WVSNP-DASH: Name-Based Segmented Video Streaming

Adolph Seema, *Member, IEEE,* Lukas Schwoebel, Tejas Shah, Jeffery Morgan, and Martin Reisslein, *Fellow, IEEE*

***Abstract*—Video streaming from sensors and miniaturized devices is attractive for a wide range of web-based applications, e.g., remote surveillance. Existing web-based video streaming frameworks, such as the hypertext transfer protocol (HTTP) live streaming (HLS) and the motion picture experts group's dynamic adaptive streaming over HTTP (MPEG-DASH), have dependencies between the individual video segments and a manifest file that contains video metadata. Also, existing web-based video players are limited to fetching video segments over TCP/IP networks. The video segment dependencies complicate video segment distribution by resource-constrained source nodes, which may employ non-TCP/IP protocols, such as Zigbee. This paper proposes and evaluates a wireless video sensor network platform compatible DASH (WVSNP-DASH) framework and a WVSNP-DASH player (WDP) for flexible web-based access of video from sensors and other miniaturized source nodes. The WVSNP-DASH framework is based on independently playable video segments with a specific naming syntax that conveys elementary metadata so as to facilitate flexible search, transfer, distribution, and playback. The WDP employs elementary processes of version 5 of the hypertext markup language (HTML5) for video buffering and playback. Video segments are fetched into the HTML5 file system space, permitting flexible video fetching over a wide range of protocols, including sensor network protocols. Comparative evaluations of a WDP prototype with optimized HLS and MPEG-DASH players indicate that WDP has low client (receiver) load, while providing significant potential for power savings on the source node serving the video streams.**

***Index Terms*—Dynamic adaptive HTTP streaming (DASH), HTTP live streaming (HLS), HTML5 file system (FS), video distribution, video streaming, wireless video sensor.**

I. Introduction

A. State-of-the-Art: Segmented Video Streaming With Manifest File Dependencies

Segmented video streaming has become common for the distribution of web-based video content over Internet Protocol (IP) communication networks. The popular HyperText Transfer Protocol Live Streaming (HLS) and the Motion Picture Experts Group Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [1]–[4] distribute streaming video through video segments with fixed playback duration. Segmented video streaming can adapt to varying congestion levels on IP networks through switching the video quality (and bit rate) when transitioning from one segment to the next. Adaptation is important for video streaming in IP networks in order to maximize performance trade-offs [5], [6]. The different video segments can flexibly provide adaptive video streaming in a wide range of contexts, such as IP Television (IPTV) [7] as well as the emerging hybrid broadband broadcast television (HbbTV) [8]–[10] and other innovative distributions strategies [11]–[15].

An important structural characteristic of the existing segmented video streaming frameworks, such as HLS and MPEG-DASH, is that they rely on special manifest files and initialization segments to convey the video meta information. These manifest files require special segment generation tools and create dependencies between the segments of a given video stream. These dependencies need to be initialized and maintained by the video server and the client playing back the video. Another important characteristic of state-of-the-art players for DASH streaming video is that they can only interact with video server nodes operating the TCP/IP networking protocol stack. Also, as reviewed in detail in Section II, the existing adaptive segmented video streaming frameworks are not directly supported by version five of the Hypertext Markup Language (HTML5).

B. Motivation: Video Acquisition by and Distribution From Miniaturized Devices and Sensors

Miniaturized devices, such as smart watches and sensors, are becoming ubiquitous and have increasingly multimedia capabilities [16]–[19]. Video data from miniaturized devices and sensors has the potential to enhance a variety of entertainment, residential, and industrial services. Additionally, general multi-dimensional (2D and 3D) data, e.g., from infrared sensors, heat maps, Light Emitting Diode (LED) pixel sensor maps, x-rays, and many other wirelessly linked remote acquisition devices and sensors are becoming ubiquitous and can augment multimedia services.

Miniaturized devices and sensors have tight resource constraints, including limited power (which typically has to be supplied by batteries) [20]. Initializing and maintaining the dependencies between the individual video segments and the manifest files can become a significant burden for power-constrained devices and sensors. Also, the file dependencies restrict the independent caching and distribution of video segments by storage-constrained video acquisition devices.

Resource-constrained devices often communicate through non-TCP/IP protocol stacks, such as

Manuscript received December 18, 2014; accepted February 1, 2015. Date of publication February 13, 2015; date of current version September 2, 2015.

A. Seema, T. Shah, J. Morgan, and M. Reisslein are with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706 USA (e-mail: reisslein@asu.edu).

L. Schwoebel is with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706 USA, and also with the Department of Applied Computer Science, University of Bamberg, Bamberg 96047, Germany.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2015.2400816

Zigbee [16], [17], [21], [22]. However, the existing segmented video streaming frameworks are not designed to communicate with non-TCP/IP protocol stacks and thus limit video streaming to video acquisition devices with a TCP/IP networking stack. Moreover, since existing DASH players do not directly support adaptive playback through HTML5, they require complex plug-ins that invite security vulnerabilities or have very limited cross-platform support [23], [24].

C. Contributions: Name-Based Distribution and Playback of Independently Playable Video Segments

This paper addresses the limitations of existing segmented video streaming frameworks for streaming from miniaturized devices outlined in the preceding section. Specifically, we introduce an alternative to the segmented video streaming with dependencies to a manifest file. This novel segmented video streaming framework is well suited for video streaming from miniaturized wireless devices and sensors and we refer to it therefore as Wireless Video Sensor Node Platform (WVSNP, may be pronounced “WaveSnap”) compatible DASH framework, abbreviated as WVSNP-DASH. Each video segment in the proposed WVSNP-DASH framework is an *independently playable* file carrying its essential metadata in its name. The proposed WVSNP-DASH framework includes a specific name syntax for video segments. The name syntax conveys essential meta information about the video segment; thus eliminating dependencies to manifest (and initialization) files. The proposed WVSNP-DASH framework is highly backward compatible and interfaces with wireless video acquisition devices and sensor networks without special re-design of video file formats, video containers, sensor nodes, or networks. WVSNP-DASH is video container agnostic and encapsulates any container, codec, or Digital Rights Management (DRM) [25], as long as the web browser supports it.

This paper also presents the design of a WVSNP-DASH Player (WDP) that is based on core elements of HTML5. The WDP provides a user interface to the WVSNP-DASH framework by allowing client (playback) devices to retrieve and play video from miniaturized video aquisition/sensor nodes. WDP does not rely on any plug-in or back-end engines. Instead, WDP employs elementary downloading through HTTP as well as the HTML5 Filesystem (FS) together with the HTML5 video tag for managing video segment retrieval, transmission, and playback. The video segment fetching into the HTML5 FS enables video segment delivery from non-TCP/IP networks, such as Zigbee networks. The video segments are displayed on the HTML5 canvas element.

The WVSNP-DASH framework is evaluated with a WDP prototype that is compared with optimized HLS and MPEG-DASH framework players. To the best of the authors’ knowledge, this is the first evaluation of DASH video streaming from sensor nodes. This study thus provides empirical baseline performance data for the streaming of sensor video with different frameworks.

II. Background and Related Work

A. HTML5 Video Playback

Playback of video is supported by most modern web browsers as a feature of HTML5 [26]–[29]. Generally, HTML5 based video players utilize the HTML5 video tag to download and play full-length video files. However, the HTML5 <video> tag is not uniformly nor completely implemented by common web browsers, requiring players to implement work-arounds if the video tag fails to open, play, or render the <video> source.

Also, with HTML5, adaptation of the video bit rate or presentation quality would require the re-download of the entire video file. Media Source Extensions (MSEs), which have been developed by the World Wide Web Consortium (W3C) [30], could serve as a basis for adaptive streaming in HTML5 based video players. However MSE support is inconsistent in popular web browsers, limiting the cross-browser compatibility of an MSE based player design.

Alternatively, web browser video plug-ins (see [31]–[34]), could adapt the video streaming through slicing the full-length video file and monitoring the download of the different presentation qualities (versions). However, such plug-ins can give rise to a multitude of security and incompatibility issues as well as the burden on the user to update and maintain the plug-ins [23], [24].

Zhu *et al*. [35] recently designed a pure HTML5 based video player that uses the canvas element to display video in different web browsers. The player circumvents the problem of accommodating different video codecs by decoding the received video chunks using JavaScript in the browser into an intermediate format. The intermediate format is then passed to the browser’s video tag for decoding with the native video decoder of the browser. However, performing both the video decoding to an intermediate format and the video drawing on the canvas in the browser leads to very high CPU load, which is prohibitive for mobile devices. A player based on 3D graphics rendering has been proposed in [36].

The proposed WDP design does not require MSE, nor plug-ins; instead, WDP relies only on the legacy single <video> tag reference. Also, WDP does not use any extra decoder module; instead, WDP uses the decoder engine native to the browser, resulting in no extra CPU load for the video decoding.

B. DASH Video Streaming and Playback

According to the Dynamic Adaptive Streaming over HTTP (DASH) specification [37], [38], a web-based DASH video player must be able to adapt the video presentation quality level during playback by switching among different quality versions of the video stream. A plug-in free player must support the playback of chunks (segments) derived from a video stream via an HTML5 video element. DASH players support interactions, such as jump backward (rewind, RW) or forward (fast forward, FF) by fetching the video segment for the desired playback point. However, existing DASH players must first process the manifest file and play the initialization segment before such playback jumps. The WDP does not require an initialization segment and thus provides truly random segment playback on demand.

Quacchio *et al*. [39] proposed a DASH player that utilizes a custom-built Web-kit based browser to achieve customized handling of the HTML5 video elements. The proposed player design relies heavily on plug-ins to achieve adaptive streaming. In order to facilitate the adoption of the

MPEG-DASH standard [37], [40], [41], the DASH Industry Forum developed a reference player, namely dash.js [42]. The dash.js player employs media source extensions (MSEs). Several derivatives of the dash.js player (see [31], [43]), have recently been proposed. These derivatives require specific web browsers and extensive plug-in support. Similarly, there are a variety of DASH video players available that work only in conjunction with specific web browsers and require plug-ins or rely on the Real Time Messaging Protocol running over TCP/IP.

More importantly, a thorough literature search and examination of a wide range of available proprietary player solutions revealed that none of the existing video players are designed to integrate with non-TCP/IP networks, such as resource-constrained wireless sensor networks (WSNs) employing the Zigbee protocol. Overall, these recent DASH player developments do not comply with the cross-platform design goal of HTML5, instead they are limited to specific web browsers with their respective plug-ins. In contrast to the recent DASH player developments, the goal of the proposed WVSNP-DASH framework is to make video data from sensor networks as widely accessible as possible with little effort, if any, from the consumer device user.

We note for completeness that recent research has sought to provide additional features, such as subtitles and annotations [44], [45], in DASH video streaming as well as examined the implications of the adaptive DASH streaming on network resource requirements [46]–[49]. Other complementary related research has sought to optimize the video encoding for DASH streaming [50], [51] and improve buffer management algorithms and segment scheduling [52], [53].

C. Video Sensor Networking

There has been a recent focus on the lower layers of the Internet protocol stack for integrating sensor networks into the overall Internet of Things (IoT) [54]–[56]. A widely considered approach is IP version 6 based Low power Wireless Personal Area Networks (6LoWPAN), which covers radios and firmware that compress IPv6 packet headers into smaller 6LoWPAN headers suitable for low-data-rate IEEE 802.15.4 personal area network standards, such as Zigbee, operating in sensor networks [57], [58]. The proposed WVSNP-DASH framework and WDP are designed to directly work within these low-data-rate mesh networks. Also, some platforms and gateways have been introduced to make the sensor data accessible over IP networks and to enable interactions between networked sensors and consumer devices [59]–[62].

Video sensor focused services may be viewed as components in service based frameworks [63] to serve as a cloud-based repository directories of sensor data using service oriented architectures (SOA), infrastructures, and protocols [64]–[67]. However, video sensors are typically not designed to take advantage of SOA data exchange structures. In particular, sensor data cloud repositories presently have no concept of video as search-able or addressable sensor data. The name syntax of the proposed WVSNP-DASH framework enables cloud repositories to offer video segments as part of data sets and services.

A multimedia playback framework based on MPEG-DASH within an information centric network [68] has recently been proposed in [69]–[71]. The framework in [70] encompasses named addressing and routing within the context of TCP/IP networks. The name-based structure of the proposed WVSNP-DASH framework is complementary to the framework in [69] and [70] and compatible with information centric networking. At the same time, the proposed WVSNP-DASH framework is designed to work beyond TCP/IP networks so that it makes the video data on sensor nodes employing other (non-TCP/IP) network protocols readily available to consumer electronics. The proposed framework can furthermore readily be combined with efficient sensor network routing techniques, e.g., routing based on virtual coordinates [72]–[74].

III. WVSNP-DASH Framework

A. Independent Video Segments

The WVSNP-DASH framework facilitates multimedia acquisition, storage, distribution, and playback through assigning a unique name to a given independently playable multimedia object. If a node has a WiFi, Zigbee, or Bluetooth radio, a video object source can be uniquely named for WDP to be able to fetch the object. This WVSNP-DASH design of independently playable media objects with a specific naming syntax enables video data object distribution across networks, including cross-network data transfers between traditional IP networks and sensor networks.

All WVSNP-DASH video segments have the same type and format; in particular, each segment is a complete video file. The sensor (server) node only processes the video file data when creating the video file. A created video file is then always ready to be fetched and transmitted without any further pre-processing on the sensor (server) node. This approach reduces sensor (server) node power consumption compared to the existing HLS and MPEG-DASH frameworks, see Section V-C.

The WVSNP-DASH framework is conceptually similar to the MSE-based MPEG-DASH [42] in that it does not require the browser to directly support DASH via the video element. MSE only exposes the HTML5 media element to its interface, which then allows flexible appending of media segments to this exposed element. The browser continues to perform the traditional decoding and rendering. The intelligence required to parse manifest files, request segments, and switch adaptively is left to the player client script. This is somewhat of an improvement over HLS in that HLS requires browsers to be re-written and the video tag to be modified to support manifest files; MSE avoids these modifications by preprocessing the stream and passing video chunks to the video tag as if each chunk was a traditional HTML5 supported video file. WDP further simplifies this concept by not requiring the server to specially format multimedia data, as MPEG-DASH and HLS do.

B. WVSNP-DASH Segment Name Syntax

The WVSNP-DASH framework prescribes a video segment naming syntax that uniquely names each video segment. The syntax of the segment name has complete information for a WVSNP-DASH player (WDP) to be able to play back the video files stored in a remote network node. The client should be able to play back an entire video-on-demand (VOD) set or live video based solely on the meta information gleaned from parsing the segment name.

The WVSNP-DASH segment naming syntax follows the simplified Backus-Naur Form [75] `<filename>-<maxpresentation>-<presentation>-<mode>-<maxindex>-<index>.<ext>`
The components of the name syntax are defined as follows:

- `<filename>`: This is a unique string for each video stream (set of video segments). The <filename> represents the path, e.g., through an IP address or a URL, to the video stream, or represents a unique prefix describing the stream.
- `<maxpresentation>`: This integer defines the index of the highest presentation quality (e.g., quality version) available for the stream.
- `<presentation>`: The actual presentation quality of this video-segment file. A lower index defines a lower quality of the stream, whereby 0 is defined as the lowest index denoting the lowest available video quality.
- `<mode>`: This string indicates the playback mode of this segment, e.g., video on demand (`VOD`) or live playback (`LIVE`).
- `<maxindex>`: This integer gives the total number of segments available for playback for the current video stream.
- `<index>`: This integer gives the index of this segment within the finite set of segments of this stream.
- `<ext>`: This string indicates the video container format of this segment, e.g., `.mp4` or `.webm`. The player decides if the container format and encoded video can be played back and informs the user accordingly.

This simple WVSNP-DASH video segment name syntax contains the complete information needed by the WVSNP-DASH client to retrieve and play the video segments in a sensor network.

C. Implications

The simplicity of this name syntax has far-reaching implications as it transfers most of the video playback and retrieval complexity to the player. The player is the consumer of the video data, and thus knows what it wants to do with the video data and how to interpret the video data.

The WDP introduced in this paper demonstrates the concept of the uniqueness of the video segment name. Since WSNs are resource constrained in terms of power, computing resources, and storage space, WVSNP-DASH enforces that video files within a sensor network are captured and stored as complete, individually playable video files of short duration, preferably no longer than ten seconds (the impact of the segment length is examined in Section V-C).

Recall that with HLS and MPEG-DASH, the player requires the manifest file in conjunction with the individual video segment files for playback. In contrast, the WVSNP-DASH video segment name syntax ensures that the name of the segment, or any other future network video object, conveys sufficient information for the player to decide how to fetch and play the video segments. The manifest files required by HLS and MPEG-DASH introduce incompatibility issues as they require browsers to support manifest files as well as live playback maintenance of the manifest files. By communicating all player pertinent metadata through the segment file name, WVSNP-DASH avoids these incompatibility issues and is thus highly backward compatible.

Another important feature of the WVSNP-DASH framework is random playback of any segment, as needed by the player. In particular, WVSNP-DASH has no special initialization segment, which is necessary for HLS and MPEG-DASH players to understand how to play a video. Each WVSNP-DASH video segment has enough metadata (in its name) to start playing the video. This feature enables unlimited "crawler type" network search algorithms. These can be used by future information centric network (ICN) routers and content distribution networks (CDNs) [68]. Future active players may start playing the video as soon as a segment is discovered.

For instance, as detailed in Section IV-B, a segment name: ***`src=filename-1-1-VOD-45-7.mp4`*** can be easily passed to the player. Based on the meta information contained in this name, the player can initiate the streaming process of segments 7 to 45 by fetching all segments and playing them one after another. The random playback feature of WVSNP-DASH enables the player to perform arbitrary fast forward (FF) and rewind (RW) of the video. The naming syntax of WVSNP-DASH also enables the player to take advantage of all DASH features, such as adaptive switching.

IV. WVSNP-DASH Player (WDP)

A. Design and Implementation

The WDP design goal is to rely only on widely supported HTML5 features, so as to achieve broad cross-platform support. Selecting HTML5 features that are reliably supported by most browsers is generally difficult. Therefore, the first prototype version of WDP is constrained to using only official core HTML5 features. WDP combines the advantages of common support for HTTP downloading via Asynchronous JavaScript and XML (AJAX) (through the XMLHttpRequest functionality) [23] as well as the HTML5 File System (FS) Application Programmers Interface (API) [76]. The HTML5 FS is run-time memory allocated to a process running on a browser. Data objects in this protected (sandbox) space can be cached for future use by the same URL on the same browser. Other URLs in separate tabs or other browser windows cannot access this memory space. Both AJAX and the FS API are used as they are broadly supported HTML5 features [23], [76]. Where the FS API is not yet fully supported, third party wrappers such as IndexedDB [77] are used to mimic the FS API. This approach helps to keep the design logic intact across browsers as well as to maintain compatibility as official support improves.

B. Flow

1) Compatibility Test: The entry point of the player is the browser detection and compatibility test module illustrated Fig. 1. This module tests if the browser supports the core

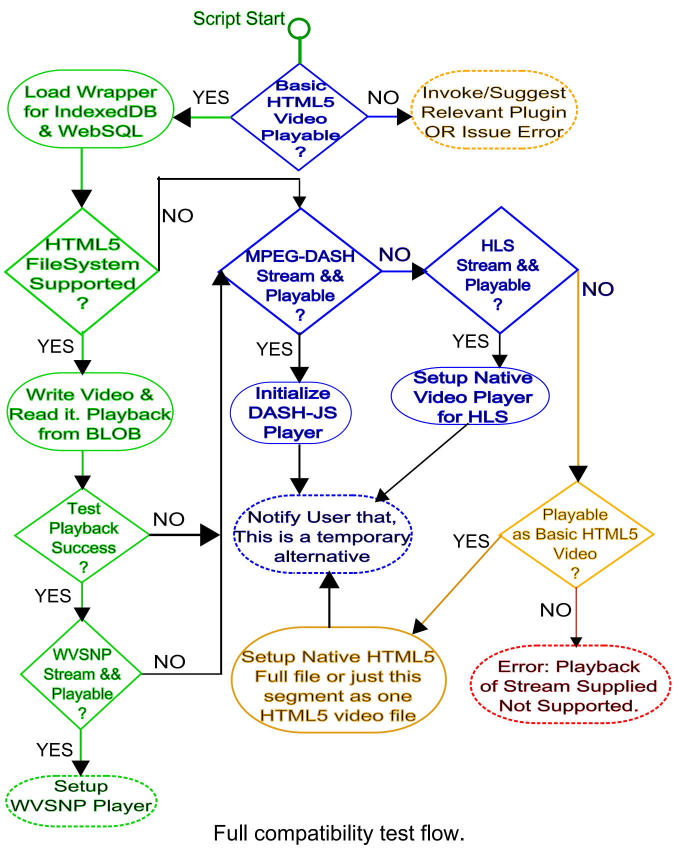
Script Start
Basic HTML5 Video Playable ?
YES
NO
Load Wrapper for IndexedDB & WebSQL
Invoke/Suggest Relevant Plugin OR Issue Error
HTML5 FileSystem Supported ?
NO
YES
MPEG-DASH Stream && Playable ?
HLS Stream && Playable ?
Write Video & Read it. Playback from BLOB
Initialize DASH-JS Player
Setup Native Video Player for HLS
Test Playback Success ?
Notify User that, This is a temporary alternative
Playable as Basic HTML5 Video ?
WVSNP Stream && Playable ?
Setup Native HTML5 Full file or just this segment as one HTML5 video file
Error: Playback of Stream Supplied Not Supported.
Setup WVSNP Player
Full compatibility test flow.

Fig. 1. Illustration of compatibility test flow.

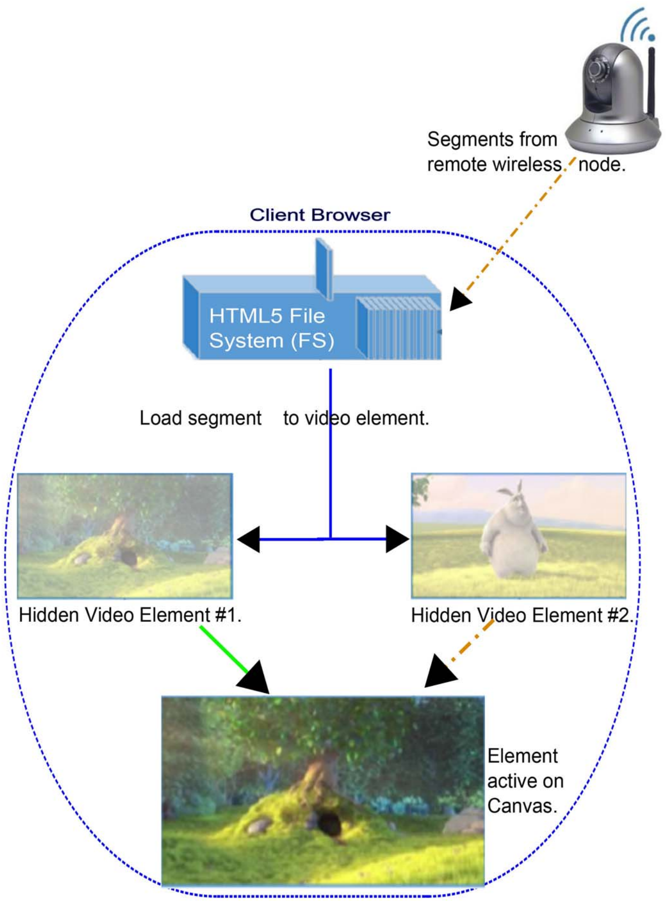
Segments from remote wireless node.
Client Browser
HTML5 File System (FS)
Load segment to video element.
Hidden Video Element #1.
Hidden Video Element #2.
Element active on Canvas.

Fig. 2. Overview of WVSNP-DASH player (WDP): video segments are fetched from the remote sensor (server) node into the HTML5 file system (FS) by the buffering process. The playback process plays the current segment from one hidden video element to the canvas element, while the other hidden video element loads the next segment from the HTML5 FS.

HTML5 features used by WDP. AJAX tools check support for downloading, saving, and playing back the requested video via FS API or a suitable wrapper. If an MPEG-DASH manifest file is provided, and the browser supports MSE, the DASH-JS player (a precursor of dash.js) is launched for playback as an independent module within the WDP modular architecture. Alternatively, for an HLS manifest file, the native video player of the web browser is set up with HLS. If the URL (name) of a requested video does not match the WVSNP-DASH video segment name syntax, and neither the HLS nor MPEG-DASH manifest syntax, WDP assumes a traditional HTML5 compatible video file. The HTML5 video file is then played back in the browser's native HTML5 video element as legacy full-length video. Otherwise, the default is to play WVSNP-DASH video.

The WDP design is relatively future proof in that WDP works on any browser with support for the HTML5 core elements. Also, any future new video codec, such as H.265/HEVC or VP9, or any future video container format, that can be played as a whole video via an HTML5 <video> element, is supported by WDP.

2) Buffering and Playback Processes: The player consists of two main processes running in parallel: a buffering process and a playback process. Each process has its own writing (buffer) and reading (player) counters. The buffering process runs continuously in a loop to fetch segments from the remote server node into the HTML5 FS, see top part in Fig. 2. Each loop iteration handles one video segment (file) for the currently selected video stream and presentation quality. The file is represented as a binary large object (Blob) [78]. A Blob is a data structure that encapsulates raw binary data and can be fed directly to an HTML5 <video> element. For VOD, the buffering process runs continuously until the last segment has been downloaded. However, the user may restart the process with an arbitrary segment by changing the presentation quality or fast-forwarding to a point beyond the buffer line.

The playback process is at least three segments behind the buffering process. As shown in Fig. 2, in order to stich the segments seamlessly together, there are two hidden <video> elements which contain consecutive video segments. When one segment ends, the corresponding <video> element triggers an event, which in turn triggers the playback of the next video segment from the other <video> element. The now unused <video> element is linked to the subsequent segment by loading the segment from the HTML5 FS. Similar to the buffering process, the file is loaded as raw binary data in a Blob variable and fed directly to the <video> element.

Glitches in the form of resizing <video> elements or black screens during the transitions from one <video> element to the next are avoided with an HTML5 <canvas> element. Each frame of the currently active video element is drawn on the <canvas> element. During a transition, the last active frame is displayed. With the canvas, the transition glitches caused by the slow JavaScript are not visible to the viewer. For streams with continuous audio, longer video segments reduce glitches in the audio track playback due to fewer transitions. Audio fading techniques can be employed to seamlessly morph the audio tracks of the video segments; such fading techniques are beyond the video-specific scope of this paper.

The use of the HTML5 FS in conjunction with the HTML5 canvas for rendering video in WDP enables the rendering of video that arrives to the player via other network mechanisms, aside from basic HTTP/AJAX mechanisms [23]. For instance, the WDP design allows the rendering of images and data that arrive to the client via Common Gateway Interface (CGI) frameworks [79], which are very important for cross-network exchanges, e.g., between Zigbee and Bluetooth networks. The WDP architecture can also readily take advantage of peer-to-peer features of the client and local hardware access features, e.g., via the emerging Web Real Time Communication (WebRTC) interface [80]. WDP's networking flexibility, combined with WVSNP-DASH's independently playable video segments (i.e., no need for manifest file or initialization segment), enables simultaneous streaming and interweaving of video segments from different video acquisition devices/sensors and different networks, e.g., from Bluetooth, Zigbee, and WiFi networks.

V. WDP Evaluation

A. Evaluation Set-up

A prototype of WDP was extensively compared with popular DASH players considering the following metrics:

- *CPU load*: Especially for mobile client (playback) devices, the central processing unit (CPU) load during video playback, which directly influences power consumption and battery drainage, should be low.
- *Memory consumption*: The consumption of working memory, which is limited in mobile devices, gives an indication of the efficiency of resource handling.
- *Power consumption*: Low power consumption at the server side, i.e., the video aquisition/sensor node, which captures, stores, and serves the video files, is critical [20].

For the prototype evaluation, a typical surveillance video was captured on the campus of Arizona State University (ASU). The 10 minute ASU video (without audio) contains timelapse captures of everyday outdoor activity and scenery around the ASU campus. The ASU video was used for both VOD and LIVE video scenarios. For the LIVE video scenario, a node camera was pointed at the full-screen display of the prerecorded video while measurements were captured in real time. Focusing and positioning the node camera to see only the full-screen video display made the LIVE video essentially identical to the pre-recoded video. The H.264 encoded .mp4 video was generated with 640×360 pixel resolution, 500 kb/s, and 25 frames/s. The video was segmented into MPEG2 Transport Stream (TS) segments for HLS playback, independent MP4 segments for WVSNP-DASH playback, and ISO Base Media File Format (BMFF) segments for MPEG-DASH [81]. The segments were created for 2, 5, 10, and 15 second segment lengths.

The WDP prototype was compared with HLS using the JWPlayer 6 with the HLSProvider plug-in and with MPEG-DASH using the DASH-JS player. The presented evaluations were performed with the Google Chrome (version 32) web browser running on a client operating on a *Ubuntu 13.10* 64 bit, *Dell OptiPlex 360* with *Intel Core2 Duo E7300* 2.66 GHz processor and 2 GB RAM. The evaluations were also run with a client operating on *Macbook Air Mid 2012* with *i5-3427U* 1.8 GHz processor and 4 GB RAM as well as a *Windows 7* 64 bit client booted on the same *Macbook* and gave similar results, which are not included due to space constraints.

The server node power consumption was measured from an *i.MX6 ARM Cortex-A9*, 1.2 GHz Quad core, 2 GB node development board. The server node captured video with a USB webcam and generated and served video segments via WiFi and a streamlined low-footprint mongoose HTTP server. The performance metrics were sampled 14,000 times during the 10 minute video transmission, which was replicated five times.

B. Client Node CPU Load and Memory Consumption

We observe from Table I that for the 5 s LIVE scenario, WDP has a similar CPU load as HLS (JWPlayer 6) while the WDP memory consumption is somewhat lower than for HLS. On the other hand, for the 5 s VOD scenario, WDP has higher CPU load and memory consumption than HLS and MPEG-DASH (DASH-JS). The higher CPU load and memory consumption of WDP for the VOD scenario are mainly due to the WDP buffering algorithm for VOD, which fetches all segments as fast as the network bandwidth and the client FS space allocation allow. In contrast, HLS buffers only approximately three segments and discards them after playback. WDP stores all segments in the FS space leading to high memory usage. This WDP approach facilitates power savings on the server node by avoiding re-fetches during quality switches or actions that reuse previously fetched VOD segments (such as rewind).

The WDP LIVE buffering algorithm behaves similar to HLS. As Table I indicates, WDP LIVE playback has lower CPU load and memory consumption than HLS. This is remarkable in that the WDP prototype uses a resource-heavy canvas element as well as two video elements concurrently to render the video, while HLS uses a highly optimized HLSProvider Flash helper plug-in to the browser with only one video element. MPEG-DASH (DASH-JS) playback, included for the 5 and 10 s VOD scenarios in Table I, has the lowest CPU load and memory consumption. This MPEG-DASH result indicates that using pure Javascript and MSE with one video element can be more efficient than using a plug-in.

In order to further examine the high CPU load and memory consumption of WDP for VOD, the WDP prototype was slightly modified to play segments directly in the video elements without multiplexing two video elements nor rendering on the canvas. This modified WDP, which is denoted by "WVSNP-no-c" in Table I, has significantly lower CPU load and memory consumption than HLS as well as similar CPU load and memory consumption as MPEG-DASH. This result

TABLE I
AVERAGE AND STANDARD DEVIATION (SD) OF CPU LOAD AND MEMORY CONSUMPTION (IN MEGABYTE) ON WIFI CLIENT NODE AS WELL AS POWER CONSUMPTION (MEASURED THROUGH DRAWN CURRENT) ON SERVER NODE

LIVE 5 second segments

	CPU load (%)		Memory (MB)		Current (mA)	
Framework	Avg.	SD	Avg.	SD	Avg.	SD
HLS	69.5	8.8	136	9	95.8	29.6
WVSNP	68.4	26.5	122	16	100	26.5
WVSNP-no-c	31.2	12.1	126	15	100	26.5

VOD 5 second segments

	CPU load (%)		Memory (MB)		Current (mA)	
Framework	Avg.	SD	Avg.	SD	Avg.	SD
HLS	61.2	6.3	145	7	37.5	9.92
WVSNP	79.6	12.1	182	18	49.5	8.27
WVSNP-no-c	36.9	13.4	189	21	49.5	8.27
DASH	30	7	143	18	N/A	N/A

VOD 2 second segments

	CPU load (%)		Memory (MB)		Current (mA)	
Framework	Avg.	SD	Avg.	SD	Avg.	SD
HLS	62.8	7.4	145	9	48.8	8.04
WVSNP	79.2	10.2	187	26	51.1	8.43
WVSNP-no-c	39.7	9	180	22	51.1	8.43

VOD 10 second segments.

	CPU load (%)		Memory (MB)		Current (mA)	
Framework	Avg.	SD	Avg.	SD	Avg.	SD
HLS	61.1	6.7	146	7	36.7	8.57
WVSNP	79.3	16	180	29	48.0	8.25
WVSNP-no-c	37.3	14.4	200	25	48.0	8.25
DASH	29.5	6.5	168	21	39.1	9.13

Progressive Download of Full Video, No segmentation.

	CPU load (%)		Memory (MB)		Current (mA)	
Framework	Avg.	SD	Avg.	SD	Avg.	SD
Full Video	28.7	4.5	91	2	43.1	9.57

for WSNP-no-c, i.e., WDP without using the canvas element, indicates that the high WDP CPU load and memory consumption for VOD are mainly due to the canvas element. This validates that the use of full (independently playable) video segments does not increase client resource usage compared to the manifest/initialization segment file approach.

C. Server Node Power Consumption

Since the voltage readings were generally consistent at 5 V, only the current drawn by the server node during each scenario is reported as a relative measure of power consumption in Table I. The results for current in Table I indicate that the LIVE scenarios have significantly higher currents and thus higher power consumption than the VOD scenarios. This is because the server node captures, transcodes, stores, and serves the video segments at the same time. More importantly for the LIVE scenario, the WDP prototype has only slightly higher power consumption than HLS. In interpreting these power results, it is important to note the differences in using ffmpeg for segment capture in the compared frameworks. HLS captured and transcoded LIVE video in an optimized built-in (native) HLS function of ffmpeg, which achieves highly efficient software-based capture and transcoding. On the other hand, WVSNP-DASH capture used a prototype-level bash script that newly invoked ffmpeg for each LIVE segment capture. This means that the WVSNP-DASH prototype incurred extra power, resources, and inefficiencies for each context switch of launching ffmpeg, transcoding, storing, and then shutting down the ffmpeg process for each video segment capture. Moreover, WVSNP-DASH interpreted a script at run time for every segment, adding to the resource usage. In contrast, HLS capture invoked ffmpeg only once at the start of the video stream capture and captured the remaining segments with the same optimized ffmpeg (from the original invocation context). These conceptual differences imply that an optimized native WVSNP-DASH capture application has considerable power savings potential for LIVE video in the WVSNP-DASH framework compared to the already optimized HLS framework.

A further potential for power savings arises from the WVSNP-DASH operation without a manifest file. HLS and MPEG-DASH require a manifest file that needs to be managed and re-read and updated during the capture and/or playback. For VOD, the manifest files and segments are typically static and the indices do not need to be continuously updated. However, for synchronization of LIVE video, the manifest files typically have to be re-fetched regularly for LIVE video synchronization. Additional processing to create special subsequent segments different from the initialization segment adds to the power consumption of HLS and MPEG-DASH for LIVE video.

For VOD, the results for current in Table I indicate that longer segments generally reduce the power consumption. Again, noting the inefficiencies in the WVSNP-DASH prototype due to script interpretation and ffmpeg invocation for each video segment, there are power saving potentials for WVSNP-DASH compared to HLS and MPEG-DASH.

In order to further examine the impact of the segmented video streaming, the full 10 minute ASU video was streamed via progressive download and the results are reported in the bottom line of Table I. Table I indicates that 10 s segmented HLS and MPEG-DASH streaming consumed less power compared to full-video streaming. These results indicate that segmented video streaming does not lead to higher server node power consumption than streaming a full (unsegmented) video to an HTML5 element. The 36.7 mA measured for HLS for 10 s VOD segments can be considered as the worst-case (maximum) current consumption expected of a WVSNP-DASH using a specialized native capture. More generally, the comparison of currents for 10 s VOD segments and full video download indicates that segmented streaming can result in about 15 % power savings compared to streaming progressively downloaded full videos. From further test evaluations, that are not included due to space constraints, it was noted that there is no benefit of using 15 s segments compared to 10 s segments.

In WVSNP-DASH, each video segment file has its own file header. The file header contains all the video file properties and internal video container metadata, such as duration, compression type, size, and bit rate, that are needed for decoding by any player. In HLS and MPEG-DASH, most of this file header metadata is moved to the first segment and the remaining segments are merely fixed data elements (blocks) that cannot be decoded independently. Thus, it might appear that WVSNP-DASH introduces some overhead by including the file header in each video segment file, whereas HLS and MPEG-DASH are continuous streams with random access points due to preset intracoded (I) frames. However, due to the self-contained file headers in each WVSNP-DASH segment, there is no look-up data that needs to be maintained and referenced at the server node every time from the initial segment when there are quality switches for dynamic adaptation. This reduction of server node processing for managing the video segments has the potential to reduce power in video sensor nodes. WVSNP-DASH segments can still be captured with specific I-frame positioning to match efficient transcoding practices.

VI. Conclusion and Future Work

This article presented the design structure of the Wireless Video Sensor Network Platform compatible Dynamic Adaptive Streaming over HTTP (WVSNP-DASH) framework. The WVSNP-DASH framework specifies a naming syntax for independently playable video segments. Existing DASH frameworks convey video metadata through a 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 WVSNP-DASH video segments convey essential metadata through their name and 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 file segment creation and streaming by a miniaturized video aquisition/sensor node.

This article also presented the design of the WVSNP-DASH Player (WDP). WDP is based on core HTML5 features ensuring wide cross-platform support. WDP fetches video segment files into the HTML5 File System (FS) and plays them without requiring plug-ins on the HTML5 canvas element. While existing DASH players require specific plug-ins or protocols running over TCP/IP, the fetching into the HTML5 FS in WDP enables video playback from non-TCP/IP server nodes, e.g., video sensor nodes operating the Zigbee protocol.

The comparative evaluation of a WDP prototype against optimized HLS and MPEG-DASH players indicated that the WDP prototype has competitive client CPU and memory consumption. Also, the independently playable WVSNP-DASH video segments create significant potential for power savings on the sensor node serving the video. To the best of the authors knowledge the presented evaluation is the first to examine the effects of different DASH frameworks on sensor node power consumption.

The source code of the WDP prototype is freely available from the authors so as to enable open-source based further development [82]. Aside from the refinements of the WDP prototype noted in Sections V-B and V-C, such as operating with a single ffmpeg initiation, there are a number of important directions for future development and research. For instance, Media Source Extensions (MSE) have recently been increasingly adopted by web browsers, thus incorporating the ability to use MSE into WDP where HTML5 FS is not supported is becoming a useful feature for ensuring broad cross-platform support. The modular WDP design is similar to how dash.js, see Section II-B, utilizes MSE, facilitating the WDP extension to MSE. Another future development direction is automatic dynamic quality level adaptation. The WDP prototype manually selects the desired quality level of a video stream. Dynamic adaptation have been extensively covered in the literature (see [83]–[92]). A module can be added to WDP for instantiating specific automated dynamic adaptation algorithms. In the wider networking context, video streaming from sensor networks requires moreover research on architectural structures and protocol mechanisms for efficient interconnection with access and metropolitan area networks [93]–[96].

References

[1] C. Concolato, J. Le Feuvre, and R. Bouqueau, "Usages of DASH for rich media services," in *Proc. ACM Conf. Multimedia Syst.*, Santa Clara, CA, USA, 2011, pp. 265–270.

[2] T. Lohmar, T. Einarsson, P. Frojdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, Lucca, Italy, 2011, pp. 1–8.

[3] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, Oct.-Dec. 2011.

[4] N. Staelens *et al.*, "Subjective quality assessment of longer duration video sequences delivered over HTTP adaptive streaming to tablet devices," *IEEE Trans. Broadcast.*, vol. 60, no. 4, pp. 707–714, Dec. 2014.

[5] A. Molnar and C. H. Muntean, "Cost-oriented adaptive multimedia delivery," *IEEE Trans. Broadcast.*, vol. 59, no. 3, pp. 484–499, Sep. 2013.

[6] R. Trestian, O. Ormond, and G.-M. Muntean, "Energy-quality-cost tradeoff in a multimedia-based heterogeneous wireless network environment," *IEEE Trans. Broadcast.*, vol. 59, no. 2, pp. 340–357, Jun. 2013.

[7] J. Maisonneuve *et al.*, "An overview of IPTV standards development," *IEEE Trans. Broadcast.*, vol. 55, no. 2, pp. 315–328, Jun. 2009.

[8] J.-C. Dufourd, S. Thomas, and C. Concolato, "Recording and delivery of HbbTV applications," in *Proc. 9th Int. Interact. Conf. Interact. Televis.*, Lisbon, Portugal, 2011, pp. 51–54.

[9] R. Malhotra, "Hybrid broadcast broadband TV: The way forward for connected TVs," *IEEE Consum. Electron. Mag.*, vol. 2, no. 3, pp. 10–16, Jul. 2013.

[10] S. Kaiser, S. Pham, and S. Arbanowski, "MPEG-DASH enabling adaptive streaming with personalized commercial breaks and second screen scenarios," in *Proc. 11th Eur. Conf. Interact. TV Video*, Como, Italy, 2013, pp. 63–66.

[11] B. Ciubotaru, G. Ghinea, and G.-M. Muntean, "Subjective assessment of region of interest-aware adaptive multimedia streaming quality," *IEEE Trans. Broadcast.*, vol. 60, no. 1, pp. 50–60, Mar. 2014.

[12] L. Shi, E. Obregon, K. W. Sung, J. Zander, and J. Bostrom, "CellTV—On the benefit of TV distribution over cellular networks: A case study," *IEEE Trans. Broadcast.*, vol. 60, no. 1, pp. 73–84, Mar. 2014.

[13] C. Xu, E. Fallon, Y. Qiao, L. Zhong, and G. Muntean, "Performance evaluation of multimedia content distribution over multi-homed wireless networks," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 204–215, Jun. 2011.

[14] C. Xu, S. Jia, L. Zhong, H. Zhang, and G.-M. Muntean, "Ant-inspired mini-community-based solution for video-on-demand services in wireless mobile networks," *IEEE Trans. Broadcast.*, vol. 60, no. 2, pp. 322–335, Jun. 2014.

[15] M. Zorrilla, A. Martin, F. Mogollon, J. Garcia, and I. G. Olaizola, "End to end solution for interactive on demand 3D media on home network devices," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Seoul, Korea, 2012, pp. 1–6.

[16] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Comput. Netw.*, vol. 51, no. 4, pp. 921–960, Mar. 2007.

[17] K. Abas, C. Porto, and K. Obraczka, "Wireless smart camera networks for the surveillance of public spaces," *IEEE Comput.*, vol. 47, no. 5, pp. 37–44, May 2014.

[18] L. D. P. Mendes, J. J. P. C. Rodrigues, J. Lloret, and S. Sendra, "Cross-layer dynamic admission control for cloud-based multimedia sensor networks," *IEEE Syst. J.*, vol. 8, no. 1, pp. 235–246, Mar. 2014.

[19] F. G. Yap and H.-H. Yen, "A survey on sensor coverage and visual data capturing/processing/transmission in wireless visual sensor networks," *Sensors*, vol. 14, no. 2, pp. 3506–3527, Feb. 2014.

[20] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Trans. Sensor Netw. (TOSN)*, vol. 5, no. 1, pp. 1–39, Feb. 2009.

[21] A. Seema and M. Reisslein, "Towards efficient wireless video sensor networks: A survey of existing node architectures and proposal for a Flexi-WVSNP design," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 462–486, Nov. 2011.

[22] B. Tavli, K. Bicakci, R. Zilan, and J. M. Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools Appl.*, vol. 60, no. 3, pp. 689–726, Oct. 2012.

[23] L. D. Paulson, "Building rich web applications with Ajax," *IEEE Comput.*, vol. 38, no. 10, pp. 14–17, Oct. 2005.

[24] C. Reis, A. Barth, and C. Pizano, "Browser security: Lessons from Google Chrome," *ACM Queue*, vol. 7, no. 5, pp. 1–8, Jun. 2009.

[25] S. Kaiser and S. Pham, "DRM-interoperable MPEG-dash end-to-end architecture," in *Proc. IEEE Int. Conf. Multimedia Expo. Workshops (ICMEW)*, Chengdu, China, 2014, pp. 1–2.

[26] M. Hoernig, A. Bigontina, and B. Radig, "A comparative evaluation of current HTML5 web video implementations," *Open J. Web Technol.*, vol. 1, no. 2, pp. 1–9, 2014.

[27] L. Stevens and R. Owen, "The truth about audio and video in HTML5," in *The Truth About HTML5*. New York, NY, USA: Apress, 2014, pp. 117–133.

[28] E. Tzoc and J. Millard, "For video streaming/delivery: Is HTML5 the real fix?" *Code4Lib J.*, vol. 2013, no. 22, Oct. 2013.

[29] S. J. Vaughan-Nichols, "Will HTML 5 restandardize the web?" *IEEE Comput.*, vol. 43, no. 4, pp. 13–15, Apr. 2010.

[30] M. W. Aaron Colwell and A. Bateman. (Mar. 3, 2014). *Media Source Extensions—W3C Editor'S Draft*. [Online]. Available: https://dvcs.w3.org/hg/html-media/raw-file/tip/media-source/media-source.html

[31] C. Müller and C. Timmerer, "A VLC media player plugin enabling dynamic adaptive streaming over HTTP," in *Proc. 19th ACM Int. Conf. Multimedia*, Scottsdale, AZ, USA, 2011, pp. 723–726.

[32] J. Emigh, "New flash player rises in the web-video market," *IEEE Comput.*, vol. 39, no. 2, pp. 14–16, Feb. 2006.

[33] B. Rainer and C. Timmerer, "Quality of experience of web-based adaptive HTTP streaming clients in real-world environments using crowdsourcing," in *Proc. ACM Workshop Design Qual. Deploy. Adapt. Video Stream.*, Sydney, NSW, Australia, 2014, pp. 19–24.

[34] L. Yan and G. Yang, "A web video service based on flash video technique," in *Proc. IEEE Int. Conf. Internet Technol. Appl. (iTAP)*, Wuhan, China, 2011, pp. 1–4.

[35] G. Zhu, F. Zhang, W. Zhu, and Y. Zheng, "HTML5 based media player for real-time video surveillance," in *Proc. Image Signal Process. (CISP)*, Chongqing, China, 2012, pp. 245–248.

[36] K. Kapetanakis, S. Panagiotakis, A. G. Malamos, and M. Zampoglou, "Adaptive video streaming on top of Web3D: A bridging technology between X3DOM and MPEG-DASH," in *Proc. IEEE Int. Conf. Telecommun. Multimedia (TEMU)*, Heraklion, Greece, 2014, pp. 226–231.

[37] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats*, ISO/IEC Standard 23009-1, 2014.

[38] T. Stockhammer, "Dynamic adaptive streaming over HTTP—Standards and design principles," in *Proc. ACM Conf. Multimedia Syst.*, Portland, OR, USA, 2011, pp. 133–144.

[39] E. Quacchio, G. Bruno, and M. Grangetto, "An HTML5 player for a gstreamer based MPEG DASH client," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, San Diego, CA, USA, 2012, p. 1.

[40] S. Lederer *et al.*, "Distributed DASH dataset," in *Proc. ACM 4th Multimedia Syst. Conf.*, Oslo, Norway, 2013, pp. 131–135.

[41] J. Le Feuvre, J.-M. Thiesse, M. Parmentier, M. Raulet, and C. Daguet, "Ultra high definition HEVC DASH data set," in *Proc. 5th ACM Multimedia Syst. Conf.*, Singapore, 2014, pp. 7–12.

[42] DASH-IF. (Apr. 2013.). *Dash Industry Forum'S Reference Client 1.0.0.* [Online]. Available: http://dashif.org/reference/players/javascript/1.0.0/index.html

[43] H. Kwan, C. Roy, and D. Das, "Demonstration of a multimedia player supporting the MPEG-DASH protocol," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, San Diego, CA, USA, 2012, p. 1.

[44] C. Concolato and J. Le Feuvre, "Live HTTP streaming of video and subtitles within a browser," in *Proc. ACM 4th Multimedia Syst. Conf.*, Oslo, Norway, 2013, pp. 146–150.

[45] D. Van Deursen, W. Van Lancker, E. Mannens, and R. Van de Walle, "Experiencing standardized media fragment annotations within HTML5," *Multimedia Tools Appl.*, vol. 70, no. 2, pp. 827–846, 2014.

[46] N. Bouzakaria, C. Concolato, and J. Le Feuvre, "Overhead and performance of low latency live streaming using MPEG-DASH," in *Proc. IEEE Int. Conf. Inf. Intell. Syst. Appl. (IISA)*, Chania, Greece, 2014, pp. 92–97.

[47] J. Hwang, J. Lee, N. Choi, and C. Yoo, "HAVS: Hybrid adaptive video streaming for mobile devices," *IEEE Trans. Consum. Electron.*, vol. 60, no. 2, pp. 210–216, May 2014.

[48] D. K. Krishnappa, D. Bhat, and M. Zink, "DASHing YouTube: An analysis of using DASH in YouTube video service," in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, Sydney, NSW, Australia, 2013, pp. 407–415.

[49] C. Wang and M. Zink, "On the feasibility of DASH streaming in the cloud," in *Proc. ACM Netw. Oper. Syst. Support Digit. Audio Video Workshop*, Singapore, 2014, p. 49.

[50] V. Adzic, H. Kalva, and B. Furht, "Optimizing video encoding for adaptive streaming over HTTP," *IEEE Trans. Consum. Electron.*, vol. 58, no. 2, pp. 397–403, May 2012.

[51] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," *IEEE Trans. Consum. Electron.*, vol. 58, no. 1, pp. 78–85, Feb. 2012.

[52] Y. Zhao, X. Gong, W. Wang, and X. Que, "A rate adaptive algorithm for HTTP streaming," in *Proc. Cloud Comput. Intell. Syst. (CCIS)*, Hangzhou, China, 2012, pp. 529–532.

[53] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.

[54] D. Corujo, M. Lebre, D. Gomes, and R. L. Aguiar, "Furthering media independence mechanisms for future internet enablement," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Ottawa, ON, Canada, 2012, pp. 6845–6849.

[55] G. Gang, L. Zeyong, and J. Jun, "Internet of things security analysis," in *Proc. Internet Technol. Appl. (iTAP)*, Wuhan, China, 2011, pp. 1–4.

[56] P. Martinez-Julia, E. T. Garcia, J. O. Murillo, and A. F. Skarmeta, "Evaluating video streaming in network architectures for the internet of things," in *Proc. IEEE Int. Conf. Innov. Mobile Internet Serv. Ubiq. Comput. (IMIS)*, Taichung, Taiwan, 2013, pp. 411–415.

[57] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. Chichester, U.K.: Wiley, 2009.

[58] N. Kushalnagar, G. Montenegro, and C. Schumacher, *IPv6 Over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*, IETF Standard RFC 4919, Aug. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4919.txt

[59] K.-I. Hwang, J. In, N. Park, and D.-S. Eom, "A design and implementation of wireless sensor gateway for efficient querying and managing through world wide web," *IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 1090–1097, Nov. 2003.

[60] S. Nichols and K. A. Hua, "Design and implementation of intelligent mesh nodes for wireless video stream sharing," in *Proc. 3rd Int. Conf. Internet Multimedia Comput. Serv. (ICIMCS)*, Chengdu, China, 2011, pp. 13–16.

[61] G. Song, Y. Zhou, W. Zhang, and A. Song, "A multi-interface gateway architecture for home automation networks," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 1110–1113, Aug. 2008.

[62] D. Thommes, A. Gerlicher, Q. Wang, and C. Grecos, "RemoteUI: A high-performance remote user interface system for mobile consumer electronic devices," *IEEE Trans. Consum. Electron.*, vol. 58, no. 3, pp. 1094–1102, Aug. 2012.

[63] J. Vasseur and A. Dunkels, *Interconnecting Smart Objects With IP: The Next Internet*. Burlington, MA, USA: Morgan Kaufmann, 2010.

[64] F. Curbera *et al.*, "Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI," *IEEE Internet Comput.*, vol. 6, no. 2, pp. 86–93, Mar./Apr. 2002.

[65] G. Moritz, F. Golatowski, and D. Timmermann, "A lightweight SOAP over CoAP transport binding for resource constraint networks," in *Proc. Mobile Adhoc Sensor Syst. (MASS)*, Valencia, Spain, 2011, pp. 861–866.

[66] B. Upadhyaya, Y. Zou, H. Xiao, J. Ng, and A. Lau, "Migration of SOAP-based services to RESTful services," in *Proc. Web Syst. Evol. (WSE)*, Williamsburg, VA, USA, 2011, pp. 105–114.

[67] R. E. Sward and J. Boleng, "Service-oriented architecture (SOA) concepts and implementations," *ACM Ada Lett.*, vol. 31, no. 3, pp. 3–4, Dec. 2011.

[68] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, “Adaptive multimedia streaming in information-centric networks,” *IEEE Netw.*, vol. 28, no. 6, pp. 91–96, Nov./Dec. 2014.

[69] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, “Offloading cellular networks with information-centric networking: The case of video streaming,” in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, San Francisco, CA, USA, 2012, pp. 1–3.

[70] A. Detti, B. Ricci, and N. Blefari-Melazzi, “Peer-to-peer live adaptive video streaming for information centric cellular networks,” in *Proc. IEEE 24th Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, London, U.K., 2013, pp. 3583–3588.

[71] A. Detti, B. Ricci, and N. Blefari-Melazzi, “Supporting mobile applications with information centric networking: The case of P2Plive adaptive video streaming,” in *Proc. ACM SIGCOMM Workshop Inf. Centr. Netw.*, Hong Kong, 2013, pp. 35–36.

[72] A. Awad, R. German, and F. Dressler, “Exploiting virtual coordinates for improved routing performance in sensor networks,” *IEEE Trans. Mobile Comput.*, vol. 10, no. 9, pp. 1214–1226, Sep. 2011.

[73] M. Caesar, M. Castro, E. B. Nightingale, G. O’Shea, and A. Rowstron, “Virtual ring routing: Network routing inspired by DHTs,” *ACM SIGCOMM Comput. Commun. Rev. Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, vol. 36, no. 4, pp. 351–362, Oct. 2006.

[74] M.-J. Tsai, H.-Y. Yang, B.-H. Liu, and W.-Q. Huang, “Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1228–1241, Aug. 2009.

[75] D. E. Knuth, “Backus normal form vs. Backus Naur form,” *Comm. ACM*, vol. 7, no. 12, pp. 735–736, Dec. 1964.

[76] E. Bidelman, *Using the HTML5 Filesystem API*. Sebastopol, CA, USA: O’Reilly, 2011.

[77] M. J. Collins, “Indexed DB,” in *Pro HTML5 With Visual Studio 2012*. New York, NY, USA: Apress, pp. 255–279.

[78] B. Nicolae, G. Antoniu, L. Bougé, D. Moise, and A. Carpen-Amarie, “BlobSeer: Next-generation data management for large scale infrastructures,” *J. Parallel Distrib. Comput.*, vol. 71, no. 2, pp. 169–184, Feb. 2011.

[79] S. Gundavaram, *CGI Programming on the World Wide Web*. Bonn, Germany: O’Reilly, 1996.

[80] F. Fund *et al.*, “Performance of DASH and WebRTC video services for mobile users,” in *Proc. Int. Packet Video Workshop*, San Jose, CA, USA, 2013, pp. 1–8.

[81] I. Kofler, R. Kuschnig, and H. Hellwagner, “Implications of the ISO base media file format on adaptive HTTP streaming of H.264/SVC,” in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2012, pp. 549–553.

[82] L. Schwoebel, “Browser- and codec-agnostic HTML5 video-streaming in the wireless video sensor network platform,” M.S. thesis, Dept. Electr. Comput. Energy Eng., Arizona State Univ., Tempe, AZ, USA, 2014.

[83] Z. Aouini, M. T. Diallo, A. Gouta, A.-M. Kermarrec, and Y. Lelouedec, “Improving caching efficiency and quality of experience with CF-Dash,” in *Proc. Netw. Oper. Syst. Support Digit. Audio Video Workshop (NOSSDAV)*, Portland, OR, USA, 2014, p. 61.

[84] S. Colonnese, F. Cuomo, R. Guida, and T. Melodia, “Performance evaluation of sender-assisted HTTP-based video streaming in wireless ad hoc networks,” *Ad Hoc Netw.*, vol. 24, pp. 74–84, Jan. 2015.

[85] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, “ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH),” in *Proc. IEEE Packet Video Workshop*, San Jose, CA, USA, 2013, pp. 1–8.

[86] S. Garcia, J. Cabrera, and N. Garcia, “Quality-optimization algorithm based on stochastic dynamic programming for MPEG DASH video streaming,” in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, 2014, pp. 574–575.

[87] I. Irondi, Q. Wang, and C. Grecos, “Empirical evaluation of H.265/HEVC-based dynamic adaptive video streaming over HTTP (HEVC-DASH),” *Proc. SPIE*, vol. 9139, May 2014, Art. ID 91390L.

[88] J. Kovacevic, G. Miljkovic, K. Lazic, and M. Stankic, “Evaluation of adaptive streaming algorithms over HTTP,” in *Proc. IEEE Int. Conf. Consum. Electron. Berlin (ICCE)*, Berlin, Germany, 2013, pp. 1–4.

[89] S. Lederer, C. Mueller, B. Rainer, C. Timmerer, and H. Hellwagner, “Adaptive streaming over content centric networks in mobile networks using multiple links,” in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Budapest, Hungary, 2013, pp. 677–681.

[90] L. Pozueco *et al.*, “Adaptation engine for a streaming service based on MPEG-DASH,” *Multimedia Tools and Applications*. New York, NY, USA: Springer, 2014, pp. 1–20.

[91] L. Rovcanin and G.-M. Muntean, “A DASH-based performance-oriented adaptive video distribution solution,” in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, London, U.K., 2013, pp. 1–7.

[92] G. Zhong and A. Bokani, “A geo-adaptive JavaScript DASH player,” in *Proc. ACM Workshop Design Qual. Deploy. Adapt. Video Stream.*, Sydney, NSW, Australia, 2014, pp. 39–40.

[93] F. Aurzada, M. Lévesque, M. Maier, and M. Reisslein, “FiWi access networks based on next-generation PON and gigabit-class WLAN technologies: A capacity and delay analysis,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1176–1189, Aug. 2014.

[94] M. Hossen and M. Hanawa, “Multi-OLT and multi-wavelength PON-based open access network for improving the throughput and quality of services,” *Opt. Switch. Netw.*, vol. 15, pp. 148–159, Jan. 2015.

[95] H.-S. Yang, M. Maier, M. Reisslein, and W. M. Carlyle, “A genetic algorithm-based methodology for optimizing multiservice convergence in a metro WDM network,” *IEEE/OSA J. Lightw. Technol.*, vol. 21, no. 5, pp. 1114–1133, May. 2003.

[96] N. Zaker, B. Kantarci, M. Erol-Kantarci, and H. T. Mouftah, “Smart grid monitoring with service differentiation via EPON and wireless sensor network convergence,” *Opt. Switch. Netw.*, vol. 14, pp. 53–68, Aug. 2014.

Adolph Seema (S’03–M’10) received the B.S. degree in computer engineering from the Rochester Institute of Technology, NY, USA, and the M.S. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, where he is currently pursuing the Ph.D. degree from the School of Electrical, Computer, and Energy Engineering. He is a Crypto and Security Design Engineer with Freescale Semiconductor, Chandler, AZ, USA.

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.

Tejas Shah received the M.S. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2014.

Jeffery Morgan received the B.S. degree in computer science from Arizona State University, Tempe, AZ, USA, in 2014. He is a Software Developer with IT Innovation Center, General Motors, Chandler, AZ, USA.

Martin Reisslein (A’96–S’97–M’98–SM’03–F’14) received the Ph.D. degree in systems engineering from the University of Pennsylvania in 1998. He is a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe.