International Journal of Digital Information and Wireless Communications (IJDIWC) 8(3): 168-173
The Society of Digital Information and Wireless Communications, 2018 ISSN: 2225-658X (Online); ISSN 2412-6551 (Print)

Simulating Image Communication over Multi-Hop Wireless Sensor Networks Using TOSSIM

Cristian Duran-Faundez*, Pablo Sáez-Srain*, Daniel G. Costa‡ and Gilbert Habib§
*Department of Electrical and Electronic Engineering,
University of the Bío-Bío, Concepción, Chile
Email: {crduran,psaez}@ubiobio.cl
†Department of Technology,
State University of Feira de Santana, Feira de Santana, Brazil,
Email: danielgcosta@uefs.br
‡Department of Computer Sciences,
Lebanese University, Fanar, Lebanon
Email: gilbert.habib@ul.edu.lb

***Abstract*—Wireless camera sensor networks have been employed for different types of visual monitoring applications, promoting innovative services for Internet of Things scenarios and smart cities. In this context, in order to better support the validation procedures of visual sensor networks, we implement various simulation cases aimed at resource consumption measurement of intermediate nodes in multi-hop networks. Moreover, we also implement a routing protocol specially conceived for image communication, and verifications are performed for non-compressed and compressed TiBS and JPEG* images. Our early results illustrate the benefits of performing image compression for resource savings, as well as the effectiveness of robust error resilient techniques to face packet losses.**

***Keywords*—Wireless camera sensor networks; Image communication; Simulation; TOSSIM; Resource consumption.**

I. Introduction

Wireless Sensor Networks (WSNs) with vision capabilities [1], [2] are emerging technologies with potential uses in many application fields. In such networks, sensor nodes are equipped with a low-power camera, allowing monitoring of the application environment and providing valuable information that have to be transmitted through the composed wireless network. Image communication over WSNs faces typical resource limitations of this kind of network (limited energy, restrictions in computational power, small storage, etc.) along with a much greater volume of information to transmit. These characteristics have motivated much research efforts, focusing on issues such as image compression [3]–[6], network protocols [7] and monitoring optimizations [8]–[10].

Whatever the subject, simulation tools must be used to study nodes behavior before implementation, or when no real-world platforms are available at the required scale. In the case of camera sensor networks, some simulation tools allow important network simulation issues [11] and image quality assessment [12], generally using simplified abstract models. Nevertheless, there is a lack of simulation tools considering real-world operating systems and platforms. Also, there is a lack of tools that would employ the same programs of the ones required by real nodes, which could prejudice the accuracy of simulated networks.

In this work, we address the issue of simulating image communication in multi-hop wireless sensor networks environments, considering the real-world-oriented simulation platform TOSSIM [13]. TOSSIM is the simulation tool of TinyOS [14], an operating system designed for resource constrained wireless devices, and it is used in WSNs nodes such as those of the Mica family (from Crossbow) and Telos. TOSSIM compiles nesC codes (the programming language of TinyOS) and executes simulations on personal computers.

In the camera sensor networks field, TinyOS has been used as the operating system of the Cyclops camera [15], a low-cost image sensor capable of being connected to a Crossbow mote such as Mica2. In this paper, we describe our first results on simulating image communication over multi-hop WSNs with TOSSIM. Particularly, we consider scenarios where all nodes are Mica2 motes and source nodes are equipped with Cyclops camera sensors. As a sample scenario, we consider definitions in [3], where resource consumption due to compression and transmission tasks are resulted from performance measurements of an implementation over a node consisting on a Cyclops camera attached to a Mica2 mote. This hardware integration is shown in Figure 1.

In [3], TiBS, an image compression algorithm specially conceived for image communication over WSNs, is introduced. TiBS is compared with a JPEG-like algorithm (noted JPEG*) showing good results in terms of resources consumption and error resilience. This last topic is important because errors are inherent to WSNs [16]. Results are usually given for a source node, but no studies about intermediate nodes are provided. In this paper, we implement a multi-hop scenario for sending images as reported in [3]. Unfortunately, Cyclops hardware has not been incorporated to TOSSIM, so we focus on the intermediate nodes behavior. From [3], we take basic information for performance assessment, such as the transmission speed and the packet types. As a routing protocol, we adopt the proposal reported in [17], which is specially conceived for

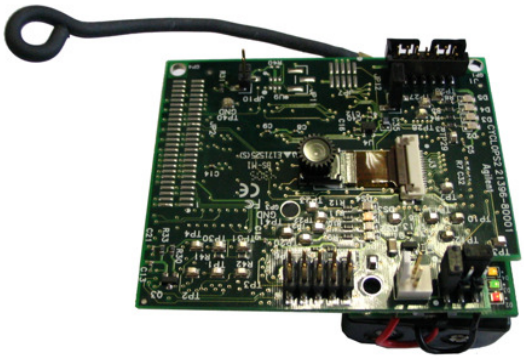


Fig. 1. A camera node consisting on a Cyclops camera attached to a Mica2 mote.

image communication in WSNs.

The rest of this paper is organized as follows: Section II presents the simulated scenarios, describing the routing algorithm and other parameters for the simulations. Section III describes the use of TOSSIM for simulating image communication through multi-hop networks. Section IV summarizes and discusses simulation results. Finally, Section V presents the conclusion of this work.

II. Multi-hop simulation scenario

We consider a multi-hop scenario composed of n nodes $S = \{s_0, s_1, s_2, \ldots, s_{n-1}\}$. From these nodes, there is a subset $C \subset S$ of nodes that have cameras. These nodes capture and send images towards a node $s_s \in S$. The considered topology is a grid of G_H rows by G_W columns nodes, where node s_i is located at coordinate $\left(S_F.\lfloor\frac{i}{G_W}\rfloor, S_F.(i \mod G_W)\right)$, S_F being the spacing factor (the distance between two adjacent nodes in the same row or column, in the grid).

In this work, we use TinyOS 1.5.x, which is compatible with Cyclops camera firmware (last version is 2.1.2, but Cyclops firmware was not updated to this version). The studied simulation scenario consists on a network of $n = 100$ nodes, located in a grid of 10×10 nodes, with only one camera source s_c. To have a considerable quantity of intermediate hops between the source and the sink, we set $s_s = s_0$ and $s_c = s_{88}$. Moreover, TOSSIM sets as default transmission radius for the nodes as 50 ft [13], thus we chose a spacing factor $S_F = 20$ (ft) with what we have at least 6 hops in the shortest path, and a neighborhood of up to 20 neighbors per node. Finally, the power output is fixed in 0db, and we set a zero BER (Bit Error Rate), which does not means that no packets are lost.

A. Image codification and transmission

Without loss of generality, a source node will capture an image of $H \times W$ pixels, then codify it in b bits per pixel (bpp) (this could include image compression), and then transmit it towards the sink into $N_P = \lceil\frac{H.W.b}{8.Payload}\rceil$ packets, where $Payload$ is the payload available for image data in bytes. In this paper, we consider the same sample cases of [3], whose parameters are summarized in Table I.

TABLE I
Image parameters for simulation (extracted from [3]).

Case	Bitrate b (bpp)
No compression	8
TiBS ($\Delta = 0$)	5.50
TiBS ($\Delta = 1$)	4.62
TiBS ($\Delta = 2$)	3.77
TiBS ($\Delta = 3$)	2.95
JPEG* ($Q_{tab} = 90$)	2.73
JPEG* ($Q_{tab} = 50$)	0.90
JPEG* ($Q_{tab} = 10$)	0.37

Captured images are 8bpp monochromatic of 128×128 pixels. Then, these images are compressed with TiBS and JPEG* with different key parameters. Images are transmitted in packets with $Paylod = 27$ bytes (default parameter for this version of TinyOS). Thus, the amount of required packets per image vary from $N_P = 607$ packets, in the case of the transmission of a non-compressed image, to $N_P = 29$, in the case of the transmission of an image compressed with JPEG* with $Q_{tab} = 10$.

B. Routing algorithm

As stated before, we adopt the routing algorithm from [17]. Each node j forwards data packets to node i, in its routing table, based on a cost metric to minimize, defined as:

$$c(i) = c_p(i) + \alpha.c_q(i) + \beta.c_e(i) \quad (1)$$

where:

- $c_p(i)$, $c_p(i) \in [0, 1]$, is a position cost. Distance and angle-based routing schemes can be selected. We adopt distance-based routing so $c_p(i)$ is defined as the euclidean distance between neighboring node i and the sink, normalized to the biggest position cost of the neighbors of node j,
- $c_q(i)$ is a queue cost, and
- $c_e(i)$ is an energy cost, defined in terms of the remaining energy of node i. The function describing $c_e(i)$ is a non-increasing function, where zero (or very low) remaining energy is associated to $c_e(i) = 1$ and maximum available energy is associated to $c_e(i) = 0$.

Actually, α and β are selectable weights, and they can be used to prioritize a given factor. In this paper, we adopt distance-based with energy cost decisions; as it is presented in [17], these parameters are $\alpha = 0$ and $\beta = 100$. As a function for $c_e(i)$, we implement the following equation:

$$c_e(i) = \begin{cases} 0 & \text{if } r_e \geq E_t \\ \frac{5}{100} + 1 & \text{otherwise} \end{cases} \quad (2)$$

where r_e is a value related to the remaining energy of node i, E_t is the maximum value for r_e, E_{max}, multiplied by some selectable value $\gamma \in]0, 1]$ (in this case, we set $\gamma = 0.75$), N_s

is the number of the desired discretized regions of function $c_e(i)$, and ς is calculated as:

$$\varsigma = \left(\left\lfloor \frac{-100.\frac{E_t}{N_s}.\left\lfloor \frac{r_e}{\left(\frac{E_t}{N_s}\right)} \right\rfloor}{E_t} \right\rfloor \right) \quad (3)$$

III. Simulation Implementation

The previously described routing algorithm was implemented in TinyOS and the details of such implementation are described in this session. Firstly, we describe the selected packets format and, secondly, the main module to be executed by the simulated motes is presented.

1) Packets format: The following two types of data packets were implemented:

- `PeriodicMsg`: which is periodically sent in broadcast by each node, in order to keep its neighbors informed about its status. The format of `PeriodicMsg` is implemented with the following code:

Listing 1
Definition of the PeriodicMsg packet format.

```
typedef struct PeriodicMsg {
 uint16_t source_addr; // contains the address of
      ↪ the source node.
 double dsink; // the Euclidean distance of the
      ↪ sender node (the neighbor) to the Sink.
 double c_e; // Energy Cost (as in the routing
      ↪ cost function)
} PeriodicMsg;
```

- `EventMsg`: which is used to emulate the image packets transmitted once an event occurs. The format of `EventMsg` is implemented with the following code:

Listing 2
Definition of the EventMsg packet format.

```
typedef struct EventMsg {
  uint16_t serialDumpHeader_s;
  uint16_t data;
} EventMsg;
```

Both packet formats are stored in a joint header file.

2) Main module implementation: Classical TinyOS codification uses 2 files: one for components linking and general definitions, and other for particular modules implementation. In our implementation, those files are `psRoutingSavidge.nc` and `psRoutingSavidgeM.nc`, respectively. The content of the first file is shown in Listing 3.

Listing 3
Content of file psRoutingSavidge.nc.

```
includes MessageType;

configuration psRoutingSavidge { }
implementation
{
  components Main, psRoutingSavidgeM, TimerC, LedsC,
       ↪ GenericComm as Comm, CC1000RadioC;

   Main.StdControl -> psRoutingSavidgeM;
   Main.StdControl -> TimerC;

   psRoutingSavidgeM.TimerSendPacket -> TimerC.Timer[
        ↪ unique("Timer")];
   psRoutingSavidgeM.TimerPeriodicPacket -> TimerC.Timer
        ↪ [unique("Timer")];
   psRoutingSavidgeM.TimerCaptTrans -> TimerC.Timer[
        ↪ unique("Timer")];
   psRoutingSavidgeM.Leds -> LedsC;
   psRoutingSavidgeM.SendPeriodicMsg -> Comm.SendMsg[
        ↪ PSTYPE_PERIODICMSG];
   psRoutingSavidgeM.ReceivePeriodicMsg -> Comm.
        ↪ ReceiveMsg[PSTYPE_PERIODICMSG];
   psRoutingSavidgeM.SendEventMsg -> Comm.SendMsg[
        ↪ PSTYPE_EVENTMSG];
   psRoutingSavidgeM.ReceiveEventMsg -> Comm.ReceiveMsg[
        ↪ PSTYPE_EVENTMSG];
   psRoutingSavidgeM.SubControl -> Comm;
   psRoutingSavidgeM.CC1000Control->CC1000RadioC;
}
```

The second file contains most of the implementation of the algorithm. The global structure of the code, along with comprehensible comments describing the implemented algorithm is shown in Listing 4:

Listing 4
Global structure of code in psRoutingSavidgeM.nc.

```
includes MessageType;

module psRoutingSavidgeM
{
 provides interface StdControl;
 uses {
  interface Timer as TimerSendPacket;
  interface Timer as TimerPeriodicPacket;
  interface Timer as TimerCaptTrans;
  interface Leds;
  interface StdControl as SubControl;
  interface SendMsg as SendPeriodicMsg;
  interface ReceiveMsg as ReceivePeriodicMsg;
  interface SendMsg as SendEventMsg;
  interface ReceiveMsg as ReceiveEventMsg;
  interface CC1000Control;
 }
}

implementation
{
 //INITIAL DECLARATIONS
   // DECLARATION OF SIMULATION PARAMETERS' CONSTANTS
   ...
   // DEFINITION OF SOURCE AND SINK ADDRESSES
   ...
   // DECLARATION OF A STRUCTURE FOR THE ROUTING TABLE
   ...
   // DECLARATION OF GLOBAL VARIABLES
   ...


 // FUNCTIONS IMPLEMENTATION:

double c_e (uint16_t r_e)
{
 // RETURNS c_e FOR CURRENT NODE GIVEN INPUT r_e
 ...
}

double dist(uint16_t node1_addr, uint16_t node2_addr)
{
 // RETURNS DE EUCLIDEAN DISTANCE BETWEEN NODES WITH
      ↪ ADDRESSES node1_addr AND node2_addr
 ...
}

uint16_t bestCiRotingTable_addr(){
 // RETURNS THE ADDRESS OF THE NODE IN THE ROUTING TABLE
      ↪  WHICH HAS THE LOWEST C_i VALUE
 ...
}

command result_t StdControl.init() {
```

```
// CALLS TO INIT ROUTINES OF HARDWARE COMPONENTS. FIRST PROCEDURE EXECUTED.
 ...
}

command result_t StdControl.start() {
 // AFTER init(). INITIAL STEPS OF THE PROGRAM.

 //VARIABLE DECLARATIONS
 ...
 //SET POWER OUTPUT
 call CC1000Control.SetRFPower(0x80); //0x02= -20db (3,7 mA); 0x80= 0db (8,5mA); 0xFF= +5db (12mA)

 //INIT ROUTING TABLE
 ...

 if(TOS_LOCAL_ADDRESS == SOURCE_ADDR){
  // THIS IS THE CODE EXECUTED BY THE SOURCE NODE.
  call TimerCaptTrans.start(TIMER_REPEAT, 60000); // IT INITIALIZES A TIMER TO SEND AN IMAGE EVERY MINUTE
 }

 call TimerPeriodicPacket.start(TIMER_REPEAT, 300000); // THIS TIMES WILL SEND PERIODIC PACKETS IN BROADCAST EVERY 5 MINUTES

 call SubControl.start();
 return SUCCESS;
}

command result_t StdControl.stop() {
 // CALLS FOR STOP TIMERS AND SUBCONTROL COMPONENTS.
 ...
}

event result_t TimerSendPacket.fired() {
 // THIS TIMER SENDS (IMAGE) DATA PACKETS (EventMsg) FROM THE SOURCE NODE TOWARDS THE SINK.
 // THE PACKET WILL CARRY IN THE data FIELD, A SEQUENTIAL NUMBER STARTING IN 0 AND ENDING IN N_PACKETS-1, WHICH IS THE NUMBER OF PACKETS REQUIRED TO TRANSMIT A DETERMINED IMAGE.
 // IF N_PACKETS TRANSMITTED, THEN IT STOPS TIMER TimerSendPacket.
 ...
}

event result_t TimerCaptTrans.fired() {
 // THIS TIMER STARTS IMAGE TRANSMISSION (SO IT SETS TimerSendPacket).
 data_to_send=0; // INITIALIZE SEQUENTIAL DATA PACKET NUMBER
 call TimerSendPacket.start(TIMER_REPEAT, 53.6903); // TimerSendPacket IS REQUIRED TO TRANSMIT A DATA PACKET EACH 54 MILI SECONDS AROUND, WHICH IS CYCLOPS+MICA2 MOTE TRANSMISSION DATA RATE.
 return SUCCESS;
}

event result_t SendPeriodicMsg.sendDone(TOS_MsgPtr msg, result_t success)
{
 // ONCE A PACKET IS SENT, A REMAINING ENERGY VARIABLE IS DECREASSED
 ...
}

event TOS_MsgPtr ReceivePeriodicMsg.receive(TOS_MsgPtr data_msg) {
 // THIS EVENT IS EXECUTED WHEN A NODE RECEIVE A PeriodicMsg.
 // THE ROUTING TABLE IS UPDATED WITH THE NEW INFORMATION FROM THE SENDER NEIGHBOR.
 ...
}

event result_t SendEventMsg.sendDone(TOS_MsgPtr msg, result_t success)
{
 // ONCE A PACKET IS SENT, A REMAINING ENERGY VARIABLE IS DECREASSED (THIS TIME IS FOR EventMsg PACKETS).
 ...
}

event TOS_MsgPtr ReceiveEventMsg.receive(TOS_MsgPtr data_msg) {
 // THIS EVENT IS EXECUTED WHEN AN EventMsg IS RECEIVED.
 // THE CODE HAS TWO POSSIBLE BEHAVIORS:
 // IF THE CURRENT NODE IS THE SINK, THEN IT SIMPLY REPORTS THE RECEPTION BY USING DEBUG MESSAGES.
 // IF THE CURRENT NODE IS NOT THE SINK, THEN FORWARD THE PACKET (IF NOT FORWARDED BEFORE) TOWARDS THE SINK, SENDING THE PACKET TO THE NEXT NODE OBTAINED WITH FUNCTION bestCiRotingTable_addr.
 ...
}

event result_t TimerPeriodicPacket.fired() {
 // WHEN THIS TIMER IS FIRED, A NEW PeriodicMsg PACKET IS PREPARED WITH CURRENT STATUS AND SENT IN BROADCAST TO THE NEIGHBORING NODES.
 ...
}
```

TABLE II
SUMMARIZED RESULTS.

Case	Total energy consumption (mJ)			PLR
	Average	Minimum	Maximum	
No compression	20118.37	20088.39	20823.77	0.51
TiBS ($\Delta = 0$)	20109.59	20088.52	20599.70	0.50
TiBS ($\Delta = 1$)	20106.46	20088.33	20527.27	0.50
TiBS ($\Delta = 2$)	20102.97	20088.20	20448.71	0.51
TiBS ($\Delta = 3$)	20023.54	20011.75	20298.37	0.50
JPEG* ($Q_{tab} = 90$)	19997.53	19986.70	20253.85	0.52
JPEG* ($Q_{tab} = 50$)	19749.29	19745.65	19834.27	0.50
JPEG* ($Q_{tab} = 10$)	19678.14	19676.51	19716.00	0.47

Full implemented files are published at the following link: http://crduran.ubb.cl/index.php?option=com_content&view=article&id=15.

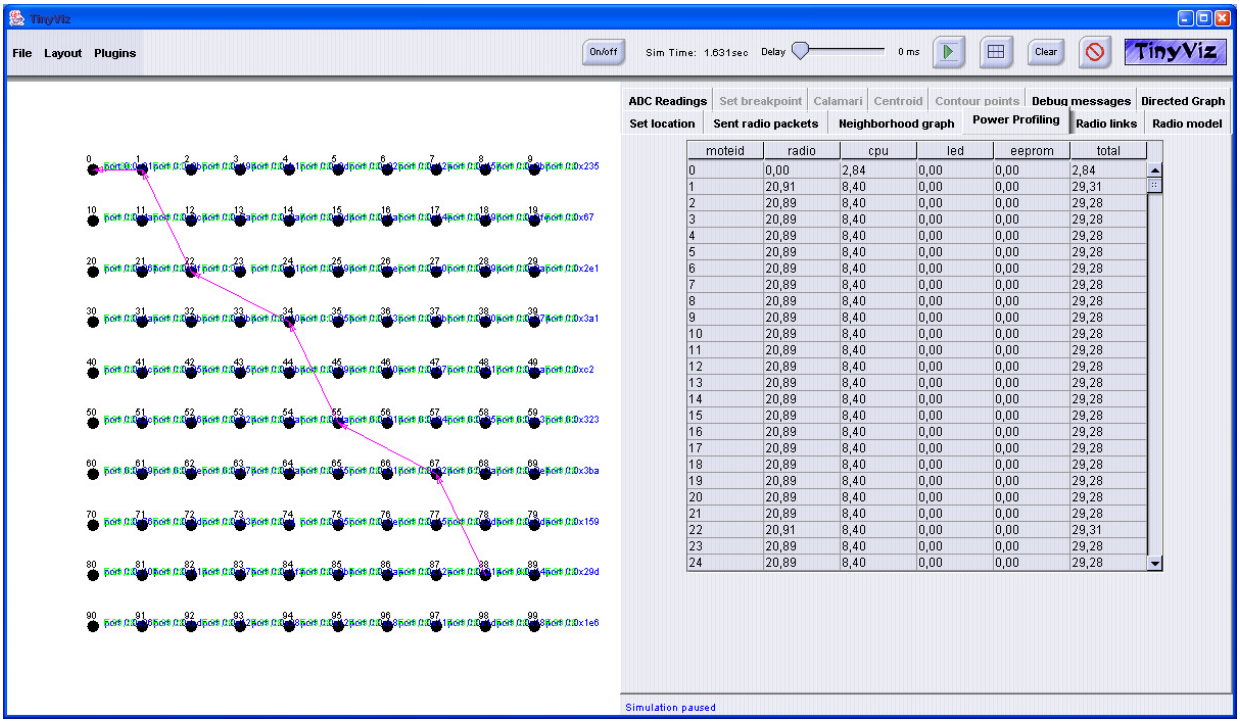
Figure 2 shows the execution of the implemented algorithm over the grid topology, over the visualisation tool TinyViz [13].

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Simulations for the various cases of Table I were performed, for $N_s = 5$. For each simulation, 10 minutes of simulated time were executed (some real hours of simulation if we consider the level of detail with which TOSSIM simulates hardware). Simulation is set to start all nodes at the same simulation time (time zero). Results in terms of energy consumption are collected with PowerTOSSIM plugin [18], which includes energy models for Mica2 components. Summarized results are given in Table II.

Of course, average energy consumptions, along with the minimum and maximum energy consumed by a node, do not comprise the source and the sink nodes, which are supposed to have different energy consumptions. From these early results, it is possible to appreciate the decrease of the average energy consumed by the network when image compression is performed. Obviously, more significant differences should be visualized in longer periods of time.

Energy differences are not just because of the decrease on the amount of data packets achieved by image compression.



moteid	radio	cpu	led	eeprom	total
0	0,00	2,84	0,00	0,00	2,84
1	20,91	8,40	0,00	0,00	29,31
2	20,89	8,40	0,00	0,00	29,28
3	20,89	8,40	0,00	0,00	29,28
4	20,89	8,40	0,00	0,00	29,28
5	20,89	8,40	0,00	0,00	29,28
6	20,89	8,40	0,00	0,00	29,28
7	20,89	8,40	0,00	0,00	29,28
8	20,89	8,40	0,00	0,00	29,28
9	20,89	8,40	0,00	0,00	29,28
10	20,89	8,40	0,00	0,00	29,28
11	20,89	8,40	0,00	0,00	29,28
12	20,89	8,40	0,00	0,00	29,28
13	20,89	8,40	0,00	0,00	29,28
14	20,89	8,40	0,00	0,00	29,28
15	20,89	8,40	0,00	0,00	29,28
16	20,89	8,40	0,00	0,00	29,28
17	20,89	8,40	0,00	0,00	29,28
18	20,89	8,40	0,00	0,00	29,28
19	20,89	8,40	0,00	0,00	29,28
20	20,89	8,40	0,00	0,00	29,28
21	20,89	8,40	0,00	0,00	29,28
22	20,91	8,40	0,00	0,00	29,31
23	20,89	8,40	0,00	0,00	29,28
24	20,89	8,40	0,00	0,00	29,28

Fig. 2. Visualization of a simulation using TinyViz.

Packet Loss Rate (PLR) has also a significant role. For the different simulated cases, PLR values are also summarized in Table II.

As shown, PLR is around 50% in all cases. This can be due to various factors. The principal factor could be the rapid transmission of image packets from the source node. As reported in [3], considering a non compressed image, the entire cycle (including capture, processing and transmission) takes about 33.56 seconds at the source node. Capture takes about 9 seconds, so the 607 packets are transmitted in about 25 seconds. One problem can be the speed required for the intermediate nodes to change between reception and transmission modes. Other reason can be the queue management. These results put in evidence that methods ensuring robustness to packet loss must be incorporated. In [3], block interleaving is applied to increase the probability of receiving correct information to reconstruct a missing block. At these levels of packet loss, TiBS still ensures good quality for the received images, while JPEG-based images start to have big degradation [3].

V. Conclusion

In this paper, we described initial results on simulating large scale multi-hop WSNs with vision capabilities. We adopt TOSSIM, a simulation tool for TinyOS programs, which is employed for simulation assuming real-world platforms as experimental references. In this context, we implemented the routing scheme presented in [17], which includes a weighted cost function considering position, queuing and energy costs of neighboring nodes. The transmission of various cases with compressed and non-compressed images are performed, showing the potential benefits of image compression in order to decrease resource consumption of nodes. Moreover, the impact of robust compression techniques to face high packet loss rates is also addressed.

Acknowledgment

This work was supported by the University of the Bío-Bío under Grants DIUBB 161610 2/R and GI 160210/EF, and the Lebanese University under Grant 2015/438.

References

[1] S. Soro and W. R. Heinzelman, “A survey of visual sensor networks,” *Adv. in MM*, vol. 2009, 2009.

[2] D. G. Costa and C. Duran-Faundez, “Assessing availability in wireless visual sensor networks based on targets’ perimeters coverage,” *Journal of Electrical and Computer Engineering*, vol. 2016, pp. Article ID 9 312 439, 1–14, 2016.

[3] C. Duran-Faundez, V. Lecuire, and F. Lepage, “Tiny block-size coding for energy-efficient image compression and communication in wireless camera sensor networks,” *Image Commun.*, vol. 26, no. 8-9, pp. 466–481, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.image.2011.07.005

[4] M. L. Kaddachi, A. Soudani, V. Lecuire, L. Makkaoui, J.-M. Moureaux, and K. Torki, “Design and performance analysis of a zonal dct-based image encoder for wireless camera sensor networks,” *Microelectronics Journal*, vol. 43, no. 11, pp. 809 – 817, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026269212001449

[5] A. Chefi, A. Soudani, and G. Sicard, “Hardware compression scheme based on low complexity arithmetic encoding for low power image transmission over {WSNs},” {*AEU*} - *International Journal of Electronics and Communications*, vol. 68, no. 3, pp. 193 – 200, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1434841113002112

[6] T. Pal, S. Bandyopadhyay, and S. Dasbit, “Energy-saving image transmission over wmsn using block size reduction technique,” in *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, Dec 2015, pp. 41–46.

[7] H. Shen and G. Bai, "Routing in wireless multimedia sensor networks: A survey and challenges ahead," *Journal of Network and Computer Applications*, vol. 71, pp. 30 – 49, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516301102

[8] D. G. Costa, L. A. Guedes, F. Vasques, and P. Portugal, "Research trends in wireless visual sensor networks when exploiting prioritization," *Sensors*, vol. 15, no. 1, pp. 1760–1784, 2015.

[9] C. Duran-Faundez, D. G. Costa, V. Lecuire, and F. Vasques, "A geometrical approach to compute source prioritization based on target viewing in wireless visual sensor networks," in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, 2016, pp. 1–7.

[10] D. G. Costa, M. Collotta, G. Pau, and C. Duran-Faundez, "A fuzzy-based approach for sensing, coding and transmission configuration of visual sensors in smart city applications," *Sensors*, vol. 17, no. 1, 2017.

[11] C. Nastasi and A. Cavallaro, "Wise-mnet: an experimental environment for wireless multimedia sensor networks," in *Proceedings of Sensor Signal Processing for Defence (SSPD)*, London, UK, September 2011.

[12] E. Orellana-Romero, J. SanMartin-Hernandez, C. Duran-Faundez, V. Lecuire, and C. Aguilera, "Sim-lit: A simulation framework for image quality assessment in wireless visual sensor networks under packet loss conditions," in *Computer Science Society (SCCC), 2011 30th International Conference of the Chilean*, Nov 2011, pp. 202–209.

[13] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 126–137. [Online]. Available: http://doi.acm.org/10.1145/958491.958506

[14] "Tinyos." [Online]. Available: www.tinyos.net

[15] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, San Diego, CA, November 2005, pp. 192–204.

[16] C. Yeo and K. Ramchandran, "Robust distributed multi-view video compression for wireless camera networks," in *Visual Communications and Image Processing*, vol. 6508, San Jose, CA, USA, January 2007.

[17] L. Savidge, H. Lee, H. Aghajan, and A. Goldsmith, "Event-driven geographic routing for wireless image sensor networks," in *in Proceedings of the Proceedings of Cognitive Systems and Interactive Sensors (COGIS 06*, 2006.

[18] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 188–200. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031518

