

# Enabling Green Video Streaming over Internet of Things

(7<sup>th</sup> Quarter Deliverable)

Dr. Ghalib Asadullah Shah (PI)

Next-generation Wireless Networking (NWN) Lab,  
Al-Khawarizmi Institute of Computer Science,  
University of Engineering and Technology, Lahore

11-09-2015

# About this Document

This document reports the activities performed in the 7th quarter of our project ‘Enabling Green Video Streaming over Internet of Things’ and the corresponding deliverable to be submitted to ICT R & D Fund.

In the 1st deliverable, we conducted a detailed literature survey of IoT communication protocols and identified the open challenges for IoM. One of these challenges was to enable green and high data rate communication for IoM. For this matter, a set of energy efficient and green communication MAC and routing protocols were proposed in 4th deliverable. The performance analysis of these communication protocols was done using the simulation studies in the 5th and 6th deliverables. We implemented our proposed energy efficient power saving MAC protocol (IEEE 802.11+) in NS-2 network simulator, and the green routing protocol (Green-RPL) was implemented in Cooja simulator. The performance analysis of these protocols suggested high gains in energy efficiency.

In this (7th) deliverable, we provide the implementation specifications for the our proposed IEEE 802.11+ power saving MAC protocol and Green-RPL routing protocol, also the implementation details for video acquisition and encoding algorithm. Our proposed power saving MAC protocol IEEE 802.11+ is implemented in Contiki-OS. The Contiki-OS is run on Atmel SAM3X8E ARM Cortex-M3 CPU based micro-controller. The IEEE 802.11 MAC layer is provided by the Atheros AR9170 wifi chip. We have compared the performance of a recently proposed power saving protocol MH-PSM and our proposed IEEE 802.11+ in a real implementation. The analysis done shows significant improvement in energy efficiency. Similarly, our proposed Green-RPL routing protocol is also implemented in Contiki-OS. The implementation required some modification and addition in the already existing RPL routing protocol implementation in Contiki-OS.

Moreover, in the previous deliverables, we identified the issue of complex multimedia encoding, which restricts its implementation in IoM. Therefore, we proposed a very simple encoder that needs to be run at the low power source with the encoded stream decoded on powerful machines at the receiving end. For this reason, we proposed to reduce video encoder complexity (both space and time), by the use of compressed sensing, while maintaining fairly low bit-rates (for transmission). In this (7th) deliverable, we have implemented the video acquisition and video compression in bare-metal C on the 168MHz STM32F4 Discovery board.

---

Both of these MAC and protocol implementation and multimedia acquisition and encoding algorithms are combined and transformed into a green camera node. This green camera node acquires the video, compress it, and transmit it in an energy efficient manner. The C language code for both of the above mentioned implementations of power saving MAC and Green-RPL routing protocol and video coding algorithm are given in the CD.

# Contents

<b>1</b>	<b>Development of Green Multi-hop Routing Protocol</b>	<b>6</b>
1.1	Introduction . . . . .	6
1.2	Development of Green-RPL . . . . .	7
1.2.1	Implementation . . . . .	7
1.3	Conclusion . . . . .	9
<b>2</b>	<b>Implementation of IEEE 802.11+ in Contiki-OS</b>	<b>10</b>
2.1	Introduction . . . . .	10
2.2	Preliminaries . . . . .	11
2.2.1	Contiki80211: Background . . . . .	11
2.2.2	MH-PSM . . . . .	12
2.3	IEEE 802.11+ implementation . . . . .	12
2.4	Results . . . . .	14
<b>3</b>	<b>Implementation of Change-based Block Suppression video coding</b>	<b>17</b>
3.1	Introduction . . . . .	17
3.2	Video Acquisition and Storage . . . . .	18
3.2.1	Data transfer from DCMI to SD card . . . . .	19
3.2.2	Conversion from Inter-leaved YUV to Planer YUV, and Buffer size . . . . .	20
3.2.3	Block Columnization . . . . .	20
3.2.4	Compressed Sensing and Further Compression . . . . .	20
3.2.5	Pass Data to Transmission Module using SPI . . . . .	21
3.2.6	Reconstruction and Recovery . . . . .	21
3.2.7	Results . . . . .	22
<b>4</b>	<b>Conference paper on energy efficiency in IEEE 802.11 for IoM</b>	<b>25</b>

# List of Figures

2.1	Our Hardware Platform . . . . .	11
2.2	Intervals in which node stays in sleep state. . . . .	16
2.3	Energy consumed by node in awake state. . . . .	16
3.1	Nested video encoding-decoding. . . . .	18
3.2	The Change based block Suppression Architecture(CBS). . . . .	18
3.3	mPSMP Operation for a 5 Node Network . . . . .	23
3.4	mPSMP Operation for a 5 Node Network . . . . .	23
3.5	mPSMP Operation for a 5 Node Network . . . . .	23
3.6	mPSMP Operation for a 5 Node Network . . . . .	24

# List of Tables

2.1	Comparison of IEEE 802.11+ and MH-PSM scheme . . . . .	15
-----	--	----

# Chapter 1

## Development of Green Multi-hop Routing Protocol

### 1.1 Introduction

In an IoT system the sensor devices are deployed in large numbers and mostly these are battery operated. Thus, their operation is must be extremely energy efficient to prolong the network lifetime. With the help of a multi-hop routing protocol, the network nodes communicate with the sink node via different multi-hop paths. Note that not every path consumes same amount of energy in LLNs. Since, nodes can experience different channel conditions or the network topology can also contribute towards the number of transmissions a packet requires before it is successfully transmitted towards the sink node. However, if a routing protocol efficiently selects those paths which consume less energy then significant amount of energy can be saved.

Moreover, the multimedia communication is bulky in nature and require large number of packet transmission. Thus, the multimedia devices are more energy hungry and their energy efficient operation is very critical for longer network lifetime. Although many previous studies have addressed energy efficiency issue in resource constrained networks. However, in prior studies have not considered multimedia communication over the resource constrained devices in an IoT system. Although lot of work is done on green communication for various wireless networking technologies, yet enabling green communication over IoT systems is not given any consideration.

Routing Protocol for Low Power and Lossy Networks (RPL) is a proactive distance vector routing protocol for LLNs. RPL forms a tree like network topology by maintaining a directed acyclic graph (DAG). In RPL each sensor node chooses a preferred parent towards the root node based on specific routing policies. RPL uses multiple routing metrics and constraints while optimizing an objective function to select the best path. The standard provides the choice to select appropriate

objective functions as per the application requirements, which makes RPL highly adaptive and dynamic. However, so far no optimizations have been made for RPL to support multimedia communication.

In this work, we implement an energy efficient green routing protocol for resource constrained multimedia devices in IoM named Green-RPL. The proposed routing protocol is an enhanced version of the RPL protocol that is designed for IoT systems. In our proposed Green-RPL implementation the carbon footprints emission is minimized provided that the application delay requirement and energy efficiency is guaranteed. The delay bound for multimedia applications is pre-determined. For a video application the frames can be of different sizes yet the transmitter node must ensure that 25 frames are successfully transmitted in 1 sec. Similarly, the energy efficiency is ensured by considering the quality of the intermediate links towards the root node, the energy already consumed by the possible preferred parent node, and by evaluating the potential of the parent node to support traffic requirements for yet another child node. To evaluate a parent node as per these constraints and requirements, an optimization model for the proposed Green-RPL routing protocol is designed in the following part of this section. Among all the parent nodes of a specific sensor node, the solutions of the optimization problem gives the preferred parent.

## 1.2 Development of Green-RPL

### 1.2.1 Implementation

The proposed Green-RPL routing protocol is implemented in Contiki v2.7. Contiki is a wireless sensor network operating system (OS) and consists of the kernel, libraries, the program loader, and a set of processes. It is the most popular operating system for IoT things and it has been vastly used by the research community for simulation and real-implementation of IoT based wireless networks. Contiki provides mechanisms that assist in programming the smart object applications. It provides libraries for memory allocation, linked list manipulation and communication abstractions. It is developed in C, all its applications are also developed in C programming language, and therefore it is highly portable to different architectures. The Contiki operating system provides modules for different tasks (layers).

Contiki-OS contains a complete communication stack that is proposed by the research community for the IoT. The RPL routing protocol functionality is divided into multiple files each performing one of the major tasks of routing operation. RPL routing protocol files and their functionality are as follows:

- `rpl-dag.c` is responsible for creating and interpreting the DODAG frames,
- `rpl-timers.c` contains all the timers for the transmission of DIO and other control packets,



- `rpl-private.h` contains the definition of routing parameters,
- `rpl-of0.c` implements the functionality of hop-count based objective function,
- `rpl-mrhof.c` implements the functionality of ETX based objective function,
- `rpl-icmp6.c` specifies the structure of formats of the control packets,
- `rpl-ext-header.c` implements the header configuration for the RPL control packets,
- `rpl-conf.c` contains the configuration settings for RPL such as which objective function is used and so on,
- `rpl.c` combines the whole functionality of RPL protocol by utilizing the functionality of other files.

This division of RPL routing protocol functionality makes it easier to program different objective functions and tuning the values of key parameters to modify the routing protocols performance. To implement our Green-RPL routing protocol we need to add another objective function in the existing RPL implementation. The new objective function is named ENERGY ROUTE. New routing metrics i.e., energy, idle time, path cf, are integrated in the `rpl.h` file in the `rpl` metric container structure. Moreover, already existing metrics i.e., `etx`, `avg delay to sink`, will also be utilized in the Green-RPL routing operation. For energy and idle time metrics, Contiki's build in `energest` model is employed. The `energest` model determines the instantaneous amount of time a node spent in listen, transmit, sleep, and idle states. Since, each network module designed by a specific vender draws a predetermined amount of current at each state. Therefore, using the `energest` model the energy utilized by a particular node can be computed. Moreover, the IoT wireless nodes usually operate using specific batteries such as AA-battery with 2700mAh capacity. Thus, we combined the `energest` model with a given set of values for current, voltage, and battery capacity, in order to compute the energy consumed by a node and accordingly the energy consumed by the route by aggregating the energy of individual nodes. For this implementation the changes were made in `energest.c` file that is one of the system files for the Contiki-OS.

The local routing metrics calculated by a network node needs to be shared with its neighboring nodes. In RPL routing protocol the routing metrics are exchanged via the DIO control packets that is transmitted periodically at specific intervals. Besides default parameters, we integrated idle time, path cf, and route energy, in the DIO packet. This information is received by the child nodes which evaluate the possible parent node using the constraints mentioned in the optimization model presented in previous section. If the constraints are satisfied then the path metric is calculated that is the path cf (carbon footprints of the route via considered possible parent node). The implementation is done in `calculate path metric()` function in the `rpl-mrhof.c` file. Once the path metric for a possible parent node

is calculated, then it is compared with the path metric of current parent node. If the new path metric is lower than the current parent node then the new parent is selected and vice versa.

### 1.3 Conclusion

In this deliverable an enhanced version of RPL for IoM, Green-RPL, is implemented in Contiki-OS. The proposed Green-RPL routing protocol minimizes carbon footprints emission and energy consumption, and supports application specific QoS requirements by considering various constraints while selecting routes towards the root node. The performance of the proposed Green-RPL scheme was extensively studied in previous deliverable with the help of a simulation study is carried out in Cooja simulator for Contiki-OS. Its performance was compared with the existing objective functions and Green-RPL performed significantly better. The same code implemented in Contiki-OS is then programmed on an Atmel SAM3X8E ARM Cortex-M3 CPU based micro-controller, that is component of the green camera node.

# Chapter 2

## Implementation of IEEE 802.11+ in Contiki-OS

### 2.1 Introduction

We have implemented our proposed power saving mechanism, i.e. IEEE 802.11+ protocol, in Contiki Operating System (Contiki OS) which is one of the most popular operating systems for embedded systems and IoT. To achieve this purpose, an open source IEEE 802.11 radio link layer implementation for Contiki-OS named as Contiki80211 is used. The Contiki80211 is proposed in [1] to enable experimentation with 802.11 MAC layer management mechanisms on embedded devices, such as sensor motes and IoT smart objects. Recently, a multi-hop IEEE 802.11 PSM mechanism, named as MH-PSM, is proposed in [2] whereby authors introduced a traffic announcement scheme which facilitates multi-hop communication. We modified the implementation of MH-PSM in Contiki-OS and compared our proposed power saving mechanism IEEE 802.11+ with MH-PSM implementation in Contiki-OS. This comparison proved that our IEEE 802.11+ protocol is more energy efficient than MH-PSM protocol.

In subsequent section, we give essential details of Contiki80211 implementation along with the IEEE 802.11+ protocol we implemented in Contiki80211 to accomplish our goal, i.e. to decrease the duty cycling of a mote (station) as much as possible. The fundamental idea behind our implementation is to enable multimedia traffic communication among stations while reducing their duty cycling period and keeping the packet loss rate within acceptable limitations.

## 2.2 Preliminaries

### 2.2.1 Contiki80211: Background

Contiki80211 is an open source implementation running on Atmel SAM3X8E ARM Cortex-M3 CPU based micro-controller, Fig.2.1. The hardware configuration for the Atmel SAM3X8E ARM Cortex-M3 MCU is 96 KB SRAM, 512 KB Flash. WiFi is connected through the USB bus and Cortex-M3 MCU. Contiki80211 solutions uses Atheros WiFi + AR9104, AR9170AR9170is 802.11n draft standard, support USB2.0 baseband AR9104processor,supports dual-band RF. Contiki80211 is composed of three main components which include IEEE80211Lib, AR9170 radio driver and AR9170 USB driver.

- **IEEE80211Lib** is a platform independent library which connects the uIP protocol stack with the radio link layer. It implements various IEEE 802.11 IBSS management operations such as scan, join, create and leave and IBSS parameter configuration. The main element of IEEE80211Lib is IEEE802.11 Scheduler which encapsulate IP Packet into IEEE802.11 frame and deliver them to AR9170 Driver and vice versa do the same for incoming frames. Duplicate frame detection by ieee802.11 module is also incorporated. Finally, it implements MH-PSM algorithm, details of which are given in next sub-section, and is responsible for ATIM Packet creation/parsing and maintaining the database of list of awake nodes.
- **AR9170 radio driver** manages the transmission (Tx) and reception (Rx) queues and handles the hardware command responses from/to the 802.11 interface. The main element of this component is AR9170 Scheduler, which at each execution round, inspects the contents of commands, Tx and Rx queues and dispatch next in-line task prioritizing command over Tx/Rx packet processing. It access Tx and Rx queues in round robin fashion to ensure fairness in packet handling. AR9170 waits for Pre-TBTT interrupt

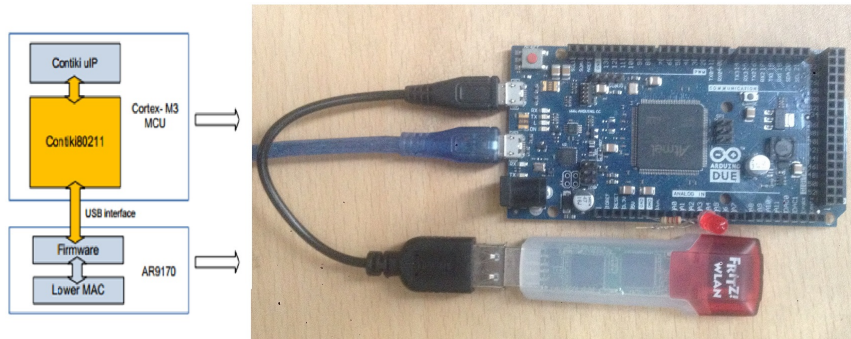


Figure 2.1: Our Hardware Platform

from 802.11 interface in order to prepare 802.11 protocol for upcoming ATIM Window. AR9170 poll handler integrate the AR9170 Scheduler and PSM Scheduler and implements lower level PSM functionality like radio duty-cycling.

- **AR9170 USB driver** is implemented using Atmel USB Host library which provide platform independent routines for installation of USB Devices, enumerations and allocation of required endpoints for communication with devices with interface over USB. It includes low level USB on-the-go drivers and on top of these routines, functions for reading and writing from/to allocated endpoints are implemented.

Thus, the main functionalities of Contiki80211 include Contiki80211 drivers, based on Linux open source drivers carl9170 (a Linux kernel driver supporting the Atheros AR9170 802.11 draft-n USB chipset), developed and optimized. IEEE 802.11 MAC Layer Management Entity (MLME) state machine that implements the Ad hoc mode and PSM mechanism and micro IPv6(uIPv6) implementation.

### **2.2.2 MH-PSM**

MH-PSM is a multi-hop power saving saving mechanism based on IEEE802.11 PSM standard. This scheme propagates Announcement Traffic Indication Messages (ATIM) along multi-hop paths to ensure that all intermediate nodes remain awake to forward the pending data frames within a single beacon interval. The focus of MH-PSM is to reduce latency and decrease end-to-end delay in multi-hop networks. However, a fundamental drawback of MH-PSM scheme occurs in situations when the received ATIM packet is not followed by data packet. This happens when a packet generated at application layer is not yet prepared at underlying layers to be transmitted in current beacon interval. Moreover, a node is supposed to stay awake after the reception of data packet for whole beacon interval.

Consequently, MH-PSM protocol although reduces the end-to-end delay over multi-hop nodes compared to standard PSM mechanism, but the possibility that ATIM packet may not be followed by data packet immediately is not perceived which results in wastage of energy since MH-PSM mandates each node to stay awake for whole beacon interval. This increases the wake-up time of each node consequently more energy will be consumed.

## **2.3 IEEE 802.11+ implementation**

In our proposed power saving mechanism, named IEEE 802.11+, we define a PS-Frame to let the receiver know that it should expect its traffic in current Beacon Interval. When a transmitting node intends to send traffic to receiving node, it sends a PS-Frame to that node in start of Beacon Interval so that the receiving

node stays awake to receive the data traffic. A node which receives PS-Frame stays awake till the reception of data packets and consequently, goes to doze state for the remaining duration of Beacon Interval.

In MH-PSM, the ATIM packet is transmitted to keep the receiving station in awake state. However, the sizes of ATIM Window and Transmission Window are kept equal. Thus, it is obligatory for each station to remain awake at-least for half of the beacon interval duration even if it has no traffic scheduled in current beacon interval. On the contrary, we modified the duration of ATIM Window in Contiki80211 (i.e. PS-Frame Window Duration) to 25% of beacon interval (50ms when beacon interval is 200ms). This helps in reducing the duty-cycling of a network node for which traffic is not scheduled in a given Beacon Interval.

Furthermore, in proposed power saving mechanism implementation, we address the issues highlighted in previous sub-section and allows a node to go into sleep mode whenever the received PS-Frame packet is not followed by data packet at the end of PS-Frame Window Duration. This is done by setting a flag at the end of PS-Frame Duration. Moreover, IEEE 802.11+ includes adaptability features which allows a station to go into doze state as soon as it receives its pending traffic.

In order to further minimize the duty-cycling, a node is allowed to go into sleep mode by considering the number of pending packets in packet reception queues at each station. When a node receives a packet it checks number of pending packets in reception queue and if there is only single packet left in reception queue then it schedules the power saving mode to turn on, and hence, goes into sleep state for the remaining duration of beacon interval. This helps in enhancing the sleep duration of each node. Finally, Beacon Interval, PS-Frame Window Duration and inter-packet arrival rate are modified and adjusted so as to reduce the traffic loss to minimum.

To carry out above mentioned protocol operations, we proposed several changes in Contiki80211 implementation. Firstly, in IEEE80211Lib, following C files are changed:

- **ieee80211\_rx.c** Within 'ieee80211\_rx\_process-data\_mpd()' function, after processing the current data packet we check if the number of remaining packet in pending reception queue linked list is only 1. If this is true, then a node is allowed to go into sleep state to conserve energy for remaining beacon interval.
- **ieee80211\_ibss.c** Within the IBSS mode implementation, BEACON\_INTERVAL and PS-Frame\_Window\_Duration (ATIM\_Window\_Duration) are modified.

In AR9170 radio driver, following changes are implemented:

- **ar9170.h** A flag named as 'recv\_ps\_frame\_flag' is declared within ps manager structure to avoid multiple callbacks to power saving function if a node is already scheduled to go into sleep state.

- **ar9170\_psm.c** 'recv\_ps.frame\_flag' is checked at the end of PS-Frame Window to determine if the pending packets in reception queue of given node is zero. If this flag is not set, then it essentially mean that currently there is no packet to be received in current beacon interval so node's power saving state (ps.state) is set to true to make the node go into sleep state.
- **rtimer.c** rtimer\_set() function is used to mark the end of PS-Frame Window duration and schedule the change in power state of a node afterwards within ar9170\_psm.c file.
- **ar9170\_scheduler.c:** The function 'ar9170\_sch\_powersave\_check()' in AR9170-Scheduler is modified to check and modify the current power state of a node multiple times within one beacon interval.
- **ar9170rx.c:** The functional routine which handles the command responses from the ar9170 device, 'ar9170\_handle\_command\_response()', to awake the node at pre-target-beacon-transmission-time and to calculate sleep duration of each node in one beacon interval.

In AR9170 USB driver following C files are changed:

- **usb\_cmd\_wrapper.c:** Doze\_ACTIVE\_PIN is set to high for LED Debugging, several header files are included for this purpose. Callbacks to function ar9170\_powersave() is modified and whenever a node is intended to go into sleep state then instead of scheduling the 'ar9170\_psm\_schedule\_powersave()' function, ar9170\_powersave() is directly called to save the scheduling time.

Finally, 'Avgsleep.h' Header file is created which contains declaration of several variables for average sleep duration calculation. This header file is included externally in several C files namely 'usb\_cmd\_wrapper.c', 'udp-server.c', 'ar9170rx.c'.

## 2.4 Results

The findings of the real implementation are given in Table 2.1, Figure 2.2 and Figure 2.3. The experiment was carried out for 316.8 secs. The video acquired at a sender node is compressed and then it is transmitted to the receiver node which is operated in power saving mode. We did many experiments and found the similar results as summarized in Table 2.1.

Firstly, the power saving node is operated with our proposed IEEE 802.11+ protocol and then the same network scenario is run with MH-PSM power saving protocol. As shown in Table 2.1, in our proposed scheme the node stays longer in sleep state and as a result the energy is saved. Alternately, the time spent in awake state is decreased which essentially means that the node consumes lesser energy.

Similarly, in Figure 2.2 and Figure 2.3, in our 802.11+ scheme the node stays longer in sleep state and the average time spent in sleep is far more as compared to MH-PSM. Correspondingly, the energy utilized in awake state is higher in MH-PSM as compared to 802.11+.

Table 2.1: Comparison of IEEE 802.11+ and MH-PSM scheme

<i>Parameters</i>	<i>IEEE 802.11+</i>	<i>MH-PSM</i>	<i>comments</i>
Experimentation Time	316.8 secs	316.8 secs	system active time
Time spent in Sleep state	192.6 secs	150.8 secs	more is good
Energy spent in Awake state	62000 mJ	83000 mJ	less is good
Duty-cycling (Radio ON time)	39.2 %	52.5 %	less is good
Energy consumed per sec	196.2 mJ	262.6 mJ	less is good



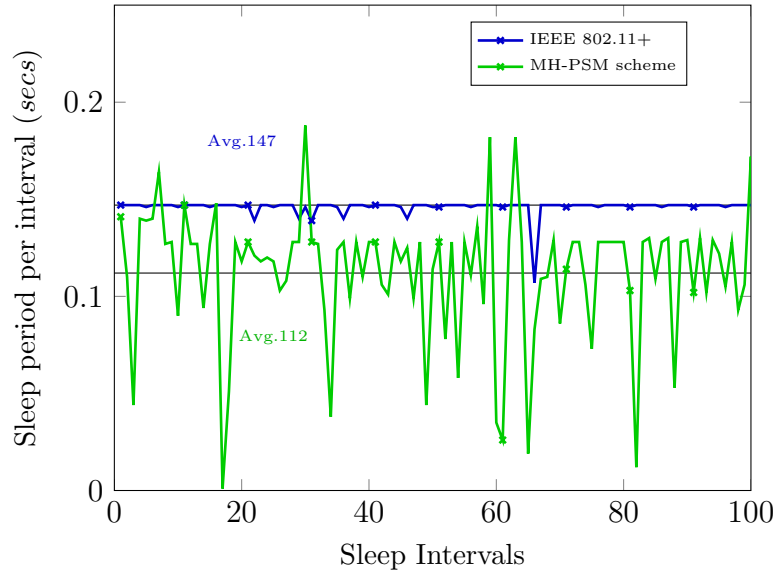


Figure 2.2: Intervals in which node stays in sleep state.

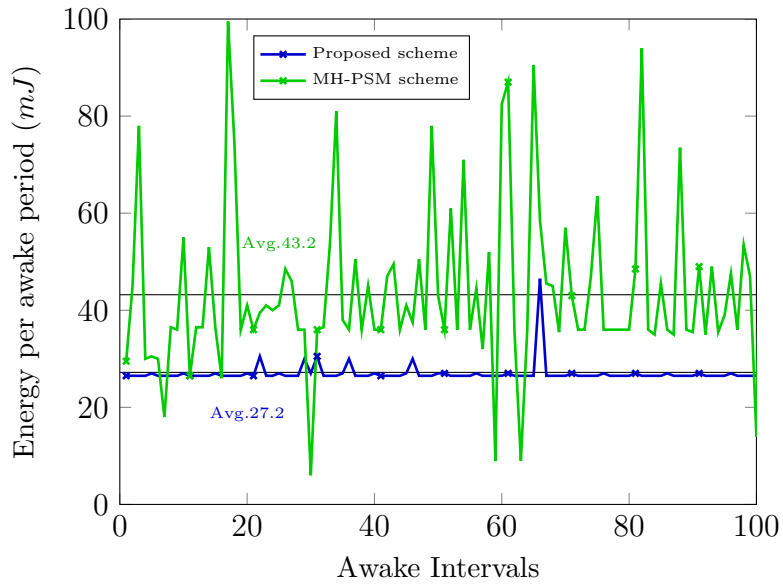


Figure 2.3: Energy consumed by node in awake state.

# Chapter 3

## Implementation of Change-based Block Suppression video coding

### 3.1 Introduction

Development of low complexity video coding techniques/algorithms has been accelerated by the expected increase in the up-link streaming from IoT (Internet of Things) enabled multimedia devices(IoM devices), and multimedia devices in general. Secondly, severe constraints on the cost, power, bandwidth and computational capability of IoM devices bar the usage of current video coding techniques(standards) such as H.265 and VP9 due to their high computational complexity on the encoder side. The high encoder complexity of H.265 and VP9 is because they are designed for down-link streaming applications. However, recently emergent techniques from the area of Compressive Sensing(CS), have given rise to a new generation of low complexity video encoders. Unlike traditional video coding, in compressive sensing based techniques, a small number of linear measurements of the scene are taken at sub-Nyquist rates, at the acquisition stage, before being passed on to the encoder. Thus, compression is inherent in the acquisition process. In order to match IoM device constraints, we have implemented a new video compression technique, which shifts the complexity from encoder to decoder.

We have designed an adaptive video encoding/decoding scheme in which compressed sensed video data is further compressed within the CS domain. In this technique we have introduced a nested approach in which CS frames are further compressed(during and) after acquisition on the encoder side, and on the decoder side the full CS frames are reconstructed(using compression information) and then recovery algorithms are used for the reconstruction of video frames. Our CBS technique is independent of the type of measurement matrix and the recovery technique used, provided that the CS technique is block based. The further compression has

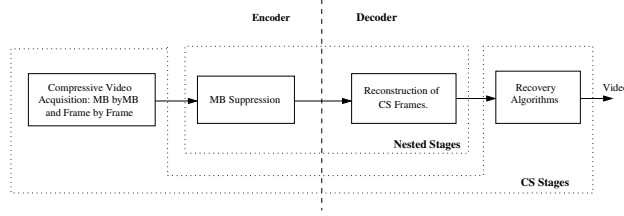


Figure 3.1: Nested video encoding-decoding.

two parts: pre-acquisition block suppression based on block analysis at the start of each GoP; post-acquisition block suppression based on block analysis on a frame by frame basis. Using our block suppression technique, individual MB's don't need to be labeled. Instead, a two dimensional binary array (with one element representing one MB) is sent with each frame.

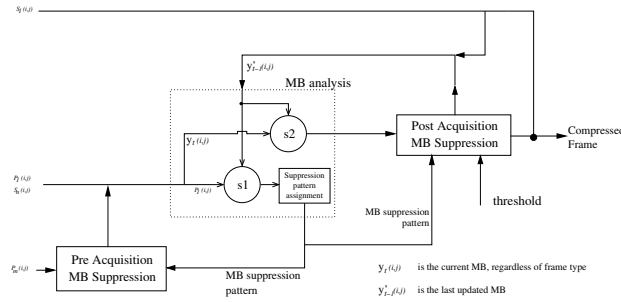


Figure 3.2: The Change based block Suppression Architecture(CBS).

How the information in  $F_t'$  and  $I_t$  (in Fig.3.2) is used to recover the original CS frame has been discussed in section 3.2.6.

### 3.2 Video Acquisition and Storage

We have used the OV7670 CMOS camera for acquisition of VERY low resolution (160\*96), raw (YUV4:2:2) video at 30fps. The OV7670 camera provides an I2C (compatible) interface for configurations. A number of different configurable options are there, for example:

- VGA, QVGA, QQVGA, CIF and QCIF resolutions
- RGB raw, RGB 5:5:5, RGB 5:6:5, YUV 4:2:2,
- Up-to 30fps frame rate.
- Color correction options, e.tc.

Particularly we have done all the processing on the Luma (Y) component i.e. on the gray scale video. This is only to help reduce the data size, and hence processing and transmission times. Also it is normally considered sufficient, as the Luma component contains all the motion information. The OV7670 provides an

8bit parallel data interface and a few other control pins compatible to the DCMI interface on the discovery board. The DCMI interface is a standard camera interface. When DCMI is used, the pixel data comes from the 8bit parallel data interface into the 32bit DCMI data register. Each pixel is of two bytes therefore two pixels are written to the DCMI data register at a time. And before the next two pixels arrive, the DCMI data register has to be read. Otherwise data will be lost. The data rate requirements for initial storage of the raw video frames are:

$$\begin{aligned} \text{datarate} &= \text{framespersec} \times \text{pixelsperframe} \times \text{bytesperpixel} \\ &= 25 \times (160 \times 96) \times 2 \\ &= 1.536\text{MB/sec} \end{aligned}$$

For instance, with the QQVGA resolution, one frame takes 37.5Kbytes of memory, so if the whole frame is stored in the SRAM for processing, and considering that the next frame will also be coming in (as well as the compressed frame), therefore the 192Kbyte SRAM will be stretched. The Kiel uVision5 compiler does not allow for creation of a buffer greater than 60Kbytes in the SRAM. So in order to store a frame in a single buffer we have gone for the resolution, 160\*96. On the other hand, the write speed for the 1MB on-board Flash of the discovery board is 128KB/sec. At this write speed, the video frame cannot be stored in real time for processing. Therefore we have used an external SD card to store video data. The read write speed of SD cards depends on which class they belong. Class2 and class4 SD cards give data read/write speeds of 2MB/sec and 4MB/sec respectively. Class4 to class10 cards give data read/write speeds from 4MB/sec to 15MB/sec. Since our required data rate is more than 4MB/sec and we also have to do other read, write operations, so we have selected the class10 SD card, which is FAT32 filesystem compatible.

### 3.2.1 Data transfer from DCMI to SD card

How the data is transferred from the 32bit DCMI data register to the SD card has a few very important twists to it. Data has to be picked, continuously, from the DCMI data register at a very high frequency so the processor will remain unnecessarily busy. However the DMA (Direct Memory Access) controller on the STM32F4 facilitates the transfer of data from one location to another in embedded system without intervention from the central processor (CPU). Therefore we are using the DMA to transfer data from the DCMI data register. On the other hand there are two standard methods for writing data to the SD card, the uSD and the Fat FS. We have preferred the later because files saved under the Fat FS filesystem are directly interpreted on any windows OS. Since we have to perform sparse recovery of our video stream on some remote computer, the video data will be more easily understood if saved under the fat FS filesystem. However the Fat FS filesystem uses the physical layer interface (SDIO/SPI) with the ST card

to transfer the data to the SD card so the DMA controller cannot be given the address of the SDIO port. Therefore we need a buffer in the middle where data is dumped by the DMA and then Fat FS write command transfers the buffered data to the SD card. This buffer is maintained in the SRAM due to its fast read/write speed.

### 3.2.2 Conversion from Inter-leaved YUV to Planer YUV, and Buffer size

The pixel data of each frame flows into the DCMI, pixel by pixel, in a raster scan fashion. This raster scanned data is Inter-leaved YUV, and since we have to process the Y component separately, we have converted the data to Planer YUV (before writing to the SD card). The Fat FS also uses a DMA channel to read and write, to and from the SD card. We are using the end of frame interrupt, to perform this data format conversion and write to the SD card, before we get the next frame. During this conversion the DCMI is disabled and as a result, the actual frame rate that we are getting is about 20Fps. It is obvious from here that the remaining four processes (Block Columnization, Compressed Sensing, Further Compression and sending data on the SPI to the transmission module) cannot be performed while video acquisition is taking place. Therefore after saving 28 Planer frames in the SD card, we disable the DCMI and start the remaining processes, one by one.

As mentioned in the previous section we are forced to perform all the data manipulations inside the 192KByte SRAM, due to its speed. Each of the above mentioned processes requires buffers to be created in the SRAM, on the other hand the SRAM is not large enough for the creation of all such buffers in the 'static' form. Therefore we have used dynamic memory allocation to create and destroy buffers for each process one-by-one.

### 3.2.3 Block Columnization

We are performing CS on a MB by MB basis (as described in our Q5 report in detail). Therefore in this process we have rearranged the data into columns, where each column is one MB. This is done by reading the whole Y component from the SD card, placing chunks into columns and then writing the columnized data back to the SD card.

### 3.2.4 Compressed Sensing and Further Compression

At this stage we have 28 frames and each frame has 60 columnized MB's. Now we apply CS on each MB one by one and write the CS MB's back to the SD card using `f.write` function of Fat FS. The measurement matrix (i.i.d Gaussian Random matrix) is stored in the SD Card. For each measurement we read one row of the

measurement matrix from the SD Card (using the `f_read` function), perform the floating point matrix multiplication using DSP library, and store the measurement in a CS-MB column-matrix. Implementation using the DSP library for matrix multiplication greatly reduces the computational complexity of the compression algorithm. Also the use of FPU for these calculations reduces the CPU usage by about 6-7 times. In this way each row of the measurement matrix is read one by one and the same procedure is repeated until the desired number of measurements are taken. The CS-MB column-matrix is then saved to the SD Card. The same procedure is repeated for all the MB's. We have successfully tested this process, however since it takes quite a bit of time, we have bypassed this stage so that the rest of the system can be tested and results be compiled.

Now that we have 60 columnized CS-MB's (for each frame), we retrieve CS-MB's (from the SD card) of neighboring frames to calculate SAD values. The SAD calculations are also performed using the DSP library in the FPU format, reducing the computational complexity of the further compression implementation. The SAD values are subsequently used for block suppression in accordance with our CBS technique. The finally compressed frames are stored back into the SD card, ready to be passed down to the transmission module.

### 3.2.5 Pass Data to Transmission Module using SPI

Once the further compression is complete, the compressed data is read from the SD card using Fat FS and stored in a dynamic buffer. The data size for 28 frames is in hundreds of Kilo Bytes, whereas the packet size range is 1-2 Kilo Bytes. On the other hand each MB is of 256 K Bytes ( $16*16$ ), so we have decided to use packet size of 1792 Bytes i.e. 7 MB's in each packet. In order to make sure that data is not lost across the SPI we have placed a 3 byte code at the start of each packet. This code is generated using MOD operations on the size of the packet. On the Transmission module the product of the first three bytes is checked against the expected packet size. If correct, it means the SPI is working fine. Recovery of these frames is performed in the MATLAB environment on any WINDOWS system as described in the next section.

### 3.2.6 Reconstruction and Recovery

Reconstruction is performed at the decoder and has two stages. First, the full CS frames are recovered and then recovery algorithms are used for recovery of video frames. The compressed frame received by the decoder has two parts. In the first part all the unsuppressed, columnized MB's are bundled together in the frame  $\hat{F}_t$  such that each column,  $\hat{F}_t(:, i)$ , is a MB. The second part is a bit-array,  $I_t$  which contains the compression information. So for reconstruction of CS frames we traverse through  $I_t$  and for every zero entry, we place a copy of the corresponding MB from the previous reconstructed CS frame,  $F_{t-1}$ , into the current reconstructed

CS frame,  $F_t$ . Similarly for every non-zero(i.e. one) entry in  $I_t$  we place the next-up MB from  $\hat{F}_t$  into the current reconstructed CS frame,  $F_t$ .

Once the current CS frame is reconstructed, the video frame can be recovered using any suitable recovery algorithm in the bases in which the signal of interest is sparse. The elements of the vectorized measurement vector are quantized individually by an 8-bit uniform scalar quantizer. At the decoder we are using total variation minimization(min-TV) [3, 4], for the recovery.

---

**Algorithm 2** Reconstruction of CS Frames

---

**Input:**  $\hat{F}_t, I_t$

**Output:**

```

for  $i = 0$  to  $size(I_t)$  do
  if ( $I_t(i) = 0$ )
     $F_t(:, i) = F_{t-1}(:, i)$ 
  else
     $F_t(:, i) = \hat{F}_t(:, j)$ 
     $j++$ 
  end if
end for

```

---

### 3.2.7 Results

The video acquired from the physical environment is encoded using our proposed encoding scheme. This gives us encoded and compressed video. The results shown on the right side (right column) depict some of the frames in which compression has been applied. On the left side, the figures show the corresponding frame which is received at the receiver node. The network packet losses result in loss of information since some of the macro-blocks are lost. However, as shown in the following figures the video quality is still reasonable which shows that the communication protocols work in an acceptable manner.



Figure 3.3: mPSMP Operation for a 5 Node Network

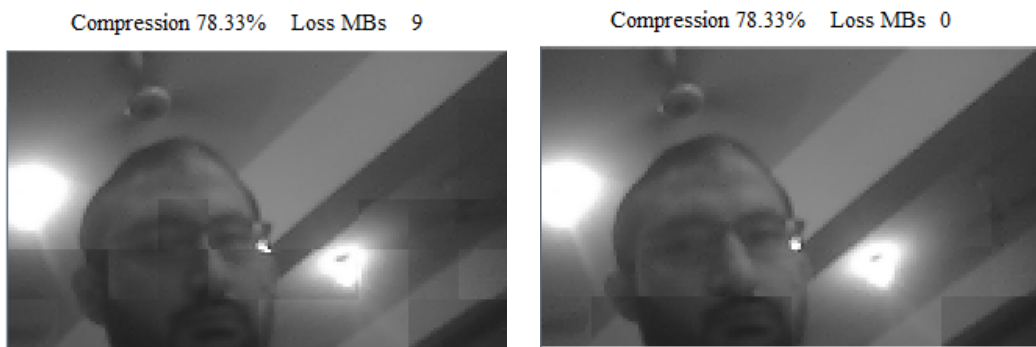


Figure 3.4: mPSMP Operation for a 5 Node Network

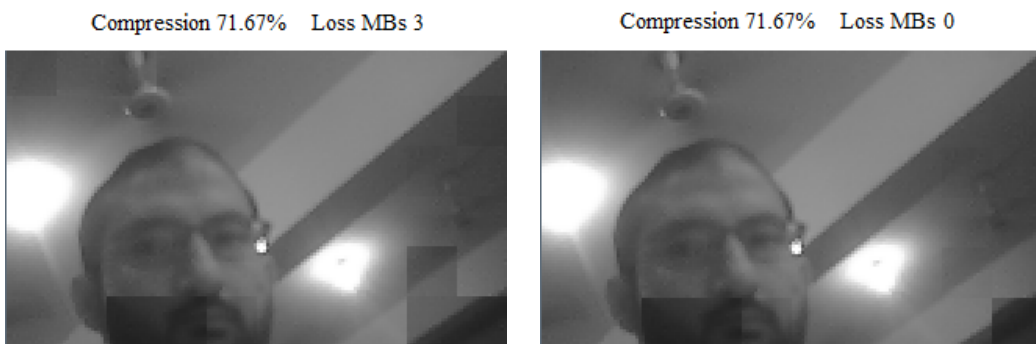


Figure 3.5: mPSMP Operation for a 5 Node Network





Figure 3.6: mPSMP Operation for a 5 Node Network


## Chapter 4

# Conference paper on energy efficiency in IEEE 802.11 for IoM

An international conference paper on energy efficiency in IEEE 802.11 for IoM that was submitted in the previous deliverable is accepted to be published in IEEE CCNC 2016 to be held in January 2016, in Las Vegas, USA.

*CHAPTER 4. CONFERENCE PAPER ON ENERGY EFFICIENCY IN  
IEEE 802.11 FOR IOM*

---



Property	Change Add	Value																												
Conference and track		<b>2016 13th Annual IEEE Consumer Communications and Networking Conference (CCNC) - CCNC 2016 - Sensing, Smart Spaces and IoT</b>																												
Authors	Only the chairs (ccnc2016-chairs@edas.info) can edit	<table border="1" style="width: 100%; border-collapse: collapse; background-color: #f2f2f2;"> <thead> <tr style="background-color: #2c5e8c; color: white;"> <th>Name</th> <th>ID</th> <th>Edit</th> <th>Flag</th> <th>Affiliation (edit for paper)</th> <th>Email</th> <th>Country</th> </tr> </thead> <tbody> <tr> <td>Bilal Afzal</td> <td>717079</td> <td></td> <td></td> <td>University of Engineering and Technology</td> <td>bilal.afzal@kics.edu.pk</td> <td>Pakistan</td> </tr> <tr> <td>Sheeraz A. Alvi</td> <td>690027</td> <td></td> <td></td> <td>University of Engineering and Technology, Lahore &amp; Al-Khawarizmi Institute of Computer Science</td> <td>sheeraz.akhtar@kics.edu.pk</td> <td>Pakistan</td> </tr> <tr> <td>Ghalib A. Shah</td> <td>632263</td> <td></td> <td></td> <td>University of Engineering and Technology</td> <td>ghalib@kics.edu.pk</td> <td>Pakistan</td> </tr> </tbody> </table>	Name	ID	Edit	Flag	Affiliation (edit for paper)	Email	Country	Bilal Afzal	717079			University of Engineering and Technology	bilal.afzal@kics.edu.pk	Pakistan	Sheeraz A. Alvi	690027			University of Engineering and Technology, Lahore & Al-Khawarizmi Institute of Computer Science	sheeraz.akhtar@kics.edu.pk	Pakistan	Ghalib A. Shah	632263			University of Engineering and Technology	ghalib@kics.edu.pk	Pakistan
Name	ID	Edit	Flag	Affiliation (edit for paper)	Email	Country																								
Bilal Afzal	717079			University of Engineering and Technology	bilal.afzal@kics.edu.pk	Pakistan																								
Sheeraz A. Alvi	690027			University of Engineering and Technology, Lahore & Al-Khawarizmi Institute of Computer Science	sheeraz.akhtar@kics.edu.pk	Pakistan																								
Ghalib A. Shah	632263			University of Engineering and Technology	ghalib@kics.edu.pk	Pakistan																								
Title	Only the chairs (ccnc2016-chairs@edas.info) can edit	<i>Adaptive Duty Cycling Based Multi-hop PSMP for Internet of Multimedia Things</i>																												
Abstract	Only the chairs (ccnc2016-chairs@edas.info) can edit	In several use-cases of Internet of Things (IoT), IEEE 802.11 based WLANs are more favorable due to higher data rate support even though their energy efficiency is not up to the mark. Particularly, wireless multimedia sensors based WLANs demand higher energy resources. In this regard, various IEEE 802.11 based power saving mechanisms are developed. IEEE 802.11n standard specifies Power Save Multiple Poll (PSMP) protocol. However, PSMP is infeasible for many IoT based systems specifically in use-cases where multi-hop communication is required. Moreover, PSMP scheduling mechanism lacks the capability to adapt to the dynamic Quality of Service (QoS) requirements in Internet of Multimedia Things (IoMT). In this paper, a QoS aware Multi-Hop PSMP (mPSMP) protocol is proposed to enable energy efficient multimedia communication over IoT. The mPSMP incorporate a traffic scheduling model to allocate channel resources in a time division multiple access manner. Therein, adaptive duty cycling is employed to minimize energy utilization, while assuring the required multimedia QoS for each node. The proposed protocol is implemented in Network Simulator-2 (NS-2). Analytical analysis and simulation study suggests reduction in end-to-end delay and duty cycling along with significant improvement in energy efficiency of IoMT devices.																												
Keywords	Only the chairs (ccnc2016-chairs@edas.info) can edit	Internet of Things; Multimedia sensors; PSMP; Energy efficiency; Multi-hop communication																												
Presenter(s)		presenter not specified																												
Registration code		none																												
DOI	Only the chairs (ccnc2016-chairs@edas.info) can edit																													
Status		Accepted																												
Revisions		...																												
Copyright form																														
Visa letter		none																												
Review manuscript		<p>However, authors cannot upload: paper status</p> <table border="1" style="width: 100%; border-collapse: collapse; background-color: #f2f2f2;"> <thead> <tr style="background-color: #2c5e8c; color: white;"> <th>Document (show)</th> <th>Pages</th> <th>File size</th> <th>Changed</th> </tr> </thead> <tbody> <tr> <td></td> <td>6</td> <td>1,057,802</td> <td>July 28, 2015 05:32:07 America/Los_Angeles</td> </tr> </tbody> </table> <p>notembedded One or more fonts are not embedded. See <a href="#">EDAS FAQ</a>. </p>	Document (show)	Pages	File size	Changed		6	1,057,802	July 28, 2015 05:32:07 America/Los_Angeles																				
Document (show)	Pages	File size	Changed																											
	6	1,057,802	July 28, 2015 05:32:07 America/Los_Angeles																											

# Bibliography

- [1] Ioannis Glaropoulos, Vladimir Vukadinovic, and Stefan Mangold. Contiki80211: An iee 802.11 radio link layer for the contiki os. In *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on*, pages 621–624. IEEE, 2014.
- [2] Vladimir Vukadinovic, Ioannis Glaropoulos, and Stefan Mangold. Enhanced power saving mode for low-latency communication in multi-hop 802.11 networks. *Ad Hoc Networks*, 23:18–33, 2014.
- [3] Z. Liu, H. Vicky Zhao, and A. Y. Elezzabi. "BLOCK-BASED ADAPTIVE COMPRESSED SENSING FOR VIDEO". In *Proceedings of 2010 IEEE 17th International Conference on Image Processing*, September 2010.
- [4] Z. Liu, A. Y. Elezzabi, and V. Zhao. "Maximum Frame Rate Video Acquisition Using Adaptive Compressed Sensing". *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 21(11), November 2011.