Multimed Tools Appl (2018) 77:21417–21443
https://doi.org/10.1007/s11042-017-5565-1

Power profiling of multimedia sensor node with name-based segment streaming

Adolph Seema[1] · Tejas Shah[1] · Lukas Schwoebel[2] · Yu Liu[3] · Martin Reisslein[1]

Received: 20 April 2017 / Revised: 7 November 2017 / Accepted: 21 December 2017/
Published online: 10 January 2018

Abstract Multimedia streaming from miniaturized sensors is attractive for a wide range of web-based applications, including surveillance and Internet of Things (IoT) applications. This paper profiles the power consumption in a wireless video sensor node. We compare the power consumption of video streaming frameworks based on a manifest file, such as the Hypertext Transfer Protocol (HTTP) Live Streaming (HLS), with a Wireless Video Sensor Network Platform compatible Dynamic Adaptive Streaming over HTTP (WVSNP-DASH) framework. The WVSNP-DASH framework is based on independently playable video segments that convey the metadata required for playback in their names (and do not require a manifest file). The power consumption components of the video capture and storage

Parts of this work were conducted while Y. Liu visited Arizona State University, Tempe, sponsored by the China Scholarship Council.

✉ Martin Reisslein
reisslein@asu.edu

Adolph Seema
adolph.seema@asu.edu

Tejas Shah
tpshah1@asu.edu

Lukas Schwoebel
lukeschwoebel@gmail.com

Yu Liu
yliu581@asu.edu; liuy@bupt.edu.cn

[1] School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706, USA

[2] Favendo GmbH, Gena, Germany

[3] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China

pipeline are evaluated. The presented extensive power profiling measurements provide real-world empirical data on architectural design decisions for multimedia sensor nodes suitable for IoT applications. Our measurement results indicate that the name-based WVSNP-DASH framework is well suited for flexible low-power web-based video streaming from miniaturized sensors.

Keywords Dynamic adaptive HTTP streaming (DASH) · HTTP live streaming (HLS) · Power measurement · Video streaming · Wireless video sensor

1 Introduction

1.1 Motivation: low-power video streaming from sensors

Wireless sensor networks and the emerging Internet of Things (IoT) increasingly involve applications that require multimedia streaming [12, 14, 32, 39, 43, 46, 66, 67, 80, 99, 101], e.g., the capturing and real-time web-based network delivery of a surveillance video feed [92]. Low-power operation is a key requirement for video streaming from miniaturized sensors, that are often resource-constrained and rely on limited battery power [15, 42, 60, 63, 68, 70]. Web-based video streaming has evolved in recent years toward the streaming of segments with a fixed video playback duration over the Hypertext Transfer Protocol (HTTP) [3, 13, 45, 65, 79, 88, 93, 104]. The popular HTTP Live Streaming (HLS) [62] and the Motion Picture Experts Group's Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [85, 89] are based on a manifest file that conveys the video meta information required for playback. The manifest file based streaming introduces dependencies between the individual video segments and the manifest file that need to be maintained by the video server node, e.g., a miniaturized video sensor node. Maintaining these dependencies may be quite demanding for power-constrained sensor nodes.

In an attempt to simplify video streaming from power-constrained sensor nodes, a name-based video segment streaming framework, the co-called Wireless Video Sensor Network Platform compatible Dynamic Adaptive Streaming over HTTP (WVSNP-DASH) framework, has recently been introduced in [78]. WVSNP-DASH is not a modification of the existing MPEG-DASH [85, 89] framework. Instead, WVSNP-DASH is a fundamentally different approach to achieve the dynamic adaptive streaming over HTTP (DASH) functionality in resource constrained sensor networks. We note that the acronym "DASH" is sometimes used only in the context of the MPEG; however, we interpret "DASH" more broadly to refer to "dynamic adaptive streaming over HTTP". Alternatively, the acronym "HAS" for "HTTP adaptive streaming" could be used to refer to the general adaptive streaming paradigm. However, we feel that the terminology "dynamic adaptive streaming over HTTP (DASH)" is more descriptive and more widely recognized and therefore use "DASH" for our WVSNP-DASH.

The WVSNP-DASH segment naming syntax conveys elementary metadata and thus obviates the need for maintaining dependencies to a manifest file. Rather, the WVSNP-DASH video segments are independently playable video files that facilitate flexible low-power streaming from miniaturized sensor nodes. We briefly note that the WVSNP-DASH naming syntax is designed to convey all the meta-data necessary for dynamic adaptive streaming. More specifically, the naming syntax conveys the file name to uniquely identify the video stream (i.e., sequence of video segments), the highest video quality available for the stream, the video quality of the considered segment, the playback mode (LIVE

or VOD), the total number of segments available for the video stream, the index (number) of the considered segment, and the video container format. This meta data is sufficient to enable a client to retrieve and play the video segments following the DASH principles [78]. A main motivation for this study is to investigate the power consumption characteristics of the relatively new name-based streaming framework [78] in comparison to the conventional manifest file based streaming framework.

1.2 Contributions

This article presents a rigorous empirical measurement study of power consumption for video streaming from miniaturized multimedia sensor nodes. Our first main contribution is a comparison of the power consumption of two frameworks for capturing, storing, transmitting, and playing back web-based video segments: Manifest file based video streaming vs. segment name based streaming. Specifically, we conduct measurements to compare the power consumption of the commercial manifest file based HLS framework against a prototype of the name-based WVSNP-DASH framework. Our measurement results indicate that WVSNP-DASH consumes substantially less power than HLS for video capture and LIVE streaming. The name-based WVSNP-DASH segment streaming obviates the maintenance of the manifest to segment file dependencies in HLS, which increase the sensor node power consumption. For video on demand (VOD) streaming of already captured and stored video segments, WVSNP-DASH and HLS have comparable power consumption since the manifest file to segment dependencies have been pre-configured during the video capture.

Our second main contribution is a detailed set of measurements for power profiling the major power consumption components of video capture and storage at a multimedia sensor node. We profile the main sensor node design aspects that influence the power consumption, including the video library tools (FFmpeg [21] vs. GStreamer [28]), the interface used in the data movement pipeline from camera to sensor node board [USB vs. Camera Serial Interface (CSI)], as well as hardware (HW) acceleration of video coding versus software (SW) only encoding. We find that the GStreamer video library tools consume less power than FFmpeg tools. Also, the CSI interface and HW encoding are preferable for low-power sensor nodes. Overall, our measurements provide an empirical power consumption reference for designers of wireless multimedia sensor nodes.

2 Background and related work

Numerous studies have noted the critical role that power plays in sensor network design and operation [8, 17, 18, 25, 41, 48, 51, 55, 61, 69, 100]. However, relatively few studies have examined the power measurement and management in sensor nodes [37, 72, 102]. In this section, we first briefly summarize the background on power measurements and then review existing sensor node power measurement studies.

2.1 Sensor node power measurement methods

The energy consumption of an electrical device is calculated by the product of current I, voltage V, and time t. Voltage and time can be measured directly. However measuring the current is challenging and the existing current measurement approaches present several tradeoffs and limitations that we summarize in the following. The existing in-system power monitoring tools include *PowerTop* [2], *powerstat* [23], software libraries, such as

PowerAPI [9, 58, 59] for process specific monitoring, and kernel specific libraries, such as *powerman* [22] and *powerscripts* [5]. These in-system methods affect the device under test (DUT) since they are power-consuming processes within the node. Also, these methods require that the operating system has been booted before they can commence measuring. Thus, these in-system methods cannot measure the node power consumption during the boot loader stages and the power-up part of the node.

The survey [35] reviews different approaches for measuring energy consumption in wireless communication devices. One popular approach is to place a *shunt resistor* in series with the total load circuit [33]. The current draw is the same across the whole circuit which implies a current draw of V/R across the shunt resistor. The shunt resistor method is simple; however, if the voltage over the resistor gets too large, the device under test may malfunction. This malfunction is avoided by using a very low shunt resistor value. The low shunt resistor value, however, makes the measurement of highly dynamic signals difficult. The higher the dynamic range of the current, the lower the accuracy of the low measured currents. The accuracy for low currents can be improved with the *Voltage to Frequency Conversion* method [47], which connects the shunt resistor to a voltage to frequency conversion block. Highly dynamic low as well as high currents can then be measured with the same high accuracy from the conversion block.

Another current measurement technique is the *inductor method*, usually used in current clamps for heavy engineering tools. Current is determined by sampling the voltage induced in the clamp inductor by the electric field around the wire supplying the load circuit. The inductor technique supports high sampling rates. The drawbacks of the inductor method include a required calibration after each measurement and noise susceptibility.

The *Coulomb counter* method uses two capacitors, which are charged and discharged in turn [57]. The capacity of the capacitor is used to calculate the discharge time. The temporal resolution of the Coulomb counter method depends on the current draw. Since the current drawn typically varies, the temporal resolution also varies. Low current draws result in low frequency, and hence low temporal resolution.

2.2 Sensor node power measurement studies

Low-cost power measurement techniques based on the shunt resistor and the current clamp for wireless sensor networks have been examined in [54]. A Sensor Node Management Device (SNMD), i.e., a wire-based testbed infrastructure, based on a shunt resistor has been studied in [34] for a wireless sensor node. A Scalable Power Observation Tool (SPOT) based on a shunt transistor has been introduced in [40].

Complementary to our measurement study, the energy efficiency of HW accelerated cryptography modules on sensor nodes has been examined in [31] with the SANDbed testbed [34] equipped with Sensor Node Management Devices (SNMDs) [31, 34]. The measurements revealed about 76 % energy savings with a VaultIC420 HW module compared to using only SW. The study [52] compared power measurements with the internal power meter of a Stargate sensor node platform with multimeter measurements. The energy efficiency of transmitting uncompressed raw video data vs. compressed video data was examined in [10], while the energy efficiency of a specific motion detection application was examined in [44, 53].

The study [86] examined the energy consumption for ambient monitoring of temperature, humidity, and light levels in a sensor node. In [30], the Avrora simulation tool [90], has been used to validate energy measurement from a SANDbed wireless sensor network testbed. In [84, 91], the energy consumption of wireless sensor node transmissions over WiFi and the

TCP/UDP transmission protocols has been modeled and verified through measurements. The energy saving effect of introducing multiple tiers in a wireless sensor network has been studied in [64]. The energy consumption of wireless sensor networks specific to agriculture applications has been investigated in [6, 27, 38].

The study [103] has examined MPEG-DASH client network scheduling by measuring client power consumption over fourth generation Long Term Evolution networks (4G LTE). The study [26] proposed an energy-efficient adaptive streaming algorithm for an MPEG-DASH client that is connected to multiple networks, such as LTE and WiFi. The proposed algorithm exploits the multiple networks to find opportunities for low-power streaming. Both studies [103] and [26] are complementary to the present study in that they focus on MPEG-DASH and specifically the client's adaptive segment fetching algorithm to reduce client power consumption. In contrast, we focus on the server node/network side and consider HLS and WVSNP-DASH. To the best of our knowledge, the present study is the first to thoroughly compare power measurements of manifest file based HLS and segment file name based WVSNP-DASH. A preliminary comparison that considered only one video quality and client operating system (OS) for one LIVE streaming segment duration and different VOD segment durations was included in [78]. In contrast, we thoroughly examine video capture, LIVE video streaming, and VOD streaming for different segment lengths in combination with different video qualities and client OSs. Furthermore, we thoroughly examine the power consumption characteristics of different video library tools, camera interfaces, as well as HW and SW video encoding. We note for completeness that analytical models of sensor node power consumption have been developed in [16, 73–76].

3 Evaluation setup

3.1 Power measurement setup

Figure 1 illustrates our power measurement setup, which is based on the inductor power measurement method. To measure current, a 10 μA resolution current clamp is placed around the wire that powers the board. Voltage is measured by connecting the oscilloscope probe to the 5 V jack on the board. The oscilloscope probe and the current clamp are connected to a digital oscilloscope with up to 100 MHz bandwidth, up to 1 giga samples/s sampling rate, and 8-bit (12-bit enhanced resolution). We adopted the described inductor power measurement setup and did not use a plug-in power meter because we could not find a meter for less than US$ 500 at the current resolutions that we expected for low-power circuits. Additionally, we adopted the measurement approach with current clamp and oscilloscope because of its convenient built-in data collection capabilities. The data output capabilities of the oscilloscope we used are rarely available on plug-in power meters at reasonable cost. The oscilloscope provides reliable sampled data which can be output to ".csv" files in different sampling resolutions. This is hard to find at USD$ 500 or less in plug-in power meters. We double checked the current clamp measurements with a 24-bit ADC open meter tool, the so-called Mooshimeter [56], which is a low-cost high-accuracy meter that simultaneously logs current and voltage.

3.2 Multimedia sensor (server) node setup

The server node was implemented with an *NXP i.MX6 ARM Cortex-A9*, 1.2 GHz Quad core, 2 GB node development board. The server node captured video with a USB webcam or a

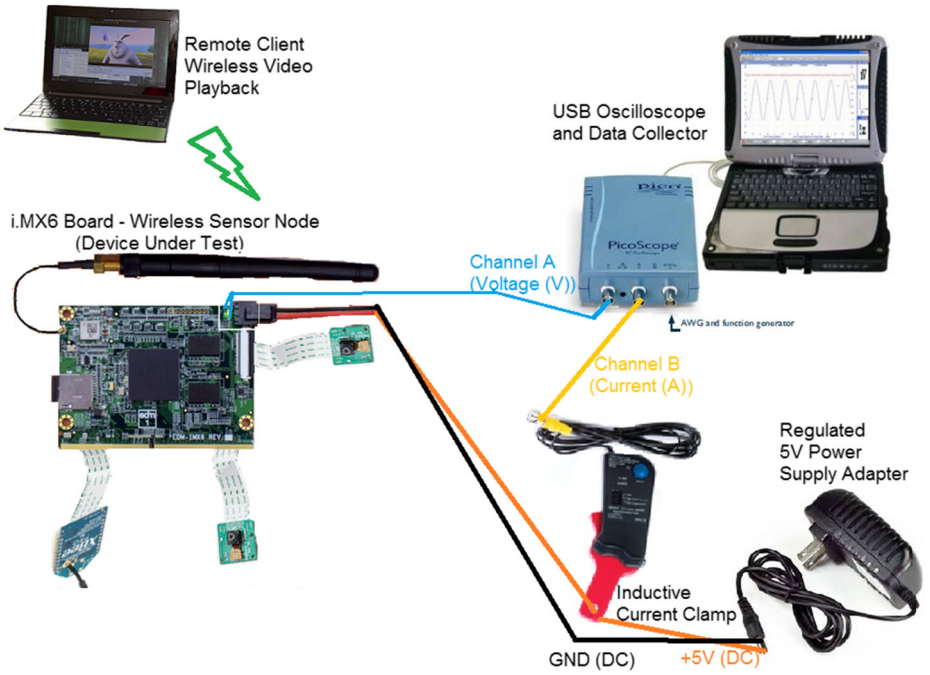

Fig. 1 Illustration of power measurement setup: an inductor based current clamp with 10 μA resolution measures the current drawn by the server node board. The current and the voltage measurements are logged by an oscilloscope and laptop computer

Camera Serial Interface (CSI) attached Wandcam [19]. The captured video segments were served over WiFi or wireline Ethernet, via a streamlined low-footprint mongoose HTTP server.

3.3 Video client setup

A WVSNP-DASH Player (WDP) prototype [78] was used to play back streamed WVSNP-DASH segment files. Streamed HLS video segments were played back with the JWPlayer 6 with the HLSProvider plug-in. The Google Chrome (version 32) web browser was used as the default client display outlet. The clients ran on a *Ubuntu 13.10* 64 bit, *Dell OptiPlex 360* with *Intel Core2 Duo E7300* 2.66 GHz CPU and 2 GB RAM. The evaluations were also run with a client operating on a *Macbook Air Mid 2012* with *i5-3427U* 1.8 GHz CPU and 4 GB RAM. For additional checks, the same Macbook Air was rebooted into a *Windows 7* 64 bit client.

The streamed video was played back using the Application Programming Interface (API) for the HTML5 File System (HTML5 FS), HTML5 Canvas, and the HTML5 Video Element. At the time when this study was designed, the HTML5 FS was emerging as a seamless uniform cross-platform standard. We acknowledge that the Media Source Extensions (MSEs), which have recently been developed by the World Wide Web Consortium (W3C) [96], can be used to form an alternative API. Power profiling of the MSEs is an interesting direction for future research.

3.4 Test video

A typical surveillance video was captured on the Arizona State University (ASU) campus. The 10 minute ASU video (without audio) shows the scenery and everyday outdoor activity on the ASU campus. Video on demand (VOD) and LIVE streaming used the ASU video. Generally, LIVE streaming refers to the capture of a live event scene, e.g., a sporting or cultural event, with a video camera and the real-time video encoding and network transmission to receivers that wish to watch the live event on their video receivers. In contrast, VOD streaming refers to the network transmission of video that has been captured and encoded a priori, e.g., a documentary on a scientific topic, or a pre-recorded lecture video. For the LIVE video capture and streaming, the node camera was pointed at the full-screen display of the pre-recorded video while measurements were conducted in real time. Positioning and focusing the node camera to see only the full-screen video display made the LIVE video essentially identical to the pre-recoded video. We consider a "SMALL" video quality with 320×180 pixel resolution, 150 kb/s bit rate, and 15 frames/s, and a "BIG" video quality with 640×360 pixel resolution, 500 kb/s, and 25 frames/s. For HLS streaming, the video was segmented into MPEG2 Transport Stream (M2TS) [83] segments, which are the required video container format for HLS. For WVSNP-DASH, which flexibly allows arbitrary video container formats, the video was segmented into independent MP4 segments. The segments were created for 2, 5, 10, and 15 second segment lengths.

3.5 Measurement procedure

Initially, we conducted measurements starting from the pre-boot time through the end of the video capture, as illustrated for a typical measurement run in Fig. 2. We observed four major phases (stages): pre-boot, boot, idle, and video capture/streaming. We verified that

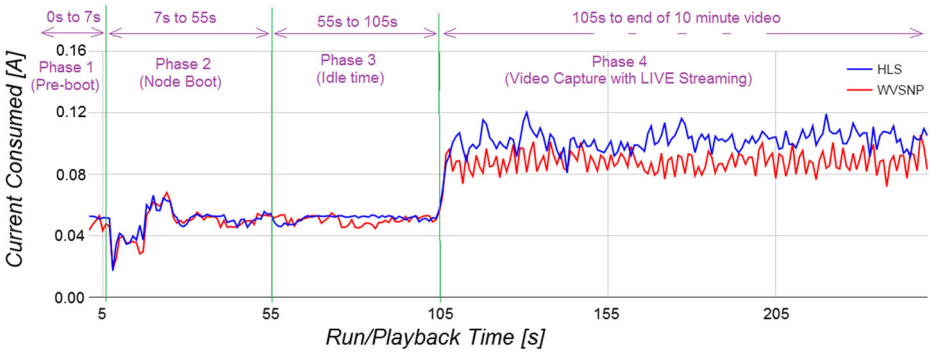

Fig. 2 Smoothed plot of server node current as a function of time for SMALL video capture and LIVE transmission of 2 s segments over WiFi with Ubuntu client (see second row in Table 2). We observe four temporal phases (stages): Pre-boot (up to 7 s), boot (7–55 s), idle time (55–105 s), and capture with LIVE streaming (from 105 s onwards to end of 10 minute video; only the first 150 s of video are plotted). The first three stages are characteristic of the test board and are essentially identical for all measurement runs. The averages reported in this paper were obtained for the video capture/streaming phase, which is characteristic of the examined video capture and streaming approaches. The current consumption is highly variable over time (see SD values in Table 2) and was smoothed for plotting to avoid visual clutter. The current consumption tends to slightly decrease over the course of the capture and streaming so as to give the averages reported in Table 2, i.e., 80.5 mA for WVSNP-DASH and 90 mA for HLS

the measurement data were essentially the same for the first three stages, which are characteristic of the node booting up. For all actual measurements that are reported in this paper, we collected measurements starting from the end of the idle time (stage 3) through the end of the capture of the ten minute test video and the associated streaming. That is, all reported averages correspond to the video capture and streaming phase and do not consider the node boot-up.

For each particular experimental condition (as per the experimental design specified in Section 3.6), the oscilloscope SW tool sampled more than 14,000 data points from a ten minute measurement run. Our measurements indicated that the voltage was always constant very close to 5 V. We report therefore only the measured current in milli Amperes (mA). In particular, we randomly sampled one current measurement per a 10/15 second period, which corresponds to ten frame periods of the SMALL video quality. The reported averages and standard deviations (SDs) are based on the resulting 900 samples from a 10 minute measurement run. Differences between the independently measured experimental conditions were analyzed with the two-sided independent sample t-test [94]. Unless otherwise noted, all discussed differences were found to be statistically significant at the $p < 0.001$ level.

3.6 Experimental design

Throughout, this study considers consumed electrical current [in milli Ampere] as the main dependent (response) variable. As noted in the preceding subsection, the server node circuit board voltage was essentially constant at 5 V. Thus, the consumed electrical current is indicative of the power consumption (see Section 2.1). We consider two main types of factors (varied independent variables), namely the streaming framework and the node data path components. The streaming framework factor is investigated in Section 4 by comparing the WVSNP-DASH name-based segment streaming framework [78] with the HLS manifest file based streaming framework [62] while keeping all the node data path component factors constant. In the investigation of the streaming framework factor, we consider a few common auxiliary (secondary) factors, namely, the video segment length (2, 5, or 10 seconds), the video quality (SMALL video with low quality and BIG video with high quality), the network link (wireless WiFi vs. wired Ethernet), and client operating system (Linux Ubuntu, Windows, or Mac).

Subsequently, in Section 5, we fix the streaming framework factor to the WVSNP-DASH framework and fix the auxiliary video quality factor to the BIG video and investigate the server node data path factors. (The auxiliary factors network link and client operating system are not relevant for this server node investigation.) In particular, we investigate the following server node data path factors: video library tool (FFmpeg vs. GStreamer), camera interface data path (USB vs. Camera Serial Interface), and video encoder (H.264, JPEG, or MPEG4 implemented in hardware or software). (For the streaming framework comparison in Section 4 these server node data path factors were set to FFmpeg, USB, and H.264 in software.) In addition, we consider the video segment length (2, 5, or 10 seconds) as an auxiliary factor in data handling along the node data path.

3.7 Threats to validity

As an empirical quantitative study, our study on power profiling of multimedia sensor nodes faces a range of threats to validity [20, 71, 97, 98]. This section outlines these threats to validity and steps undertaken to mitigate the threats. In order to address construct validity [98], i.e., that the current measurements really represent the power consumed in the

multimedia sensor node, we verified that the voltage at the sensor node board stayed at a constant 5 V for the entire durations of the measurement runs. More specifically, the voltage was supplied by an industrially certified consumer power adapter with a nominal voltage setting of 5 V and an accuracy specification of $\pm$ (0.05 % + 2 mV). The corresponding voltage uncertainty is 5 V $\cdot$(0.05 %) + 2 mV = 4.5 mV.

Threats to internal validity [98] can be other factors that were not explicitly measured but could influence the outcomes of the experiments. In our study, the environmental conditions during the measurements, such as temperature and wireless interference, could influence the outcomes of the experiments. In order to mitigate the influence of the environmental conditions, we conducted all measurements in the same environmental setting and kept all the characteristics of the environment essentially constant, e.g., we conducted all measurements in the same temperature controlled lab room. We controlled the sensor node and, if applicable, the streaming client so that only the processes related to the experiment were running during the data collection. The WiFi network setup was deliberately set to the peer-to-peer (P2P) configuration to avoid effects due to access point negotiations with other possible clients on the network.

External validity [98] addresses the generalization of the experimental results to other settings. In order to ensure external validity, we considered the widely employed HLS approach as representative benchmark for manifest file based streaming. Also, we considered the full range of typically considered segment lengths ranging from 2 s to 10 s and compared with progressive streaming of the full video. We also considered two video quality levels as well as wireless and wired network transmission to clients with different operating systems. Moreover, the node data path investigations considered the two dominant multimedia tool sets (FFmpeg and GStreamer) as well as the two main commercially employed camera interfaces (USB and CSI) as well as a range of video encoding approaches, both with software and hardware implementation. A limitation of our study is that we did not conduct measurements for all possible combinations of all considered factors. In order to keep this research project feasible and to be able to concisely present insightful results in a journal article, we conducted measurements for selected combinations of factors that are of high interest for multimedia systems and provide interesting contrasting insights. Nevertheless, we believe that the measurement results are overall applicable to a wide set of practical multimedia systems.

4 Framework power profiling: WVSNP-DASH vs. HLS

This section compares the current consumption of WVSNP-DASH and HLS for video capture (without streaming), video capture and LIVE streaming, as well as VOD streaming (of previously captured segments). For the framework comparison in this section, the video was captured with the USB camera and the FFmpeg video library tools. The video was H.264 encoded in software (with the x264 software library of FFmpeg).

4.1 Video capture

4.1.1 WVSNP-DASH vs. HLS comparison

For an initial WVSNP-DASH vs. HLS comparison, we examine their video segment capture in isolation. Table 1 reports the consumed currents for only the video capture at the server (camera) node, without any networking or transmission. We observe from Table 1

Table 1 Averages and standard deviations (SD) of current (mA) consumed by multimedia server node while capturing WVSNP-DASH versus HLS 2 s, 5 s, and 10 s BIG video segments

	2 seconds				5 seconds				10 seconds			
	WVSNP		HLS		WVSNP		HLS		WVSNP		HLS	
	Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD
Current [mA]	77.07	17.00	79.84	17.00	80.66	16.41	79.77	16.41	81.56	16.38	83.79	16.48

that WVSNP-DASH consumes less current than HLS for the 2 and 10 s segments (statistically significant at $p < 0.001$ levels); while consuming very slightly more current than HLS for the 5 segments (this difference is not statistically significant, $p = 0.25$). For the interpretation of these current consumption measurements, it is important to keep in mind that the examined WVSNP-DASH prototype system and the commercial HLS system utilize FFmpeg differently for segment capture. The commercial HLS system captures video segments with an optimized built-in (native) HLS function of FFmpeg, which achieves highly efficient SW-based capture. In contrast, the WVSNP-DASH prototype captures video segments by interpreting a bash script at run time. The script newly invokes FFmpeg for the capture of each individual segment. Thus, the WVSNP-DASH prototype consumed extra current for each context switch of launching an FFmpeg process, encoding and storing the video segment, and then shutting down the FFmpeg process for each individual video segment capture. On the other hand, the commercial HLS system invoked FFmpeg only once at the start of the video stream capture and captured the remaining segments with the same optimized FFmpeg process (from the original invocation context). Despite the inefficient script loop that opens and closes FFmpeg for every segment, the examined WVSNP-DASH prototype tends to consume less power than HLS (except for the 5 s segments where WVSNP-DASH and HLS have essentially equivalent current consumption). This indicates that when WVSNP-DASH were to employ an optimized native capture application (which HLS already uses), WVSNP-DASH would significantly reduce the current consumption compared to HLS.

4.1.2 Stream segmenter in HLS vs. independent WVSNP-DASH segments

Figures 3 and 4 illustrate the conceptual structure of the video segment capture-store-stream processing flows of WVSNP-DASH and HLS. Note in Fig. 4 that HLS has a *Stream*

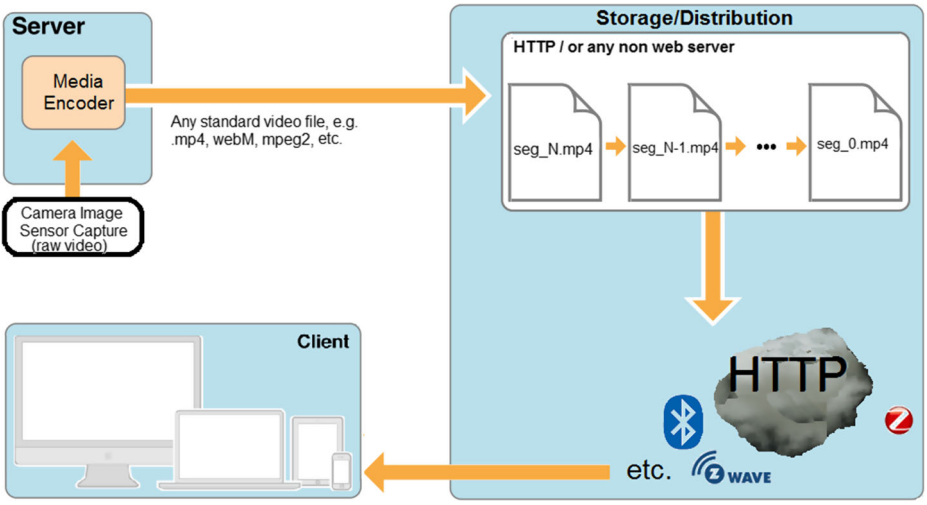

Fig. 3 Conceptual structure of WVSNP-DASH capture-store-stream flow: The sensor (server) node stores the captured video in independently playable video segments that convey essential meta data through their file names

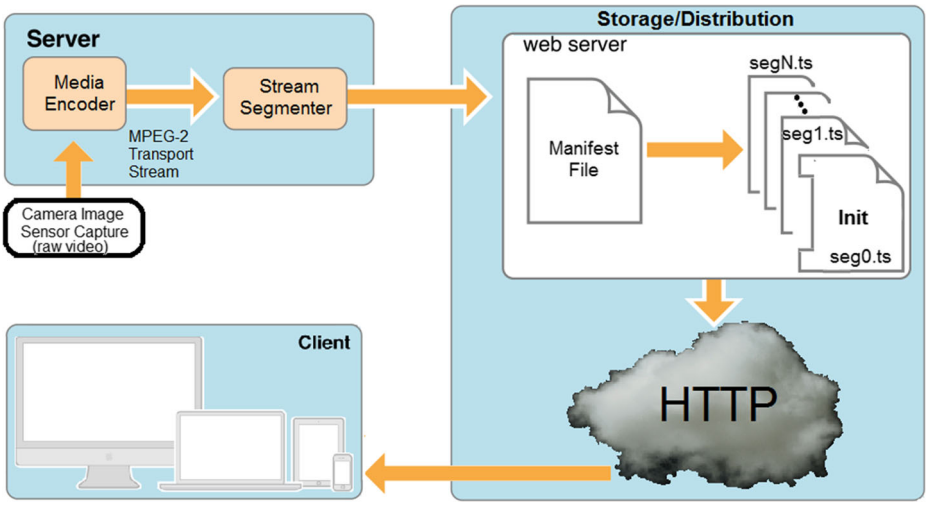

Fig. 4 Conceptual structure of HLS capture-store-stream flow: The stream segmenter creates the video segments in dependency to the manifest file, which conveys metadata for playback

Segmenter stage that manages the manifest file required in HLS. The stream segmenter continuously re-reads and updates the manifest file during the video segment capture [7, 83, 95]. More specifically, the segmenter stage keeps track of the segments as they are created relative to the initialization segment file, i.e., the segmenter has to buffer information about the initialization segment file and update it as dependent segments are created. (HLS initialization segments have extra file header information and other metadata that help players or browsers decode the complete video stream [82, 83, 95].) The segmenter also keeps track of the manifest file and updates the manifest file every "refresh" time, which is set at invocation. Moreover, the segmenter keeps track of all segments in relation to each other. The stream segmenter stage thus adds to the workload during video segment capture.

WVSNP-DASH does not have a stream segmenter stage, see Fig. 3. More specifically, WVSNP-DASH only has information provided at invocation for the type of segment to create. This information is identical for each segment, except for the incrementing index. Thus, there is no "segmenting" closed-loop control needed for reopening and updating files (as done in the HLS segmenter). This simple capture operation allows WVSNP-DASH to capture video segments with the same or somewhat lower current consumption than HLS despite the inefficient WVSNP-DASH prototype operation (with script-based FFmpeg initializations for each segment).

4.1.3 Impact of segment length

Table 1 reveals another interesting result: Capturing short two-second segments does not necessarily result in higher current cost as one might expect due to the more frequent file operations inherent in handling more files for the same captured video duration. In particular, the measurement results in Table 1 indicate that shorter segments generally tend to reduce the consumed current for both WVSNP-DASH and HLS. (All segment length comparisons are statistically significant at the $p < 0.001$ level; except the WVSNP-DASH 5 s vs. 10 s and the HLS 2 s vs. 5 s differences are not statistically significant.) These results

indicate that for most examined comparison scenarios, keeping the large file for a long video segment open and updating the file consumes slightly more power than working with small files for short segments that do not have to be open for too long. More specifically, the results indicate that the current cost of opening and closing small files quickly and therefore swapping files for short segments in and out of dynamic memory is slightly lower than the cost for keeping memory, cache, and SD card synchronized while the large file for a long segment is open and being appended to. The longer the video segment, the longer the file will need to be open and appended to during capture.

4.2 Video capture with LIVE video streaming

4.2.1 LIVE streaming vs. capture

Table 2 reports the measured current consumption for the simultaneous capture and LIVE streaming of WVSNP-DASH and HLS video segments. An initial comparison between Tables 1 and 2 indicates that simultaneous video capture and LIVE streaming results in significantly higher current draw and thus higher power consumption than video capture alone. This is because with simultaneous video capture and LIVE streaming, the server node captures, encodes, stores, and serves (transmits) the video segments at the same time; whereas, for video capture, the server node only captures, encodes, and stores the video segments.

4.2.2 WVSNP-DASH vs. HLS comparison

We observe from Table 2 that WVSNP-DASH tends to have generally lower sensor node power consumption than HLS for the considered link and client OS scenarios and segment lengths. (All comparisons are statistically significant at the $p < 0.001$ level; except the 5 s Eth. Win., BIG WVSNP vs. HLS comparison is only significant at the $p = 0.003$ level, and the 5 s WiFi, Ub., BIG WVSNP vs. HLS comparison is not statistically significant.) The power savings with the WVSNP-DASH framework are achieved through the avoidance of a manifest file. Manifest file based streaming frameworks, such as HLS and MPEG-DASH, must continuously update the manifest file during the video capture and LIVE playback [7, 83, 95]. In particular, for LIVE streaming, the manifest file needs to be continuously updated and re-transmitted with every newly generated segment. Additional processing to

Table 2 Averages and standard deviations (SD) of current (mA) consumed by the server node while capturing and LIVE streaming of 2 s and 5 s WVSNP-DASH vs. HLS segments

	2 seconds				5 seconds			
Video type	WVSNP		HLS		WVSNP		HLS	
Link, Client OS, Vid. Qu.	Avg	SD	Avg	SD	Avg	SD	Avg	SD
WiFi, Ub., BIG	85.05	24.50	106.24	30.82	98.30	28.96	95.80	29.56
WiFi, Ub., SM.	80.52	20.66	90.04	24.80	87.58	23.33	92.84	25.34
Eth., Win., BIG	.	.	.	.	100.17	27.98	104.23	29.98
Eth., Mac, BIG	.	.	.	.	91.59	26.04	101.75	29.73

create special subsequent segments different from the initialization segment adds to the power consumption of HLS for LIVE video. Another reason is that LIVE HLS requires an additional stream segmentation stage (see Fig. 4) in the capture stage, which needs to be fully active during LIVE playback consuming more power than WVSNP-DASH which does not have a stream segmentation stage (see Fig. 3).

4.2.3 Impact of video quality

We observe from Table 2 that SMALL quality segments (see Section 3.4 for definition of SMALL and BIG quality segments) result generally in lower current draw than BIG quality segments due to their lower data rate and size.

4.2.4 Impact of segment length

We furthermore observe from Table 2 that for WVSNP-DASH, capturing and LIVE streaming short 2 s segments consume less current than 5 s segments. (The segment length comparisons for WVSNP-DASH are statistically significant at the $p < 0.001$ level; the BIG HLS segments do not follow this trend, while for the SMALL HLS segments, the length comparison is statistically significant only at the $p = 0.018$ level.) This result is consistent with the observations for current consumption for capturing different segment lengths (see Section 4.1.3). Similar to the capturing of long video segments in large files, the simultaneous capturing and LIVE streaming of long video segments requires sensor node operations with large files. In particular, for long video segments, large files need to be packaged, saved, and streamed, requiring file open and close operations for large files as well as writing to and reading from large files. The more extensive file operations required for simultaneous capturing and streaming lead generally to more pronounced power reductions with short segments in Table 2 compared to the power reductions for only capturing in Table 1. Overall, the current consumption results in Table 2 support the capturing and streaming of LIVE video with short 2 s segments in WVSNP-DASH (while the segment length results for HLS are inconclusive). Short video segments are generally preferable for DASH LIVE video streaming due to the critical temporal aspect of LIVE video applications, i.e., many LIVE video applications prefer to have as little time lag as possible between the live events captured in the video and the corresponding playback of the video segment containing the events on the client side.

4.3 VOD Video Streaming

4.3.1 General WVSNP-DASH vs. HLS comparison

Table 3 compares the current consumed by the server node while streaming WVSNP-DASH vs. HLS for 2 s, 5 s, and 10 s Video On Demand (VOD) segments. We observe from Table 3 that the results are overall mixed: the WiFi streaming scenarios do not exhibit a clear trend, either WVSNP-DASH or HLS consume less current, depending on the specific scenario. For the Ethernet scenarios, HLS tends to consume less current than WVSNP-DASH. For VOD streaming, all video segments are already available at the start of the streaming, i.e., all initialization files and manifest files have been precomputed for HLS (during the video segment capture). For VOD, the manifest files and segments are static and the indices do not need to be continuously updated (unlike for normal HLS LIVE streaming, where the manifest file needs to be continuously updated and re-transmitted with every new generated

Table 3 Averages and standard deviations (SD) of current (mA) consumed by the multimedia (server) sensor node while streaming WVSNP-DASH versus HLS for 2 s, 5 s, and 10 s VOD segments

	2 seconds				5 seconds				10 seconds			
	WVSNP		HLS		WVSNP		HLS		WVSNP		HLS	
Link, Client OS, Vid. Qu.	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD
WiFi, Mac, BIG	38.23	9.77	46.90	8.79	36.30	9.24	45.57	8.98	36.36	8.88	34.80	8.86
WiFi, Ub., SM.	42.00	9.51	39.70	8.64	40.39	9.09	43.31	8.58	37.83	8.76	38.95	8.17
WiFi, Win., BIG	40.23	9.84	37.42	9.36	39.71	9.42	35.14	8.79	37.81	9.01	40.04	9.34
Eth., Win., BIG	41.26	9.98	40.32	9.28	39.89	9.63	39.81	8.86	38.32	9.55	38.03	8.92
Eth., Mac, BIG	42.23	9.84	38.70	9.49	41.12	8.69	37.64	8.74	39.88	9.13	36.82	8.88

segment), saving HLS some power in VOD operation compared to LIVE operation. Also, during VOD streaming, the Stream Segmenter stage is not active in HLS, therefore saving power compared to LIVE streaming. WVSNP-DASH has the same video segment structure and segment fetch operation for both VOD and LIVE streaming.

One would think that HLS will save more power than WVSNP-DASH given that WVSNP-DASH segments have complete self-initializing header data for each segment. That is, each WVSNP-DASH segment contains a file header with the video file properties and video container metadata, e.g., compression type, bit rate, and size, needed for decoding at the client (player) side [78]. This would imply that each WVSNP-DASH segment is slightly larger than HLS video segments that follow an HLS initialization segment. Surprisingly, our measurements indicated that for the same video resolution and quality, WVSNP-DASH segment files are on average smaller than the corresponding HLS segment files. The sizes vary widely within a range from segment to segment due to motion based video compression. On average, for 2 s segments, the HLS BIG segments are 21 % larger than the corresponding WVSNP-DASH segments, while the SMALL HLS segments are 28 % larger. For 5 s segments, the HLS BIG segments are 14 % larger than the corresponding WVSNP-DASH segments, while the HLS SMALL segments are 9.7 % larger. For 10 second segments the HLS BIG segments are on average 11.5 % larger, while the HLS SMALL segments are 9.7% larger.

The HLS segments are larger because the HLS design is based on the MPEG 2 Transport Stream (M2TS) [83] with a 188-Byte packet size, which was originally chosen for compatibility with Asynchronous Transfer Mode (ATM) systems. The packet size gets larger with additional headers, e.g., for synchronization, time code, and broadcasting meta-data. On the other hand, WVSNP-DASH is container agnostic, i.e., one can select whichever container is most efficient for WVSNP-DASH streaming. The MP4 container used by WVSNP-DASH in our evaluations follows an atom/box structure in a hierarchical form with four Bytes for the atom length, four Bytes for the atom name, and optional Bytes for any data the segment holds. The length of a box is determined by its own size plus all atoms in the level immediately below it. A basic WVSNP-DASH MP4 segment has three boxes: `ftyp`, `moov`, and `mdat`. Our segment file size measurement results indicate that the MP4 container utilized in WVSNP-DASH requires on average less overhead than the default HLS M2TS container. Generally, smaller segment files consume less power than large segment files; thus, the more efficient MP4 containers that become possible with WVSNP-DASH are generally preferable for low-power streaming.

4.3.2 Segmented vs. progressive (full) video streaming

In additional evaluations, we have measured the current consumption for progressive VOD (download) streaming of the full ten minute BIG video. We measured for WiFi transmission to the Mac an average current of 43.25 mA (SD = 9.44 mA) and for WiFi transmission to Windows an average current of 43.90 (SD = 8.72 mA). These currents for progressive streaming tend to be generally higher than for the corresponding segmented VOD video streaming considered in Table 3. (The differences to segment streaming are statistically significant at the $p < 0.001$ level for all scenarios, except for the Mac HLS 2 s and 5 s scenarios. Thus, the higher current consumption for progressive streaming compared to segmented streaming are conclusive for WVSNP-DASH, while for HLS this result is not conclusive for streaming to a Mac client.) This is because progressive download requires a large file in the server node. This large file requires relatively large amounts of energy for opening the file (before reading from the file). In particular, with progressive download,

the whole file is opened and a portion of the file (according to Byte ranges) is transmitted. However, when downloading the full video, the client has no control over how the HTTP server delivers the video data. More specifically, HTTP communication follows an open-loop communication paradigm. That is, the client requests the full video, but depending on the load and activity on the server node, the server node may switch contexts in between the transmission of Byte ranges of video data or may round robin schedule the video data transmission with its other processes. Whenever the server node switches to a different context, it has to close the video file. When the server node switches the context back to video data transmission, the large file has to be opened again and the node has to seek to the latest previously transmitted Byte range. The opening and closing of large full video files and the remembering of the position of the last transmitted video data Byte range consumes power.

In contrast, DASH clients request the HTTP server to send short video segments. These short segments do not need to be opened, they only need to be transmitted. Overall, we thus conclude that segmented videos are recommended for low-power streaming from a multimedia sensor node. The video segments also facilitate duty cycling [11] of sensor node applications and work well with the limited sensor node storage space.

4.3.3 Impact of segment length

We observe from Table 3 a general trend of VOD streaming of longer video segments consuming less power on the server node compared to shorter segments. (All comparisons with a mean difference of more than 1.5 mA are statistically significant at the $p < 0.001$) level; thus, the segment length results are statistically significant for most comparisons in Table 3.) Shorter segments require more server open and close activities as well as more frequent segment fetches, whereby each segment fetch operation requires connection setup resources. In additional evaluations, we found that 15 s segments consume slightly more current than 10 s segments. Overall, we conclude that for VOD, longer segment lengths up to 10 s are generally more power efficient than shorter segments. However, extending the segments length beyond 10 s then increases power consumption (due to the handling of large files).

5 Power consumption of node data path components

This section examines the power consumption of the main components of the multimedia server node that are critical for the video capture and store data path. We focus on the WVSNP-DASH video streaming framework in this path component study. Table 4 empirically shows the contribution of major video capture and store data path components.

5.1 FFmpeg vs. GStreamer

We first compare the use of the FFmpeg [21] and GStreamer (Gst) [28] video library tools, which are standard open source multimedia frameworks used in many embedded video systems. The top four rows in Table 4 indicate that GStreamer consumes substantially less power than FFmpeg (with statistical significance levels $p < 0.001$). Specifically, GStreamer consumes less power both for a USB attached camera and for a camera attached with the specialized Camera Serial Interface (CSI), for all considered video segment lengths. Based on this result, we employ GStreamer for the remaining evaluations in this section.

Table 4 Averages and standard deviations (SD) of current (mA) consumed by the server node while capturing WVSNP-DASH video segments. Comparison of major data path components for BIG video quality

Video types		2 s segments		5 s segments		10 s segments		Full Video	
Activity	Compared	Avg	SD	Avg	SD	Avg	SD	Avg	SD
SW-Enc FFmpeg H264 Capture	USB	77.07	17.00	80.66	16.41	81.56	16.38	78.38	14.78
	CSI	69.32	15.42	72.88	15.19	71.58	16.20	69.80	15.77
SW-Enc Gst H264 Capture	USB	59.49	11.89	60.13	12.07	58.70	12.32	62.84	11.81
	CSI	47.96	12.79	48.79	12.51	48.25	12.82	54.79	11.17
HW-Enc Gst H264 Capture	USB	57.97	13.36	59.07	12.80	58.80	13.54	58.82	13.62
	CSI	40.66	9.38	40.23	9.53	39.17	9.80	37.87	9.42
Gst H264 USB Capture	HW-ENC	57.97	13.37	58.46	13.46	58.80	13.54	59.11	13.29
	SW-ENC	59.49	11.89	59.60	12.65	59.30	11.64	62.69	11.93
Gst H264 CSI Capture	HW-ENC	40.66	9.38	40.23	9.53	39.05	9.90	38.15	9.17
	SW-ENC	48.02	12.73	48.83	12.47	48.67	12.39	54.79	11.17
Gst JPEG USB Capture	HW-ENC	67.53	16.19	67.28	16.96	72.84	16.50	.	.
	SW-ENC	61.94	13.41	62.24	13.37	62.50	12.65	.	.
Gst JPEG CSI Capture	HW-ENC	43.55	9.67	.	.	.	.	.	.
	SW-ENC	56.01	13.49	.	.	.	.	.	.
Gst MPEG4 USB Capture	HW-ENC	65.30	15.96	67.26	15.73	66.55	15.64	.	.
	SW-ENC	61.00	13.37	61.56	13.31	61.73	13.84	.	.
Gst MPEG4 CSI Capture	HW-ENC	44.21	8.19	.	.	.	.	.	.
	SW-ENC	49.01	11.07	.	.	.	.	.	.

5.2 USB vs. camera serial interface (CSI)

The top six rows in Table 4 compare the attachment of a Logitech "webcam" via the general USB interface with the attachment of a "wandcam" camera via the specialized camera serial interface (CSI). We observe from Table 4 that the CSI camera consumes less power than the USB camera (with statistical significance levels $p < 0.001$). This is because the USB camera processes frames through an additional USB Video Class (UVC). The raw data collected by the USB camera has to pass through multiple USB, peripheral, and other bus stacks, increasing the required processing resources. Additionally, the USB camera requires a SW hand-off of raw data on the board for video encoding to the SW/HW encoder which increases the current consumption. The CSI stack is smaller and more specifically optimized for the i.MX6 System-On-Chip (SoC), which has a direct path from the Wandcam CMOS imager [19] driver to the encoder [81]. Specifically, the i.MX SoC has an accelerated path from the CSI camera to the video processing unit (VPU) and onward to direct memory access. Also, the ARM core on the i.MX6 SoC is completely decoupled from the video capture to memory path [81].

5.3 Impact of segment length

The top ten rows in Table 4 compare capturing the full 10 minute video with capturing 2, 5, and 10 second segments. Generally, there are no pronounced differences in the current consumption when capturing segmented video with the different segment lengths or the full video. This is because FFmpeg and GStreamer have the capability of segmenting videos while the capturing is ongoing without turning off the camera. However, similar to the results in Section 4.1.3, we observe from Table 4 a tendency for shorter segments to consume less current than longer segments or the full video. However, this segment length effect is small compared to the effects of the examined node data path factors.

5.4 Hardware vs. software encoder

We observe from Table 4 that video capture with HW encoder consumes in most cases less current than a SW encoder, except for JPEG and MPEG4 encoding with the USB attached camera. (All HW vs. SW comparisons are statistically significant at the $p < 0.0001$ level; except the Gst H.264 USB 2 s comparison is significant at only the $p < 0.01$ level and the corresponding 5 s and 10 s comparisons are not significant.) HW encoding is mainly performed by dedicated SoC components, such as a Video Processing Unit (VPU). In contrast, a SW encoder mainly utilizes the board's CPU for the video processing. Dedicated processors are generally more efficient if they rely on accelerated, optimized job-specific instructions. For the considered i.MX6 SoC, the VPU is the HW accelerator for image and video processing. As a result, significant reductions in consumed current can be achieved with HW encoders that employ the VPU instead of the general CPU for video processing.

Considering HW or SW encoder in combination with either CSI or USB camera, we observe from Table 4 that for the CSI camera, HW encoding consumes about 15 % less current than SW encoding. As noted in Section 5.2, the USB stack adds processing layers that consume substantial amounts of power and skew the difference in observed current consumption for HW vs. SW encoder. As a result of the USB effects, we observe only small current reductions for H264 video captured with USB camera and HW encoder vs. the corresponding SW encoder scenarios in Table 4; namely a current reduction at the $p < 0.0001$ and $p < 0.01$ statistical significance levels for the full video and the 2 s segment

length, respectively, and no statistically significant current reductions for the 5 s and 10 s segment lengths.

Aside from the H.264 (vpuenc, codec=6) encoder, we evaluated the SW jpegenc encoder and AVI (vpuenc, codec=0) HW encoder, i.e., the Audio Video Interleaved (AVI) encoder, which is based on the JPEG encoder, in conjunction with the AVI video container. Surprisingly, we found that for the JPEG (AVI) encoder with USB capture in rows 11 and 12 of Table 4, the HW encoder consumes more current than the software encoder. This is an unexpected result that requires follow-up in future research. Possibly, the implementation of the HW encoder has performance issues or is not fully equivalent to GStreamer's jpegenc SW encoder. Nevertheless, the CSI capture with JPEG encoding results follow the expected trend that HW encoding consumes less current than SW encoding. Thus, it is possible that the current consumption for JPEG encoding (and analogously for the MPEG4 SW encoder and HW (vpuenc, codec=12) encoder with MP4 video container) is skewed by the extra USB processing.

Overall, we conclude that HW encoding of video captured with the CSI camera is more energy-efficient than using a USB camera with the same HW codec. We also observe from Table 4 for the CSI capture with HW encoding a slight trend of decreasing current consumption as the segment length increases. For short segments there are more frequent application layer software hand-offs of raw data to the HW encoder, i.e., the VPU, which result in a slight under utilization of the VPU. Longer segments reduce the hand-off frequency, increasing VPU utilization, and increasing the overall energy efficiency of the video capture.

6 Conclusion and future work

This paper presented extensive measurements for power profiling of Dynamic Streaming over HTTP (DASH) frameworks for miniaturized wireless multimedia sensor nodes. Manifest file based DASH frameworks, such as HTTP Live Streaming (HLS), convey video meta data through the manifest file and begin video streaming with a special initialization video segment; subsequent video segments depend on the manifest file and initialization segment for playback. In contrast, the name based WVSNP-DASH framework conveys essential meta data through the names of the video segment files, which can be played independently; i.e., each individual WVSNP-DASH segment is fully playable without reference to any other file or segment. This file independence simplifies the video capture and video segment file creation and streaming by a sensor node.

Our measurements indicated reduced power consumption on the sensor node for video capture and LIVE streaming with the WVSNP-DASH framework compared to the HLS framework. We also found that capturing and LIVE streaming with short 2 s video segments, which are preferable to avoid long time lags and large buffering requirements, consumes less power than longer segments. For VOD streaming of previously captured video segments, we found that both frameworks have comparable power consumption. We also measured the power consumption characteristics of sensor node design choices for employed video library tools, interfaces for the camera, and hardware vs. software video encoding. We found that the video library tools from the GStreamer open source multimedia framework and the Camera Serial Interface (CSI) have favorable power consumption characteristics. Also, we extensively quantified the power savings due to hardware video encoding.

Overall, this study has provided extensive empirical baseline power consumption data for key sensor node architectural design components. In summary, we recommend the name-

based WVSNP-DASH framework instead of a manifest file based framework, such as HLS, for capturing and streaming video from miniaturized multimedia sensor nodes. WVSNP-DASH generally consumed less power for live video capture and real-time LIVE streaming.

An added WVSNP-DASH advantage is its flexibility: HLS is presently limited to the MPEG2-TS video container, while WVSNP-DASH video segments can be encapsulated into arbitrary video containers, such as MP4, AVI, MPEG2-TS, and WebM. WVSNP-DASH is easier to implement for a server node and different OSs as it is officially playable on different browsers (e.g., Chrome, Windows Internet Explorer, Firefox), while HLS is officially only playable on Safari or Chrome (at the time of these experiments). Additionally, WVSNP-DASH segments can be flexibly cached and distributed as well as flexibly searched and retrieved [49] due to their name based structure. On the other hand, video segment files of manifest file based frameworks cannot be independently cached and distributed by storage-constrained sensor nodes. Any work-around to centrally manage the manifest file creates additional local network coupling, that will likely further increase the power consumption in the server node or network.

We acknowledge that our measurement set-up and experimental procedures had a number of key limitations. First, our measurements did not consider the dynamic adaptation feature of the HLS and MPEG-DASH streaming frameworks; rather, we streamed either a SMALL or BIG video quality (see Section 3.4) for the full duration of a given measurement run. Future measurement studies should examine the impacts of varying frequencies of video quality switches on the power consumption. Second, we conducted the WVSNP-DASH measurements with a prototype system that opened and closed the video capture process for each segment with a power-inefficient script (whereas the compared commercial HLS system employed an optimized capture process). Thus, WVSNP-DASH has significant further power saving potential that should be quantified in future work. In particular, future work should evaluate WVSNP-DASH capture with optimized capture processing through a native application running on the server (camera) node (and not a script). Third, we employed the FFmpeg capture with a USB interface and H.264 software encoding in the comparison of the WVSNP-DASH and HLS frameworks in Section 4. In future evaluations, it would be of interest to repeat the WVSNP-DASH vs. HLS framework comparison with the node data path components that have been found to reduce power consumption in the WVSNP-DASH framework in Section 5, namely GStreamer capture with the CSI interface and hardware H.264 encoding.

There are many additional interesting directions for future research on low-power streaming from miniaturized sensor nodes. One important direction is to investigate the impact of different non-TCP/IP networking protocol stacks for resource-constrained sensor (server) nodes, such as Zigbee [1, 4, 77, 87] and Bluetooth. The present study focused on the TCP/IP networking protocol stack since the existing manifest file based DASH players are designed to work only with video server nodes operating the TCP/IP networking protocol stack. Thus, a comparison with the existing commercial manifest file based streaming framework was only possible with the TCP/IP networking stack. Moreover, it would be interesting to compare the LIVE streaming of multiple HLS streams from multiple distinct source nodes to one player with the corresponding streaming of multiple WVSNP-DASH source nodes to one player. Such a comparison would more comprehensively evaluate the cost of a manifest file, by accounting for the cost for switching due to bandwidth and quality variations as well as the cost for the switching among different sources (which may require route recomputations). Another important direction is to examine the impact of security and privacy mechanisms [24, 29, 36, 50] on sensor node power consumption.

References

1. Abas K, Porto C, Obraczka K (2014) Wireless smart camera networks for the surveillance of public spaces. IEEE Comput 47(5):37–44
2. Accardi K, Yates A (2017) Powertop user's guide, https://01.org/powertop
3. Adzic V, Kalva H, Furht B (2012) Optimizing video encoding for adaptive streaming over http. IEEE Trans Consum Electron 58(2):397–403
4. Akyildiz IF, Melodia T, Chowdhury KR (2007) A survey on wireless multimedia sensor networks. Comput Netw 51(4):921–960
5. Android power management on i.MX6DQ/DL (2012) https://community.freescale.com/docs/DOC-93884/version/1
6. Anisi MH, Abdul-Salaam G, Abdullah AH (2015) A survey of wireless sensor network approaches and their energy consumption for monitoring farm fields in precision agriculture. Precis Agric 16(2):216–238
7. Apple Inc. (2012) Example playlist files for use with http live streaming—technical note tn2288. technical report, https://developer.apple.com/library/content/technotes/tn2288/
8. Bicakci K, Gultekin H, Tavli B (2009) The impact of one-time energy costs on network lifetime in wireless sensor networks. IEEE Commun Lett 13:12
9. Bourdon A, Noureddine A, Rouvoy R, Seinturier L (2012) Powerapi: A software library to monitor the energy consumed at the process-level. http://abourdon.github.com/powerapi-akka
10. Braeckman G, Hanca J, Kleihorst R, Munteanu A (2015) Power consumption analysis of a wireless 1K-pixel visual sensor node: to compress or not? In: Proceedings of the ACM International Conference on Distributed Smart Cameras, pp 170–174
11. Carrano RC, Passos D, Magalhaes LC, Albuquerque CV (2014) Survey and taxonomy of duty cycling mechanisms in wireless sensor networks. IEEE Commun Surv Tutorials 16(1):181–194
12. Castellanos WE, Guerri JC, Arce P (2017) SVCEval-RA: An evaluation framework for adaptive scalable video streaming. Multimed Tools Appl 76(1):437–461
13. Chan K, Lee JY (2016) Improving adaptive HTTP streaming performance with predictive transmission and cross-layer client buffer estimation. Multimed Tools Appl 75(10):5917–5937
14. Chang H-Y (2018) A connectivity-increasing mechanism of ZigBee-based IoT devices for wireless multimedia sensor networks. Multimed Tools Appl print, pp 1–18
15. Chen S, Yuan Z, Muntean G-M (2016) An energy-aware routing algorithm for quality-oriented wireless video delivery. IEEE Trans Broadcast 62(1):55–68
16. Chien S-Y, Cheng T-Y, Ou S-H, Chiu C-C, Lee C-H, Somayazulu VS, Chen Y-K (2013) Power consumption analysis for distributed video sensors in machine-to-machine networks. IEEE J Emerg Sel Top Circ Syst 3(1):55–64
17. Cotuk H, Bicakci K, Tavli B, Uzun E (2014) The impact of transmission power control strategies on lifetime of wireless sensor networks. IEEE Trans Comput 63(11):2866–2879
18. Cotuk H, Tavli B, Bicakci K, Akgun MB (2014) The impact of bandwidth constraints on the energy consumption of wireless sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp 2787–2792
19. Engineering Services (2017) Avnet. WandCam (AES-WCAM-ADPT-G)—getting started guide. [Online]. Available: http://www.em.avnet.com/wandcam
20. Feldt R, Magazinius A (2010) Validity threats in empirical software engineering research–an initial survey. In: SEKE, pp 374–379
21. FFmpeg (2017) [Online]. Available: http://ffmpeg.org
22. Free Electrons (2011) Power management. http://free-electrons.com/doc/power-management.pdf
23. Gayan (2012) Powerstat: Power Consumption Calculator for Ubuntu Linux. http://www.hecticgeek.com/2012/02/powerstat-power-calculator-ubuntu-linux
24. Ghadi M, Laouamer L, Moulahi T (2016) Securing data exchange in wireless multimedia sensor networks: perspectives and challenges. Multimed Tools Appl 75(6):3425–3451
25. Ghasemzadeh H, Panuccio P, Trovato S, Fortino G, Jafari R (2014) Power-aware activity monitoring using distributed wearable sensors. IEEE Trans Human-Mach Syst 44(4):537–544
26. Go Y, Kwon OC, Song H (2015) An energy-efficient HTTP adaptive video streaming with networking cost constraint over heterogeneous wireless networks. IEEE Trans Multimed 17(9):1646–1657
27. González S, Vargas TR, Arce P, Guerri JC (2016) Energy optimization for video monitoring system in agricultural areas using single board computer nodes and wireless ad hoc networks. In: Proceedings of the IEEE Symposium on Signal Processing, Images and Artificial Vision (STSIVA), pp 1–7
28. GStreamer: Open source multimedia framework (2017) [Online]. Available: http://gstreamer.freedesktop.org/

29. Guerrero-Zapata M, Zilan R, Barceló-Ordinas JM, Bicakci K, Tavli B (2010) The future of security in wireless multimedia sensor networks. Telecommun Syst 45(1):77–91
30. Haas C, Wilke J, Stöhr V (2012) Realistic simulation of energy consumption in wireless sensor networks. In: Wireless Sensor Networks. Springer, pp 82–97
31. Haas C, Munz S, Wilke J, Hergenroder A (2013) Evaluating energy-efficiency of hardware-based security mechanisms. In: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp 560–565
32. Hamid Z, Hussain FB, Pyun J-Y (2016) Delay and link utilization aware routing protocol for wireless multimedia sensor networks. Multimed Tools Appl 75(14):8195–8216
33. Heitmann N, Kindt P, Chakraborty S (2016) EG0N: Portable in-situ energy measurement for low-power sensor devices. In: IEEE International Symposium on VLSI Design and Test (VDAT), pp 1–6
34. Hergenroeder A, Wilke J, Meier D (2010) Distributed energy measurements in WSN testbeds with a sensor node management device (SNMD). In: Proceedings of the International Conference on Architecture of Computing Systems (ARCS), pp 1–7
35. Hergenroder A, Furthmuller J (2012) On energy measurement methods in wireless networks. In: Proceedings of the IEEE International Conference on Communications (ICC), pp 6268–6272
36. Heydari N, Minaei-Bidgoli B (2017) Reduce energy consumption and send secure data wireless multimedia sensor networks using a combination of techniques for multi-layer watermark and deep learning. Int J Comp Sci Netw Sec (IJCSNS) 17(2):98–105
37. Horneber J, Hergenröder A (2014) A survey on testbeds and experimentation environments for wireless sensor networks. IEEE Commun Surv Tutorials 16(4):1820–1838
38. Huang S-C, Chang H-Y (2017) A farmland multimedia data collection method using mobile sink for wireless sensor networks. Multimed Tools Appl 76(19):19 463–19 478
39. Javaid S, Fahim H, Hamid Z, Hussain FB (2018) Traffic-aware congestion control (TACC) for wireless multimedia sensor networks. Multimed Tools Appl, in print, pp 1–20
40. Jiang X, Dutta P, Culler D, Stoica I (2007) Micro power meter for energy monitoring of wireless sensor networks at scale. In: Proceedings of the International Symposium on Information Processing in Sensor Network (IPSN), pp 186–195
41. Karakus C, Gurbuz AC, Tavli B (2013) Analysis of energy efficiency of compressive sensing in wireless sensor networks. IEEE Sens J 13(5):1999–2008
42. Kidwai NR, Khan E, Reisslein M (2016) ZM-SPECK: A fast and memoryless image coder for multimedia sensor networks. IEEE Sens J 16(8):2575–2587
43. Kim Y, Bok K, Son I, Park J, Lee B, Yoo J (2017) Positioning sensor nodes and smart devices for multimedia data transmission in wireless sensor and mobile P2P networks. Multimed Tools Appl 76(16):17 193–17 211
44. Ko JH, Mudassar BA, Mukhopadhyay S (2015) An energy-efficient wireless video sensor node for moving object surveillance. IEEE Trans Multi-Scale Comput Syst 1(1):7–18
45. Kua J, Armitage G, Branch P (2017) A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP. IEEE Commun Surv Tutorials 19(3):1842–1866. Third Qu
46. Lee J-Y, Jung K-D, Moon S-J, Jeong H-Y (2017) Improvement on LEACH protocol of a wide-area wireless sensor network. Multimed Tools Appl 76(19):19 843–19 860
47. Lenk J (1997) Simplified design of voltage/frequency converters, series edn series for design engineers. elsevier science. [Online]. Available: http://books.google.com/books?id=a8avGaqz8AoC
48. Li Y, Shen D, Zhou G (2017) Energy optimization for mobile video streaming via an aggregate model. Multimed Tools Appl 76(20):20 781–20 797
49. Liu B, Gupta A, Jain R (2008) MedSMan: A live multimedia stream querying system. Multimed Tools Appl 38(2):209–232
50. Liu CZ, Kavakli M (2018) A data-aware confidential tunnel for wireless sensor media networks. Multimed Tools Appl, in print, pp 1–23
51. Magno M, Boyle D, Brunelli D, Popovici E, Benini L (2014) Ensuring survivability of resource-intensive sensor networks through ultra-low power overlays. IEEE Trans Ind Inf 10(2):946–956
52. Margi CB, Petkov V, Obraczka K, Manduchi R (2006) Characterizing energy consumption in a visual sensor network testbed. In: Proceedings of the International Conference on Testbeds & Research Infrastructure for the DEvelopment of NeTworks & COMmunities, pp 1–8
53. Mekonnen T, Harjula E, Ylianttila M (2016) Energy efficient motion detection in a high-resolution wireless surveillance camera node. In: Proceedings of the ACM International Conference on Mobile and Ubique Multimedia, pp 373–375
54. Milenkovic A, Milenkovic M, Jovanov E, Hite D, Raskovic D (2005) An environment for runtime power monitoring of wireless sensor network platforms. In: Southeastern Symposium on System Theory, (SSST), pp 406–410

55. Misra S, Mohanta D (2010) Adaptive listen for energy-efficient medium access control in wireless sensor networks. Multimed Tools Appl 47(1):121–145
56. Mooshimeter M (2016) Dmm-ble-2x01a mooshimeter technical specifications. [online]. available: https://moosh.im/mooshimeter/specs/
57. Naderiparizi S, Parks AN, Parizi FS, Smith J (2016) μmonitor: In-situ energy monitoring with microwatt power consumption. In: Proceedings of the IEEE International Conference on RFID (RFID), pp 1–8
58. Noureddine A, Bourdon A, Rouvoy R, Seinturier L (2012) A preliminary study of the impact of software engineering on GreenIT. In: Proceedings of the International Workshop on Green and Sustainable Software, Zurich, pp 21–27
59. Noureddine A, Bourdon A, Rouvoy R, Seinturier L (2012) Runtime monitoring of software energy hotspots. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE), pp 160–169
60. Pantazis N, Vergados D (2007) A survey on power control issues in wireless sensor networks. IEEE Commun Surv Tutorials 9(4):86–107. 4th Quarter
61. Pantazis NA, Nikolidakis SA, Vergados DD (2013) Energy-efficient routing protocols in wireless sensor networks: A survey. IEEE Commun Surv Tutorials 15(2):551–591
62. Pantos R, May W (2017) HTTP live streaming, working draft, IETF secretariat, Internet-Draft draft-pantos-http-live-streaming-21.txt. http://developer.apple.com/resources/http-streaming. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-21
63. Popovici E, Magno M, Marinkovic S (2013) Power management techniques for wireless sensor networks: a review. In: Proceedings of the IEEE International Workshop on Advance in Sensors and Interface (IWASI), pp 194–198
64. Porambage P, Heikkinen A, Harjula E, Gurtov A, Ylianttila M (2016) Quantitative power consumption analysis of a multi-tier wireless multimedia sensor network. In: Proceedings of the VDE Eu Wireless Conference, pp 1–6
65. Rainer B, Petscharnig S, Timmerer C, Hellwagner H (2017) Statistically indifferent quality variation: An approach for reducing multimedia distribution cost for adaptive video streaming services. IEEE Trans Multimed 19(4):849–860
66. Ramakrishna M, Karunakar A (2017) SIP and SDP based content adaptation during real-time video streaming in future internets. Multimed Tools Appl 76(20):21 171–21 191
67. Rashid B, Rehmani MH (2016) Applications of wireless sensor networks for urban areas: A survey. J Netw Comput Appl 60:192–219
68. Rault T, Bouabdallah A, Challal Y (2014) Energy efficiency in wireless sensor networks: A top-down survey. Comput Netw 67:104–122
69. Redondi A, Buranapanichkit D, Cesana M, Tagliasacchi M, Andreopoulos Y (2014) Energy consumption of visual sensor networks: Impact of spatio-temporal coverage. IEEE Trans Circ Syst Video Technol 24(12):2117–2131
70. Rein S, Reisslein M (2011) Low-memory wavelet transforms for wireless sensor networks: A tutorial. IEEE Commun Surv Tutorials 13(2):291–307
71. Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131–164
72. Saginbekov S, Shakenov C (2016) Testing wireless sensor networks with hybrid simulators, arXiv:1602.01567
73. SanMiguel JC, Cavallaro A (2017) Energy consumption models for smart camera networks. IEEE Trans Circuits Syst Video Technol 27(12):2661–2674
74. Sarif BA, Pourazad M, Nasiopoulos P, Leung VC (2014) Analysis of power consumption of H.264/AVC-based video sensor networks through modeling the encoding complexity and bitrate. In: Proceedings of the International Confernece on Digital Society (ICDS), pp 1–6
75. Sarif BA, Pourazad M, Nasiopoulos P, Leung VC (2015) A new scheme for estimating H.264/AVC-based video sensor network power consumption. In: Proceedings of the World Congress on Information Technology Applications and Services, pp 1–3
76. Sarif BA, Pourazad M, Nasiopoulos P, Leung VC (2015) A study on the power consumption of H.264/AVC-based video sensor network, International Journal of Distributed Sensor Networks
77. Seema A, Reisslein M (2011) Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a Flexi-WVSNP design. IEEE Commun Surv Tutorials 13(3):462–486. 3rd Quarter
78. Seema A, Schwoebel L, Shah T, Morgan J, Reisslein M (2015) WVSNP-DASH: Name-based segmented video streaming. IEEE Trans Broadcast 61(3):346–355

79. Schroeder D, Ilangovan A, Reisslein M, Steinbach E (2018) Efficient multi-rate video encoding for HEVC-based adaptive HTTP streaming. IEEE Transactions on Circuits and Systems for Video Technology, in print
80. Shin H, Park J-S (2017) Optimizing random network coding for multimedia content distribution over smartphones. Multimed Tools Appl 76(19):19 379–19 395
81. Slotfeldt T (2016) Simplify graphical user interface and video integration for i.MX 6 series processors. [Online]. Available: https://community.nxp.com/docs/DOC-104267
82. Smith J, Colwell A, Wolenetz M (2016) Webm byte stream format, w3c, W3C note. https://www.w3.org/TR/2016/NOTE-mse-byte-stream-format-webm-20161004/
83. Smith J, Watson M, Wolenetz M, Bateman A, Colwell A (2016) Mpeg-2 TS byte stream format, w3c, W3C note. https://www.w3.org/TR/2016/NOTE-mse-byte-stream-format-mp2t-20161004/
84. Snajder B, Jelicic V, Kalafatic Z, Bilas V (2016) Wireless sensor node modelling for energy efficiency analysis in data-intensive periodic monitoring. Ad Hoc Netw 49:29–41
85. Sodagar I (2011) The MPEG-DASH standard for multimedia streaming over the internet. IEEE MultiMed 18(4):62–67
86. Stamatescu G, Chiţu C, Vasile C, Stamatescu I, Popescu D, Sgârciu V (2014) Analytical and experimental sensor node energy modeling in ambient monitoring. In: Proceedings of the International Conference on Industrial Electronics and Applications (ICIEA), pp 1615–1620
87. Tavli B, Bicakci K, Zilan R, Barcelo-Ordinas JM (2012) A survey of visual sensor network platforms. Multimed Tools Appl 60(3):689–726
88. Thang TC, Ho Q-D, Kang J-W, Pham A (2012) Adaptive streaming of audiovisual content using MPEG DASH. IEEE Trans Consum Electron 58(1):78–85
89. Thomas E, van Deventer M, Stockhammer T, Begen AC, Champel M-L, Oyman O (2016) Applications and deployments of server and network assisted DASH (SAND). In: Proceedings of the IET IBC Conference, pp 1–8
90. Titzer B, Lee D, Palsberg J (2005) Avrora: scalable sensor network simulation with precise timing. In: Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN), pp 477–482
91. Trasvina-Moreno CA, Asensio A, Casas R, Blasco R, Marco A (2014) WiFi sensor networks: A study of energy consumption. In: Proceedings of the IEEE International Multi-Conference on System, Signals Devices (SSD), pp 1–6
92. van Rest J, Grootjen F, Grootjen M, Wijn R, Aarts O, Roelofs M, Burghouts GJ, Bouma H, Alic L, Kraaij W (2014) Requirements for multimedia metadata schemes in surveillance applications for security. Multimed Tools Appl 70(1):573–598
93. Vergados DJ, Michalas A, Sgora A, Vergados DD, Chatzimisios P (2016) FDASH: A fuzzy-based MPEG/DASH adaptation algorithm. IEEE Syst J 10(2):859–868
94. Walpole RE, Myers RH, Myers SL, Ye K (2016) Probability and statistics for engineers and scientists, 9th edn. Pearson, UK
95. Watson M, Colwell A, Bateman A, Smith J, Wolenetz M (2016) Iso bmff byte stream format, w3c, W3C note. https://www.w3.org/TR/2016/NOTE-mse-byte-stream-format-isobmff-20161004/
96. Watson M, Colwell A, Bateman A, Smith J, Wolenetz M (2016) Media source extensions™, W3C W3C Recommendation. https://www.w3.org/TR/2016/REC-media-source-20161117/
97. Wohlin C, Höst M, Henningsson K (2003) Empirical research methods in software engineering. In: Empirical Methods and Studies in Software Engineering, Lecture Notes in Computer Science (LNCS), Volume 2765. Springer, pp 7–23
98. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer-Verlag, Berlin
99. Wunderlich S, Cabrera J, Fitzek F, Reisslein M (2017) Network coding in heterogeneous multicore IoT nodes with DAG scheduling of parallel matrix block operations. IEEE Internet Things J 4(4):917–933
100. Yan R, Sun H, Qian Y (2013) Energy-aware sensor node design with its application in wireless sensor networks. IEEE Trans Instrum Measur 62(5):1183–1191
101. Yap FG, Yen H-H (2014) A survey on sensor coverage and visual data capturing/processing/transmission in wireless visual sensor networks. Sensors 14(2):3506–3527
102. Yuan D, Kanhere SS, Hollick M (2017) Instrumenting wireless sensor networks—a survey on the metrics that matter. Pervasive Mob Comput 37:45–62
103. Zhang J, Fang G, Peng C, Guo M, Wei S, Swaminathan V (2016) Profiling energy consumption of DASH video streaming over 4G LTE networks, pp 3:1–3:6
104. Zhang C, Liu J, Chen F, Cui Y, Ngai EC-H, Hu Y (2017) Dependency-and similarity-aware caching for HTTP adaptive streaming. multimedia tools and applications, in print, pp 1–22

Adolph Seema has a B.S. in Computer Engineering from the Rochester Institute of Technology., NY, USA, He has a Ph.D. in Electrical Engineering from the School of Electrical, Computer, and Energy Engineering at Arizona State University. (ASU), Tempe, AZ, USA. He is also a Cryptography & Security IP Design Engineer at NXP Semiconductors in Chandler, AZ, USA.

Tejas Shah received the M.S. degree in Electrical Engineering from Arizona State University, Tempe, AZ, USA in 2014. He currently works at Qualcomm Inc. in San Diego, CA.

Lukas Schwoebel received the B.S. degree in computer science from TU Darmstadt, Germany, and the M.S. degree in applied computer science from the University of Bamberg, Germany. His M.S. thesis examined the WVSNP project at Arizona State University, Tempe, AZ, USA. He is currently the Chief Technical Officer with Favendo GmbH, Germany.

Yu Liu is an associate professor in the School of Information and Communication Engineering at Beijing University of Posts and Telecommunications and a visiting scholar in the School of Electrical, Computer, and Energy Engineering at Arizona State University. She specializes in video compression and information processing in wireless networks.

Martin Reisslein is a Professor in the School of Electrical, Computer, and Energy Engineering at Arizona State University (ASU), Tempe. He received the Ph.D. in systems engineering from the University of Pennsylvania in 1998. He currently serves as Associate Editor for the IEEE Transactions on Mobile Computing, the IEEE Transactions on Education, and IEEE Access as well as Computer Networks and Optical Switching and Networking. He is Associate Editor-in-Chief for the IEEE Communications Surveys & Tutorials and chairs the steering committee of the IEEE Transactions on Multimedia.