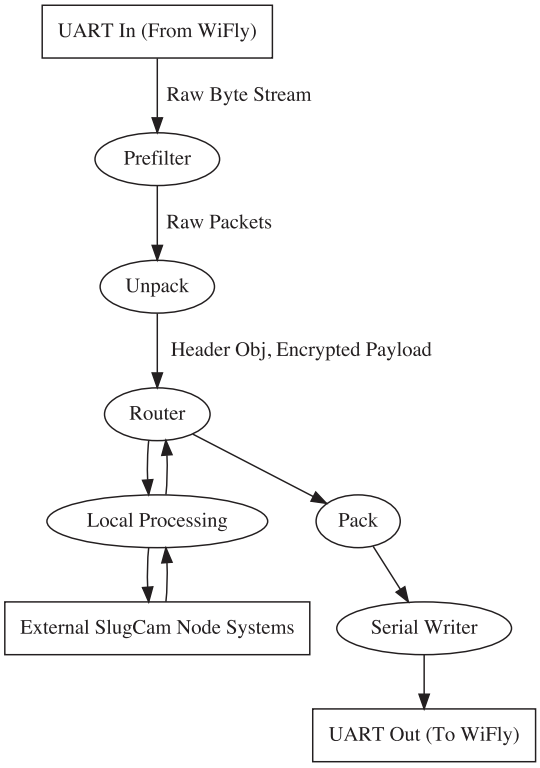

Fig. 8. SlugCam networking manager software architecture.

manager's modules, we designed them as a set of asynchronous threads as shown in Fig. 8.

Input processing

The first stage of data processing happens at the interface with the WiFly wireless hardware module. Its main function is to provide an abstraction of the underlying communication hardware, such that if other wireless hardware modules were used, this is where the changes would need to be implemented allowing the rest of the code base to be reused.

The WiFly module operates either in data mode or command mode communication occurs over the 802.11 WiFi connections established. The module begins in data mode in which data can be sent and received by directly reading from and writing to the serial interface. An escape sequence is written to the module to enter command mode from data mode. In command mode we have the ability to write commands to the serial interface in order to configure the module, connect to networks, and establish TCP connections. The inability to access the WiFly's firmware, however, does not allow us to change its network stack, including routing and forwarding. Therefore, we had to perform these functions at the application layer. Routing at the application layer has been widely used and evaluated to be effective even with the added delay of traversing the network stack [17].

In order to support SlugCam's multi-point communication, we use "broadcast" UDP for data transmission [4], which allows to broadcast-, multi-cast, or uni-cast data (see Section 5.4 for details on our application-layer addressing scheme). When receiving data, the WiFly passes the incoming UDP data stream to the application layer; for data transmission, the WiFly allows us to send a UDP data stream by writing to the serial interface. Since we have to perform routing and forwarding at the application layer, besides the payload, application-layer packets also need to carry control information. Our application-layer addressing and packet formatting scheme is described in Section 5.4.

[4] TCP does not support multi-point connections.

Once data is received from the WiFly module, the *pre-filter* scans the incoming byte stream and outputs raw packets containing a pre-header, header, and payload. As previously pointed out, SlugCam's packet addressing and formatting scheme is discussed in detail in Section 5.4 below. As illustrated in Table 2, the pre-header contains the packet's receiver address. If the receiver's address does not match the receiving node's address, the packet is dropped.

Unpacking, the next stage in packet processing, consists of un-marshaling the header into a structure for processing. Once packets are unpacked, they are passed to the routing module to be forwarded.

Routing

The *routing* module takes all packets originating locally or being forwarded by the node and then directs them to one of two sub-modules, namely: flooding or DSR-based point-to-point routing. This module is also responsible for forwarding locally addressed packets to the *local processing* module.

Local processing

In addition to routing and forwarding, SlugCam's networking manager also provides an abstraction layer over network resources to other services running on a node. It does this by exposing an API that is accessible via Unix domain sockets. The API provides an interface for reliably sending messages and video data across the network, demonstrated in our evaluation in 6. It also acts as a client interface to other domain sockets on the node to relay received data to the process responsible for handling certain inbound requests.

5.4. Packet formatting and addressing

Addressing

Since all outbound traffic is done over broadcast UDP as discussed in Section 5.3, we use our own application-layer addressing schemer. Each node is assigned a unique 32 bit unsigned integer as an identifier. This identifier is valid for all values x, where $0 < x < 4294967295$. Both zero and the value in which every bit is set (0xFFFFFFFF) are reserved addresses. Similarly to IPv4, 0xFFFFFFFF is reserved as a broadcast address.

Formatting

As shown in Table 1, SlugCam consist of a pre-header, a header, and the payload. The packet itself and each of its fields are delimited by ASCII control characters. The use of control characters mandates that the data contained within the packet does not contain these control characters. To this end, in our implementation, we encoded each field using the Ascii85 encoding scheme as it was relatively efficient and readily available.

The pre-header and header are both encoded in Ascii85, but are transmitted in the clear, while the payload is encrypted, signed, and encoded at the sending node before being split into packets. This means that data can only be decrypted and verified after all its packets are received.

The packet's pre-header (Table 2), consists of a receiver field and an offset field. As discussed in Section 5.3, data packets are sent as broadcast packets, so the receiver field indicates the packet's destination. Because it is in the packet's pre-header, it allows packets to be dropped even while scanning the input from the wireless module if the packet is not destined to the node.

The offset field is used to reconstruct the original data in the case it got fragmented into multiple packets (e.g., in the case of video file transfers).

The size and content of SlugCam's packet header may vary, e.g., depending on which routing scheme (i.e., flooding or DSR) is being used [5] DSR, for example, specifies headers of different types and

[5] Although varying size headers are more costly to process, they allow network protocol evolution.

Table 1
SlugCam packet.

Section	Description
Delimiter (0 × 01)	The ASCII SOH character
Pre-header	Encoded pre-header
Delimiter (0 × 00)	Null byte
Header	Encoded header
Delimiter (0 × 00)	Null byte
Payload	Encrypted and encoded
Delimiter (0 × 04)	The ASCII EOT character

Table 2
Pre-header description.

Field	Type	Description
Receiver	uint32	Who should receive this packet
Offset	int64	The offset of the payload data

different lengths [14].

In order to meet these requirements, we defined a header structure in which each packet has a required source identifier, which will always identify the node that originated the packet, but the rest of the header depends on the packet type. There are optional destination, TTL, and priority fields. If any of these are missing when needed (e.g., the destination address in a DSR packet), the packet is dropped.

5.5. Modifications to DSR

We augmented DSR with multi-path routing and priority forwarding functionality. We have also addressed power awareness in routing and the ability for nodes to advertise a routing cost. To address the latter, we change DSR to include a node's cost in route replies. More specifically, when a node receives a route request, it appends both its cost and address to the route request before forwarding it. When the originator of a route request then receives its route replies, it will store the costs of each node in a cost table and the address data in the route cache. When a route is needed, we look through the route cache for the set of all routes to a target (R) and choose a route r from this set by computing its probability $Pr(r, R)$ using Eq. (1), where $Cost(i)$ is the sum of the costs of all intermediate nodes in route i.

$$\Pr(r, R) = \frac{w_r}{\sum_{x \in R} w_x}; w_i = \frac{1}{Cost(i)} \quad (1)$$

Thus a route is chosen on a per-packet basis using a simple stochastic policy. In SlugCam, we use this multi-path routing technique as a way to load balance across participating nodes to achieve power efficiency. This is a well-known technique which has been used in powerware routing [18].

6. SlugCam validation, evaluation, and deployment

This section describes our experimental evaluation of SlugCam's networking functionality using a lab testbed, thorough power characterization of SlugCam's node, and real deployment of SlugCam in the field.

6.1. Network testbed experiments

We setup a SlugCam wireless testbed in our lab. In the first set of experiments, we use 3- and 4-node single-path line topologies. The second set of experiments employs a 4-node topology that contains alternate paths so that we can experiment with SlugCam's multi-path routing feature.

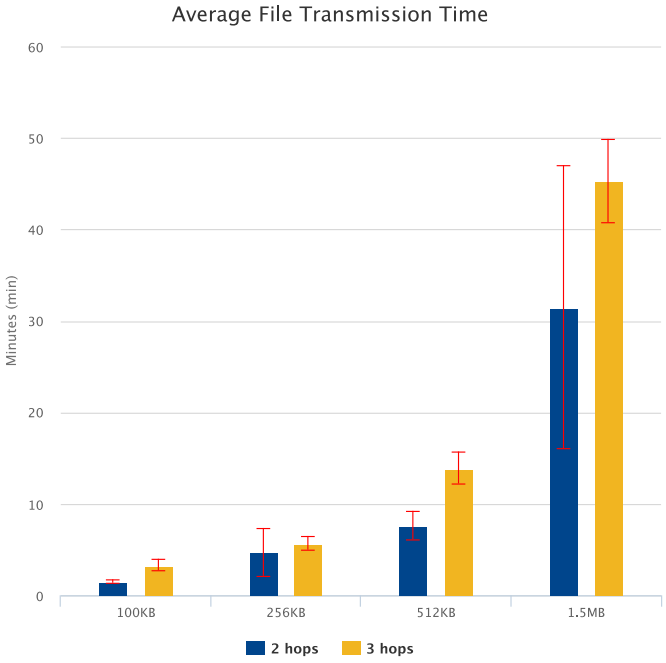

Fig. 9. Average transmission times and standard deviation for different file sizes in 2-hop and 3-hop topologies.

Single path experiments

The SlugCam network treats both video and message data the same as far as the reliability and acknowledgment mechanisms are concerned. Once data is registered with the daemon it splits it into packets which are sent to a destination through the DSR module. The data is stored and resent until all the data is acknowledged as being entirely received by the destination node. Since message data was treated the same as video data and video data consists of multiple packets we chose to run our tests using video data.

We use four different file sizes corresponding to video clips of approximately one- to fifteen minutes in length. We expect these to be the average length of most video clips recorded by SlugCam. Fig. 9 shows end-to-end delay for the different file sizes when transmitted over the 2- and 3-hop line topologies. As expected, the delay increased significantly for the largest file; several factors may have contributed to this delay increase including wireless interference which increases significantly with the number of wireless hops. We also observe that delay does not increase significantly when comparing the 2- and 3-hop topologies.

In our current SlugCam implementation, we included the ability to log all incoming and outgoing packets. Since we can count all packets sent over each link and all packets received over a link, we are able to calculate packet loss over the entire network and on each link. We report average packet loss which is calculated for each file size by averaging packet loss over the entire line topology. We observe from the results in Fig. 10 that loss remains fairly constant for any file size, but slightly higher loss ratios are reported if the data needs to traverse more wireless links which is expected.

Multi-path experiments

In the second set of experiments, we use the topology shown in Fig. 11. Setting up a 4 node topology of SlugCam nodes for our SlugCam routing tests gave us results with real-world anomalies that wouldn't have been possible to retrieve from a WVSN simulation. For example, Loss was not consistent over the wireless links due to occasional interference and in all three flow graphs from Fig. 11 packets were sometimes able to travel from the farthest apart nodes with our efforts to distance them apart. We believe the data gathered from our WVSN testbed is extremely valuable information for the field and a contribution not yet seen before in other works. We conducted 3 separate

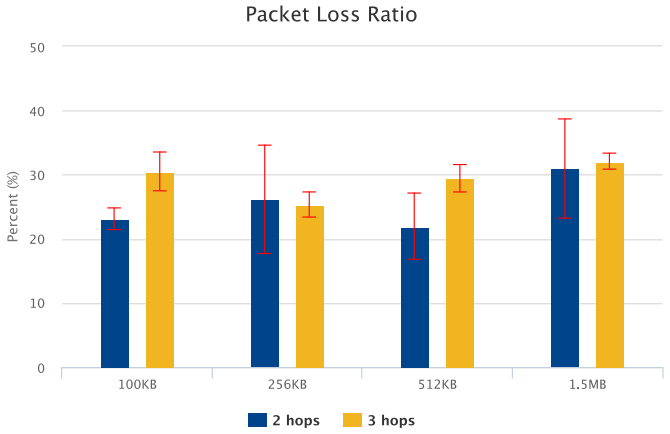

Fig. 10. Average end-to-end packet loss and standard deviation for different file sizes traveling over a 2-hop and 3= hop line topology.

experiments, shown as the 3 packet flow diagrams in Fig. 11, showing different scenarios detailing how SlugCam adapts to three distinct node states in its mesh network.

- *Balanced cost scenario*: In diagram (a) of Fig. 11. we set the source node sc3 to transmit a video file to the destination node sc1 using our implementation of DSR while having all nodes advertise an equal cost to each packet. As was expected we were able to see an equal number of packets flow through each of the available routes to the destination node. Even the most difficult link to traverse due to wireless range, node sc3 to sc1, was seen as successfully transmitting some packets.
- *Node network drop-off*: The purpose of scenario (b) was to show SlugCam's behavior while one of the nodes is unavailable either due to a sleeping duty cycle or the depletion of is rechargeable battery waiting for sunlight. The logged traffic shows packets all traveling through the only available route, sc3 to sc2 arriving at the sink node at sc1.
- *Cost difference test*: To showcase our power aware modification to DSR, one relay SlugCam node advertised a high cost indicating a low battery state and then we continued with the transmission experiment. Based on the results displayed in diagram (c), packets did favor the route with the lower cost while avoiding node sc4. Some packets did traverse routes through sc4 however and that is due to the very nature of our lottery system SlugCam employs described in detail in Section 4.

6.2. Evaluating SlugCam's power efficiency

In order to prove the systems has enough energy efficiency to deploy the system in real world scenarios we needed to do some accurate power analysis of the system. Using a simple 10 point moving average filter, we were able to smooth out our current consumption data for readability without sacrificing accuracy. We provide a proper current consumption analysis of the system both evaluating the duty cycles we defined and analyzing how the system achieves efficiency through some example application scenarios.

Energy gains using the MSP430

One well-known and widely-used sensor network power savings techniques is to keep the system in a low-power state whenever it is idle, i.e., not executing any specific task. In SlugCam, we go a step further and use a low-power micro-controller (the MSP430) to control: (1) a PIR motion sensor to wake-up the Pi when it detects motion and (2) a relay circuit to cut power to the Pi based on an on-board MSP430 timer, which can be dynamically set by the Pi. In order to show the power savings achieved through our design, we run SlugCam for 5 min during which the system wakes up 6 times. When it wakes up (triggered by motion detected by the PIR), the Pi turns on the camera for 30 s before going back to sleep. As baseline, we run the system uninterruptedly for 5 min. Fig. 12 shows the current consumed in the two runs. We use a 10-point moving average filter to smooth out the sampled current consumption data for readability without sacrificing accuracy. Calculating the area under both curves, we observe that the system saves around 30% when monitoring a relatively busy scene. The dips in the curve show that in the low power state, the system is consuming between 60 and 70 mA, while in the "full on" state, it consumes around 400 mA. We note that the power savings can be even greater if we take advantage of the MSP430 deep sleep mode when it only requires current in the order of microAmps. We also performed a comparison between the recent Raspberry Pi B+, which SlugCam is using, and its precursor, the Raspberry Pi B. We found that latter consumes on average 150 mA with no connected components and the new B+ model consumed around 95 mA.

Adaptive duty cycling

Duty cycling has been adopted by wireless sensor networks in general, and visual sensor networks in particular, as an effective energy savings technique [1,19]. It typically works by having sensor network nodes switch to low-power states as much as possible in order to save overall energy consumed. SlugCam uses what we call "adaptive" duty cycling which allows the system to use current battery level information to guide its choice of duty cycles. In particular, when battery levels are too low (i.e., lower than a given threshold), the system avoids duty cycles that power hungry. Clearly, this typically comes at the price of degrading performance. This trade-off between performance and energy efficiency can be adjusted to meet application requirements by controlling the battery level threshold parameter.

We show SlugCam's power consumption characteristics for the six different example duty cycles described below:

- *Duty cycle A:* This duty cycle illustrates the case of the remaining battery level being below the threshold. In this particular experiment, the system wakes up and runs system services in which it determines the battery remaining is below the threshold, and then returns to a deep sleep state.
- *Duty cycle B:* In duty cycle B, the Pi is turned on by the PIR sensor and triggers the camera. The camera is turned on but does not run any computer vision functions. The WiFi module is also asleep as this duty cycle only records video and stores it locally. An alternate, lower power duty cycle to this one could take a high resolution image, instead of recording a video.
- *Duty cycle C:* Duty cycle C is the same as duty cycle B except that the system uses computer vision algorithms when recording data. In this particular instance, noticing the data is not important (based on the computer vision results), the system quickly transitions to low power state and the MSP430 cuts power to the Pi via the relay. As shown in the bottom left graph of Fig. 13, at low-power state, the system (essentially the MSP430 in idle mode) consumes around 70mA. However, as previously pointed out, power consumption can be substantially decreased by having the MSP430 go to sleep.
- *Duty cycle D:* Duty Cycle D is very similar to C with the only difference being that in C nothing is recorded while in D the video is stored locally. This explains the difference, albeit small, in power consumption between the C current consumption curve (in the bottom left graph) and D's (in the bottom left and right graphs of Fig. 13, respectively).
- *Duty cycle E:* The most complex and longest running duty cycle is E, during which SlugCam records video (longer than D) with computer vision turned on and then immediately transmits the data. In Fig. 13, for the E current consumption curve in the bottom right graph, the first drop in current consumption happens when the camera turns off and the WiFi module is on in order to transmit the

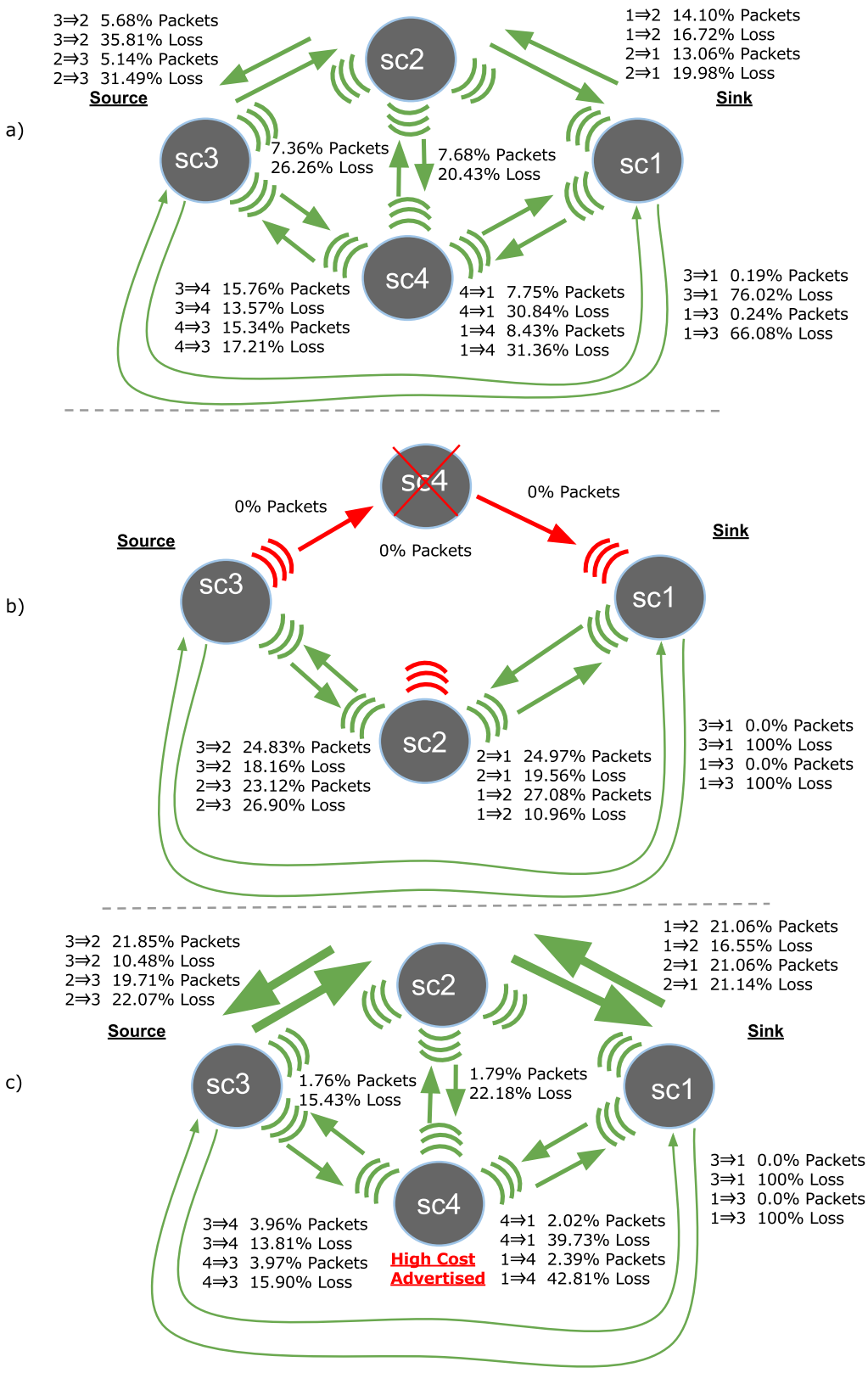

Fig. 11. a) The multi-path packet flow distribution and loss ratio of 4 SlugCam all making advertisements of equal cost. b) A camera node becomes inactive due to battery depletion and the packet flow with packet loss is shown. c) SlugCam 4 node advertises a higher cost due to low battery levels so more traffic is routed through node 2.

video file to our Web server. The second in current consumption drop corresponds to the system going back to its sleep state with the MSP430 in idle state.

- *Duty cycle F:* In duty Cycle F, the system wakes up on its own to ping the Web server for any requests or to report status updates. This time is also used to transmit any unsent video data. Notice though that the camera is never turned on and SlugCam consumes around 350 mA during data transmission. The drop at the end shows the WiFi module and the Pi being put to sleep and the Raspberry Pi turning to idle.

Another interesting feature to notice from the current consumption

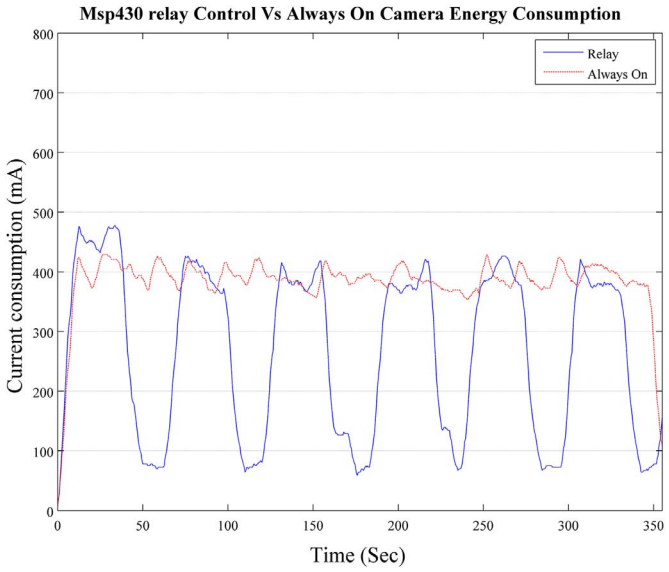

Fig. 12. SlugCam power consumption showing power savings by using the MSP430 to control the PIR motion sensor and power relay.

curves in Fig. 13 is there is very little difference in power consumption when the camera is on and computer vision software is executed. This is the case without using the Pi's GPU which we plan to use in the future. Another interesting observation when looking at the upper right graph comparing duty cycles B and F is that the camera consumes more power than the WiFi module. Comparing the curves in the bottom right graph for duty cycles D and E, we observe the difference in duty cycle duration; more specifically, duty cycle E's "active" period, i.e., when the system is not in idle state, is almost twice as long as duty cycle D. In this particular comparison, it shows the difference between storing the data locally (duty cycle D) and transmitting to a remote server (duty cycle E).

In the next set of experiments, we have SlugCam cycle through different duty cycles as it would in normal operation. More specifically, we run the system for 5 min during which 3 distinct events occur. The first event is a false positive, e.g., light or heat radiation triggers the PIR but the video analysis detect no important activities; therefore no video is recorded or transmitted. The second event is data that should be recorded for reference later but does not get transmitted because it is deemed not urgent or the battery level is too low for transmission. The third and final event records video and transmits it to the remote server as it is considered by the user to be an urgent event. The user's ability to set these alarms is described in detail in Section 6.

In Fig. 14 we show the events played over 5 min, while SlugCam uses different duty cycles in all three scenarios. The first scenario exemplifies a system with no adaptive abilities as baseline for comparison. In this scenario, all three events are handled by duty cycle E, as previously described, where data is recorded and transmitted for all events. The second scenario showcases the power saving benefits of adding computer vision to capture and analyze video footage. SlugCam is able to correctly identify the first false positive event and immediately go back to low power state. This operation is represented by duty cycle C. In this scenario however, SlugCam then decides that the next two events should not only be recorded but immediately transmitted as well, which is handled by duty cycle E. Scenario 3 showcases the system ability to ignore the first event (duty cycle C), record and not transmit the second event (duty cycle D), and record and transmit the third event (duty cycle E).

6.3. SlugCam deployment

In this section, we describe our experience deploying SlugCam on the UCSC campus which also allowed us to showcase its management- and user interface capabilities. We carried out a single-camera deployment as follows. At the beginning of the experiment, all persistent data on the remote server was cleared. We set up a SlugCam node outdoors and ran through a duty cycle in which the camera would turn

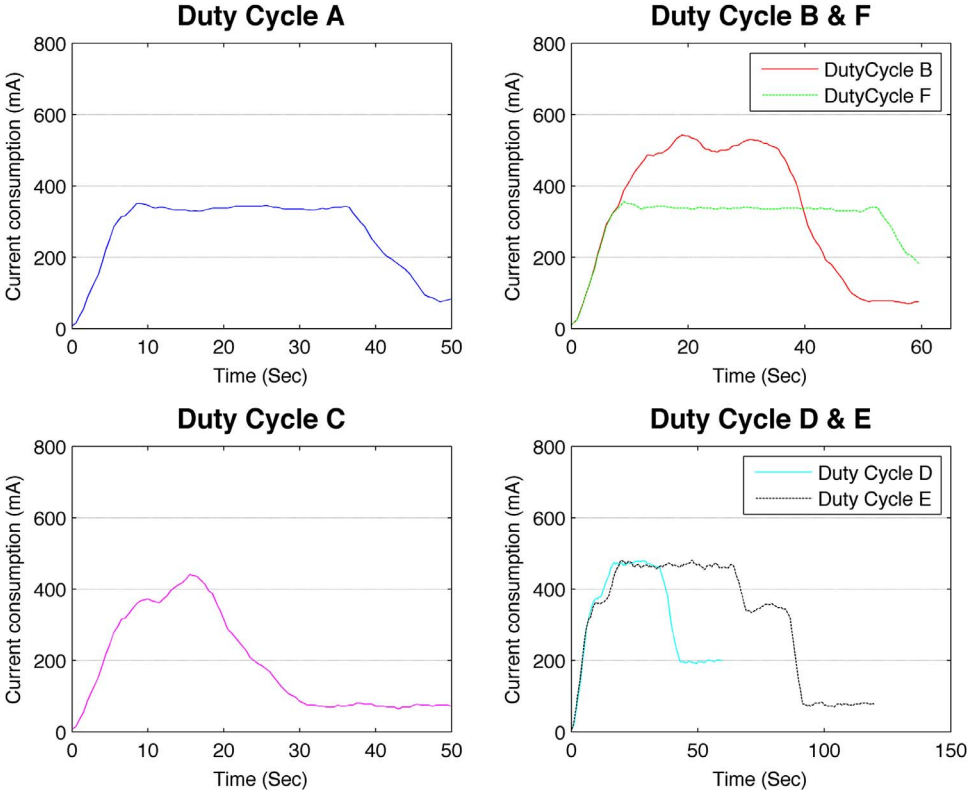

Fig. 13. Power consumption characterization for SlugCam's duty cycles.

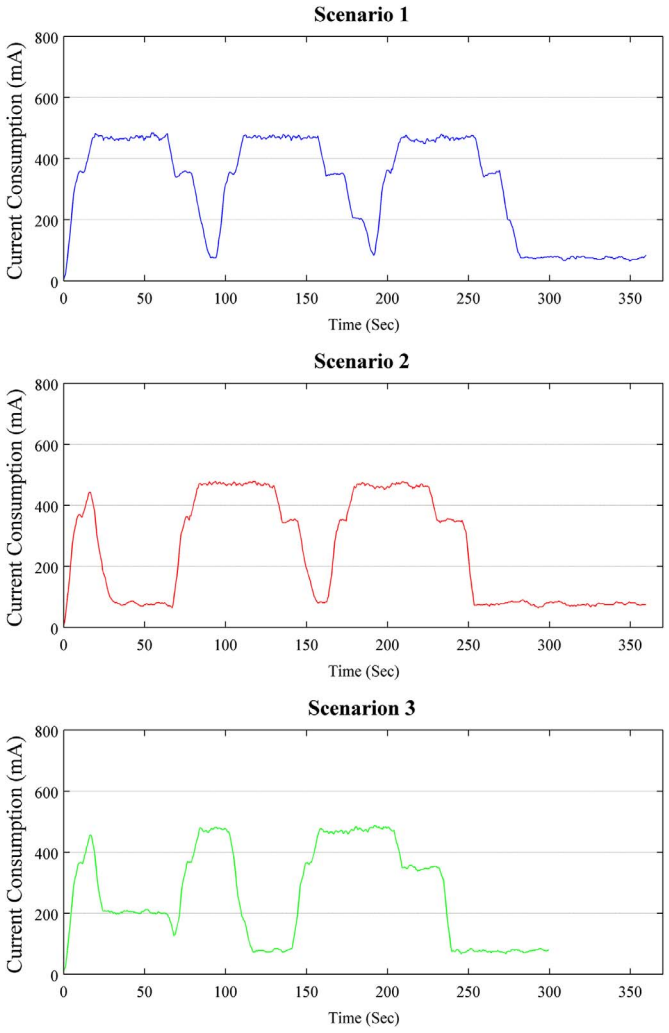

Fig. 14. Current consumption data for the three application scenarios.

on, begin recording and tagging the video, and then send the recorded video to the server. The duty cycle was run several times after which we started the client application to examine the data collected by the system. When opening the client application from a web browser, we were brought to the dashboard and notified that our camera node needed a location set in order to show on the map. We then followed the link to the camera configuration page. On the camera configuration page we were able to successfully set the camera location automatically using our browsers location data. Upon returning to the dashboard, we were able to open the video on the activity bar (discussed below) for analysis. We saw all of the sent tags and videos on the activity bar and we were able to view the videos in the video player. We then used a camera simulation script to send a message to the server to test the map display with multiple cameras. After setting the location for the second camera we returned to the dashboard and were able to switch between the cameras and still view the videos with tags. The videos also show the object recognition algorithm results represented as black squares around detected objects.

Since SlugCam deployments may consist of potentially large numbers of nodes, the system's management tool displays all currently deployed nodes in a map of the area being monitored. This view is called "dashboard" as shown in Fig. 6. We also built what we call the activity bar which displays system activity over a period of time. This is an important aspect of our visual presentation as it allows the user to correlate collected tags to the videos in which activity is happening in an intuitive way, but without hiding any information. This is also shown in Fig. 6 above the video preview.

Another important feature is the ability to set certain areas within the camera's field of view that will trigger alarms. When an alarm is triggered it could mean that the user is notified and the recorded video is instantly sent to the server, though we plan to allow users to define actions taken when alarms are triggered. The interface we have developed is intuitive and allows the user to select an area over the recorded video taking into account the current position of the camera. The alarm selection interface is currently available anywhere that video is displayed, and is demonstrated in our screenshot of the dashboard in Fig. 6.

The application also includes more traditional methods of filtering and querying the data. For example, as shown in Fig. 15, the user can query the specific data and have it displayed in tabular format. This allows for information to be viewed in a more targeted way. There are also views that present camera configuration options and status (as shown in Fig. 16). These currently allow for the camera location to be set and plan to include additional features such as battery level and usage statistics.

7. Background and related work

Over the past few years, smart camera systems have received considerable attention from academia and industry as reported in our recent survey [1]. Rather than a thorough survey of the state-of-the-art in wireless smart camera networks (which can be found in [1]), this section describes smart camera systems that are more closely related to SlugCam.

7.1. Selected smart camera systems

Smart cameras typically combine energy efficient hardware with computer vision software which enables them to capture selected data from the scene to either store, transmit, or both [20,21]. Computer vision techniques have advanced considerably and are able to not only identify objects accurately but also detect object behavior [22]. In order to keep up with advances in computer vision, smart camera systems require greater processing capabilities. This is especially the case in wireless platforms designed to be deployed in remote, hard-to-reach outdoor areas. Recent solutions, including SlugCam, have been using symbolic information from object behavior in the video footage in order to extract more meaningful information from the environments they monitor [23]. Wiseeye is another more recent system that looks at this kind of "meta data" (they refer to as "traps"), or sensor-captured events to trigger video recording [11].

Although sensor node duty cycling has been used in smart camera platforms as a way to save battery power [24], SlugCam takes a unique approach to duty-cycle based battery management. In contrast to other solar-powered smart camera systems that take snapshots to save battery [25], SlugCam is able to capture video sequences by powering on node components based on remaining battery life. Additionally, it uses knowledge that the battery will recharge when sunlight returns.

In our survey [1], we have examined recently developed smart camera platforms, many of which accurately track and detect objects in an energy efficient way. A common feature of these smart camera systems is to employ on board processing for better video filtering and selection. This results in lower resolution data being transmitted which translates into network- and energy efficiency gains. A notable example is Citric, a smart camera platform targeting dense smart camera deployments [21]. It uses audio sensors (instead of the PIR used in SlugCam) to keep the camera "off" and only turn it on when activity is detected. Audio sensors also help track and locate objects for the cameras. Flexi-WVSN presents an extensive evaluation of previous wireless visual sensor networks [19]. Realizing it was hard to stay up to date with hardware advances, Flexi-WVSN developed a modular system so no sub-component is dependent on any other piece of hardware. For

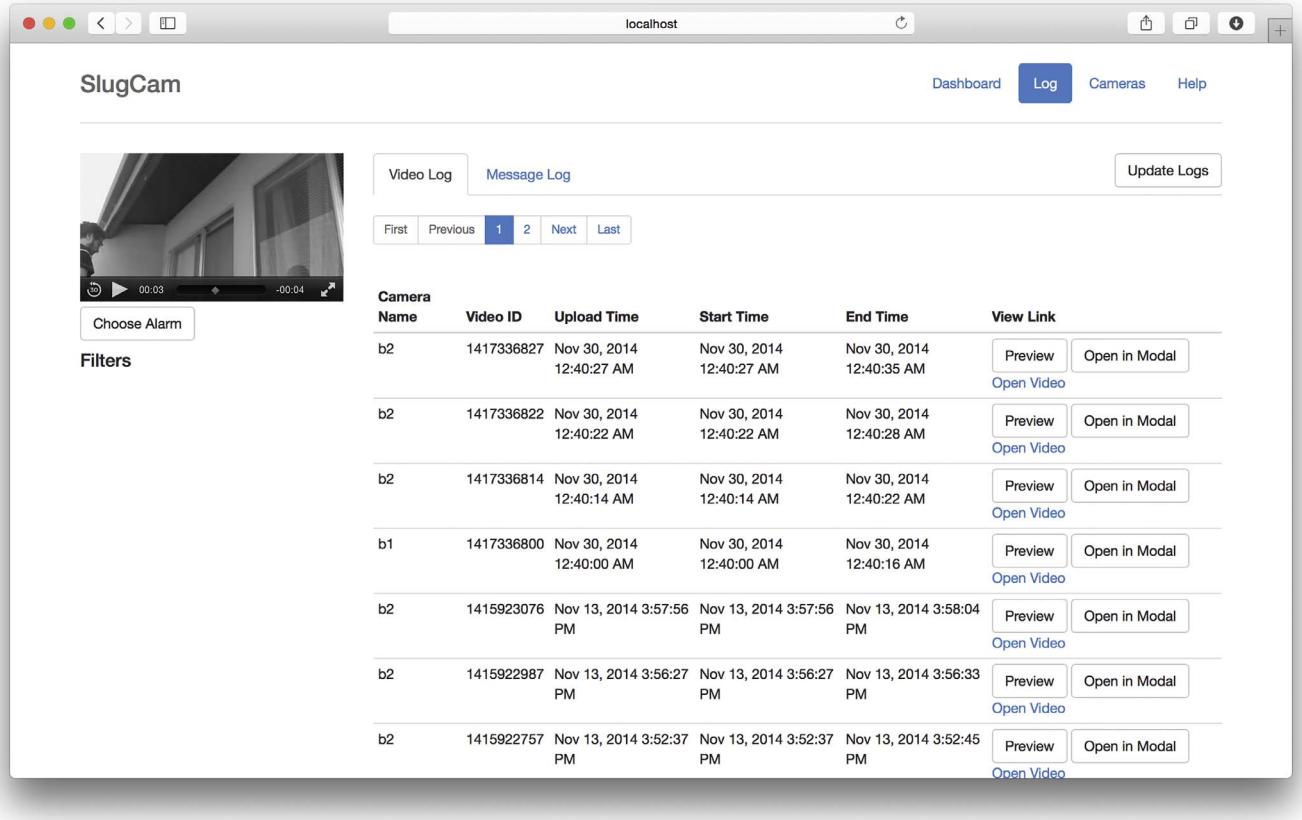

Fig. 15. SlugCam data viewed by the client application.

energy efficiency, rather than a two microprocessor solution like SlugCam's, they chose a two radio system. When smaller data files need to be sent, the system can switch to a more energy efficient Zigbee radio; otherwise they use WiFi to transmit video data.

7.2. Solar power in visual sensor platforms

Many sensor networks have been deployed outdoors with the ability to harvest solar energy [11]. Solar power combined with rechargeable batteries allows sensor networks to be less reliant on batteries alone and on the power grid [3]. Energy harvesting capable nodes are becoming much more efficient: there are now nodes capable of completely self-powering the image sensor [26]. UC Berkley has developed the world's largest outdoor solar sensor network testbed with 557 nodes [27]. They use a combination of 802.15.4 (Zigbee) at the leaf nodes and an 802.11 backbone. To cope with variability in sun exposure, they too chose to use different operation duty cycles based on the node's available energy. Although visual sensors were included, their focus was on achieving the maximum sensor fusion capabilities possible.

FireWxNet [28] is another wireless sensor network with solar capabilities. It adapts to available energy resources by adapting its duty cycling. While their visual sensor nodes do not perform visual processing, the authors claim that the system's auxiliary sensor nodes provide enough information to allow users to decide if they wish to receive a live video stream of the monitored area. In contrast to SlugCam's use of 802.11 WiFi to transmit high resolution video data, FireWxNet uses 900 Mhz radios to achieve long range communication. Doing this however sacrifices the ability to perform larger data transfers. Another notable solar-powered system is described in [20]. It throttles its image sensors based on the relevance of the data recorded. The system also uses computer vision algorithms based on background subtraction.

In light of existing wireless camera systems, one main distinguishing feature of SlugCam is that it provides visual sensing applications with a complete solution, including hardware and software, based on an open system design philosophy, including: (1) low-cost, off-the-shelf hardware components, (2) open-source management and user interface software, (3) multi-hop communication capabilities, and (4) open-source computer vision mechanisms. In SlugCam, energy efficiency is achieved both by fine-grained management of low-power hardware components, as well as by having the system's operation duty cycles automatically adapt to the current state of the battery in order to balance the trade-off between application-level requirements and power efficiency. SlugCam's on-board processing capability allows sophisticated computer vision algorithms to run locally which contributes to the system's autonomous operation capabilities, energy-, and bandwidth efficiency.

7.3. Wireless multi-hop ad-hoc networking

SlugCam employs the IEEE802.11's adhoc networking capabilities in order to address deployments in remote areas where connectivity is sparse or non-existent. However, using this self-organizing network structure does have some challenges; specifically, as described in Section 5, SlugCam implements an on-demand, wireless, multi-hop routing protocol to enable transmission of data (video, text, alarms, etc.) between its wireless camera nodes.

The performance of on-demand and proactive routing protocols for wireless multi-hop ad-hoc networks (MANETs) has been studied extensively using simulations [29]. Implementations of both types of protocols have been developed for the Linux kernel to allow their functionality and performance to be studied in more realistic scenarios [15].

In general, existing wireless sensor networking systems have a different focus than SlugCam's. Though they share an interest in power

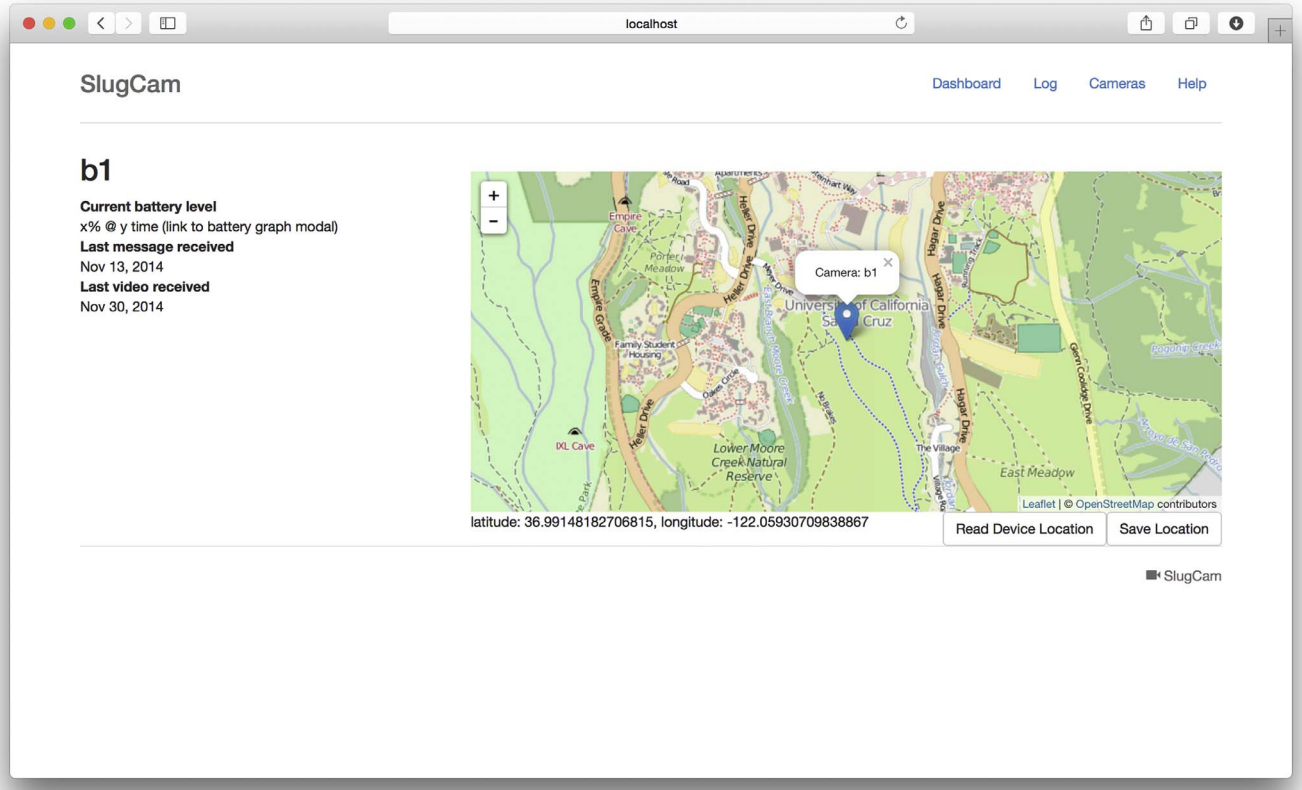

Fig. 16. Camera configuration application, displaying camera placement information.

awareness, they often target unidirectional communication to a data sink. In SlugCam, we also allow for peer-to-peer communication between nodes, as well as full- or scoped broadcast, e.g., to propagate control signals from the sink and communicate alerts. In addition, control signals to the nodes from our central server is an important part of our communication needs. A number of routing schemes for wireless visual sensor networks (WVSNs) have focused on streaming video in real time [30–36] and most of them are implemented using simulation platforms. Live streaming is not a requirement for SlugCam which is geared towards a range of applications that are tolerant of delayed video transmissions but require timely transmission of alarms (e.g., in case of event detection). The wireless smart camera network described in [37] focuses on deployment in extreme environments with intermittent connectivity; however, they do not explicitly address energy efficiency and power awareness.

Transmission of large video data in WVSNs is a challenge and it is well known that there is no “one size fits all” protocol for the wide range of requirements imposed by applications. In [30], the LANMAR routing protocol was used because nodes need to stream video data while remaining mobile which is different circumstances than in the SlugCam network where stationary nodes are used. Similarly to SlugCam, the Improved Energy Efficient Ant-Based Routing algorithm [38] focuses on improving energy efficiency using a stochastic forwarding policy; however they conclude that a reactive approach is something they wish to investigate in the future. The work reported in [39] created and evaluated a hybrid routing scheme implementation for a WVSN of Bluetooth-enabled search and rescue robot node. Results using simulations and a latency-constrained 5-node network were reported. Since SlugCam targets deployments using low-power WiFi for higher bandwidth and longer-range communication range outdoors, we deploy SlugCam in our SlugCam wireless visual sensor network testbed and test it for transmitting real recorded video data over different network topologies.

8. Conclusion

In this paper, we introduced SlugCam, a solar-powered, wireless, smart camera network that can be used in a variety of outdoor applications. SlugCam’s main contributions include: (1) open-system hardware and software design which makes SlugCam low cost and modular and facilitates prototyping and evolution; (2) energy efficiency achieved both by fine-grained management of low-power hardware components, as well as by having the system’s operation duty cycles automatically adapt to the current state of the battery balancing the trade-off between application-level requirements and power efficiency; (3) on-board processing capability which allows SlugCam to run computer vision algorithms locally, contributing to the system’s autonomous operation, as well as energy- and bandwidth efficiency; and (4) power-aware networking functionality which allows efficient point-to-point and multi-point data communication.

References

[1] K. Abas, C. Porto, K. Obraczka, Wireless smart camera networks for the surveillance of public spaces, Computer 47 (5) (2014) 37–44, http://dx.doi.org/10.1109/MC.2014.140.
[2] A. Pinto, Z. Zhang, X. Dong, S. Velipasalar, M.C. Vuran, M.C. Gursoy, Analysis of the accuracy-latency-energy tradeoff for wireless embedded camera networks, Proceedings of the Wireless Communications and Networking Conference (WCNC), IEEE, 2011, pp. 2101–2106.
[3] S. Sudevalayam, P. Kulkarni, Energy harvesting sensor nodes: survey and implications, IEEE Commun. Surv. Tutorials 13 (3) (2011) 443–461.
[4] Low-powered texas instruments microcontroller, 2014. Available at http://www.ti.com/msp430launchpad.
[5] National renewable energy laboratory, 2014. http://www.nrel.gov/rredc/.
[6] M. Dirjish, Whats the difference between thin-film and crystalline-silicon solar panels, 2012. http://electronicdesign.com/power-sources/.
[7] A. Industries, Li-ion/li-polymer battery charge management controller, 2014. https://www.adafruit.com/products/390.

[8] A. Microsystems, Acs712 hall effect current sensor, 2014. http://www.allegromicro.com/.
[9] A.L. Group, The arch linux operating system for the raspberry pi, 2014. https://www.archlinux.org/.
[10] W.R. Stevens, B. Fenner, A.M. Rudoff, UNIX Network Programming, 1 Addison-Wesley Professional, 2004.
[11] S. Nazir, S. Newey, R.J. Irvine, F. Verdicchio, P. Davidson, G. Fairhurst, R.v.d. Wal, Wiseeye: next generation expandable and programmable camera trap platform for wildlife research, PLoS ONE 12 (1) (2017) 1–15, http://dx.doi.org/10.1371/journal.pone.0169758 URL http://srl.informatik.uni-freiburg.de/conferences/965icra09ws/papers/16P-Johnsen.pdf.
[12] S. Johnsen, A. Tews, Real-time object tracking and classification using a static camera, Proceedings of IEEE International Conference on Robotics and Automation, Workshop on People Detection and Tracking, (2009) URL http://srl.informatik.uni-freiburg.de/conferences/icra09ws/papers/16P-Johnsen.pdf.
[13] L. Miller, K. Abas, K. Obraczka, Scmesh: solar-powered wireless smart camera mesh network, (2015).
[14] D. Johnson, Y. Hu, D. Maltz, RFC: 4728, The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPV4 (2007).
[15] V. Kawadia, System services for ad-hoc routing: Architecture, implementation and experiences, Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM Press, 2003, pp. 99–112.
[16] J. Al-Karaki, A. Kamal, Routing techniques in wireless sensor networks: a survey, IEEE Wirel. Commun. 11 (6) (2004) 6–28, http://dx.doi.org/10.1109/MWC.2004.1368893.
[17] P. Zave, Requirements for routing in the application layer, Coordination Models and Languages, Springer, 2007, pp. 19–36 URL http://link.springer.com/chapter/10.1007/978-3-540-72794-1_2.
[18] C. Schurgers, M.B. Srivastava, Energy efficient routing in wireless sensor networks, Proceedings of the IEEE Military Communications Conference on Communications for Network-Centric Operations: Creating the Information Force. 1 IEEE, 2001, pp. 357–361 URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=985819.
[19] A. Seema, M. Reisslein, Towards efficient wireless video sensor networks: a survey of existing node architectures and proposal for a flexi-WVSNP design, Commun. Surv. Tutorials IEEE 13 (3) (2011) 462–486.
[20] M. Magno, D. Brunelli, P. Zappi, L. Benini, A solar-powered video sensor node for energy efficient multimodal surveillance, Proceedings of the 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, (2008), pp. 512–519, http://dx.doi.org/10.1109/DSD.2008.23.
[21] P. Chen, K. Hong, N. Naikal, S.S. Sastry, D. Tygar, P. Yan, A.Y. Yang, L.-C. Chang, L. Lin, S. Wang, et al., A low-bandwidth camera sensor platform with applications in smart camera networks, ACM Trans. Sens. Netw. 9 (2) (2013) 21.
[22] W. Hu, T. Tan, L. Wang, S. Maybank, A survey on visual surveillance of object motion and behaviors, Syst. Man Cybern. Part C Appl. Rev. IEEE Trans. 34 (3) (2004) 334–352, http://dx.doi.org/10.1109/TSMCC.2004.829274.
[23] T. Teixeira, D. Lymberopoulos, E. Culurciello, Y. Aloimonos, A. Savvides, A light-weight camera sensor network operating on symbolic information, Proceedings of the 1st Workshop on Distributed Smart Cameras, (2006).
[24] V. Jelicic, M. Magno, D. Brunelli, V. Bilas, L. Benini, Benefits of wake-up radio in energy-efficient multi-modal surveillance wireless sensor network, Sens. J. IEEE 14 (9) (2014) 3210–3220, http://dx.doi.org/10.1109/JSEN.2014.2326799.
[25] M. Magno, F. Tombari, D. Brunelli, L. Di Stefano, L. Benini, Multimodal video analysis on self-powered resource-limited wireless smart camera, Emerg. Sel. Topics Circuits Syst IEEE J. 3 (2) (2013) 223–235, http://dx.doi.org/10.1109/JETCAS.2013.2256833.
[26] S.K. Nayar, D.C. Sims, M. Fridberg, Towards self-powered cameras, Proceedings of the IEEE International Conference on Computational Photography (ICCP), (2015), pp. 1–10, http://dx.doi.org/10.1109/ICCPHOT.2015.7168377.
[27] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, D. Culler, Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments, Proceedings of the 5th International Conference on Information Processing in Sensor Networks, IPSN '06, ACM, New York, NY, USA, 2006, pp. 407–415, http://dx.doi.org/10.1145/1127777.1127839.
[28] C. Hartung, R. Han, C. Seielstad, S. Holbrook, Firewxnet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments, Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, MobiSys '06, ACM, New York, NY, USA, 2006, pp. 28–41, http://dx.doi.org/10.1145/1134680.1134685.
[29] S. Mohseni, R. Hassan, A. Patel, R. Razali, Comparative review study of reactive and proactive routing protocols in manets, Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST), IEEE, 2010, pp. 304–309.
[30] T. Zahariadis, K. Petrakou, S. Voliotis, Enabling QoS in visual sensor networks, Proceedings of the 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, (2006), pp. 327–330, http://dx.doi.org/10.1109/ELMAR.2006.329577.
[31] Y. Chang, X. Jia, Rate-adaptive broadcast routing and scheduling for video streaming in wireless mesh networks, Proceedings of the 23rd International Conference on Computer Communication and Networks (ICCCN), IEEE, 2014, pp. 1–8.
[32] K. Lee, S. Lee, S. Lee, Optimization of delay-constrained video transmission for ad hoc surveillance, IEEE Trans. Veh. Technol. 63 (4) (2014) 1855–1869, http://dx.doi.org/10.1109/TVT.2013.2289393.
[33] Y. He, L. Guan, Optimal resource allocation for video communication over distributed systems, Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), (2009), pp. 1414–1423, http://dx.doi.org/10.1109/ICME.2009.5202768.
[34] A. Kandhalu, A. Rowe, R. Rajkumar, C. Huang, C.-C. Yeh, Real-time video surveillance over IEEE 802.11 mesh networks, Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), (2009), pp. 205–214, http://dx.doi.org/10.1109/RTAS.2009.38.
[35] T. Zahariadis, S. Voliotis, Open issues in wireless visual sensor networking, Proceedings of the 14th International Workshop on Systems, Signals and Image Processing and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. IEEE, 2007, pp. 335–338.
[36] F. Licandro, A. Lombardo, G. Schembra, Multipath routing and rate-controlled video encoding in wireless video surveillance networks, Multimed. Syst. 14 (3) (2008) 155–165.
[37] A. Papalambrou, P. Soufrilas, A.G. Voyiatzis, D.N. Serpanos, A Secure DTN-based Smart Camera Surveillance System, Proceedings of the Workshop on Embedded Systems Security, WESS '11, ACM, New York, NY, USA, 2011, pp. 3:1–3:4, http://dx.doi.org/10.1145/2072274.2072277.
[38] A.M. Zungeru, L.-M. Ang, S.R.S. Prabaharan, K.P. Seng, Ant based routing protocol for visual sensors, in: A.A. Manaf, A. Zeki, M. Zamani, S. Chuprat, E. El-Qawasmeh (Eds.), Informatics Engineering and Information Science, no. 252, Communications in Computer and Information Science, Springer, Berlin Heidelberg, 2011, pp. 250–264 URL http://link.springer.com/chapter/10.1007/978-3-642-25453-6_23.
[39] J. Bae, R.M. Voyles, Wireless video sensor networks over bluetooth for a team of urban search and rescue robots. ICWN, Citeseer, 2006, pp. 409–415.