Delay Accounting Optimization Procedure to Enhance End-to-End Delay Estimation in WSNs

Pedro Pinto[1(✉)], António Pinto[2], and Manuel Ricardo[3]

[1] ESTG, Instituto Politécnico de Viana do Castelo and INESC TEC, Viana do Castelo and Porto, Portugal
pedropinto@estg.ipvc.pt
[2] CIICESI, ESTGF, Politécnico do Porto and INESC TEC, Porto, Portugal
apinto@inescporto.pt
[3] INESC TEC, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
mricardo@inescporto.pt

Abstract. Real-time monitoring applications may generate delay sensitive traffic that is expected to be delivered within a firm delay boundary in order to be useful. In this context, a previous work proposed an End-to-End Delay (EED) estimation mechanism for Wireless Sensor Networks (WSNs) to preview potential useless packets, and to early discard them in order to save processing and energy resources. Such estimation mechanism accounts delays using timers that make use of an Exponentially Weighted Moving Average (EWMA) function where the smoothing factor is a constant defined prior to the WSN deployment. Later experiments showed that, in order to enhance the estimation results, such smoothing factor should be defined as a function of the network load.

The current work proposes an optimization of the previous estimation mechanism that works by evaluating the network load and by adapting the smoothing factor of the EWMA function accordingly. Results show that this optimization leads to a more accurate EED estimation for different network loads.

Keywords: End-to-End Delay · Delay Estimation · EWMA · Smoothing factor

1 Introduction

Real-time applications may generate packet flows requiring specific service levels from the network. Applications that use delay sensitive flows can assume that such flows are only useful if received within a strict delay limit and useless otherwise. The deployment of these applications on top of Wireless Sensor Networks (WSNs) with scarce energy and processing resources, requires additional efforts in order to preview and avoid the transmission of potential useless data packets. Our previous work [1] presents an End-to-End Delay Estimation Mechanism (EEDEM) for delay sensitive applications deployed on a WSN that tries to accurately classify the usefulness of data packets in real-time. EEDEM estimates the

S. Mumtaz et al. (Eds.): WICON 2014, LNICST 146, pp. 14–19, 2015.
DOI: 10.1007/978-3-319-18802-7_3

End-to-End Delay (EED) based on the internal delays experienced by previously sent packets, and delay information from other nodes through the use of Routing Protocol for Low-power and Lossy Networks (RPL). All internal delays are accounted using an Exponentially Weighted Moving Average (EWMA) function, which defines the weight of the last value in relation to the history value, using a constant smoothing factor (β) defined *a priori*. In order to enhance the EED estimation, the β factor should be defined as a function of the network load.

This paper presents a Delay Accounting Optimization Procedure (DAOP) which dynamically applies, at each node, the best β as function of the network load. Thus, DAOP enables the lowest estimation error for multiple network loads.

The structure of this paper is as follows. Section 2 presents the related work. Section 3 details the preliminary experiments conducted. Section 4 describes the current proposal. Section 5 shows the results obtained. Section 6 concludes paper.

2 Related Work

EEDEM [1] assumes that a set of generator/forwarder nodes generate and forward data to a sink node, and that the sink node is the ultimate destination of all data. Each node accounts for the time elapsed while the packet is processed within the stack of the generator node, the time elapsed while in the MAC layer queuing, and the time elapsed in packet transmissions. Delay accounting is accomplished by using timers that register delays between labels inserted into parts of the code where the data passes through, ranging from the application in the generator node to the application in the sink node (see Fig. 1).

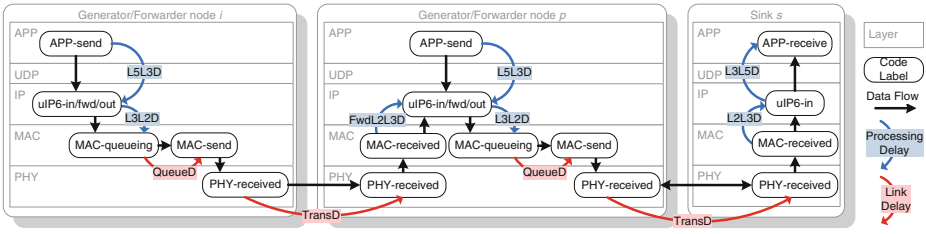

Fig. 1. EEDEM overview

The internal delays used are: 1) Generation Internal Delay (GenIntDelay) registered when packets are generated; 2) Forward Internal Delay (FwdIntDelay) registered when packets are being forwarded; 3) Receiving Internal Delay (RecIntDelay) registered when packets reach the destination. The *GenIntDelay* obtained at a node i with a parent p is calculated as follows:

$$GenIntDelay_{ip} = L5L3D_i + L3L2D_i + QueueD_i + TransD_{ip} \quad (1)$$

The *FwdIntDelay* obtained at a node p with a parent s is calculated as follows:

$$FwdIntDelay_{ps} = FwdL2L3D_p + L3L2D_p + QueueD_p + TransD_{ps} \quad (2)$$

The *RecIntDelay* obtained at the sink s node is calculated as follows:

$$RecIntDelay_s = L2L3D_s + L3L5D_s \quad (3)$$

where *LxLyD* is the delay between layer x and layer y, *QueueD* is the MAC queuing delay, and *TransD* is the transmission delay.

Each node calculates the delay of all the path up to the sink node by using a feedback mechanism that announces back the cumulative delays to other nodes using RPL [2] with delay-based metrics (RPLMetrics). Each node provides a real-time EED estimation up to the sink, per generated packet, by combining the internal delays with the *RPLMetrics*. In [3] a set of RPL modifications was proposed to improve EEDEM work. Regarding the internal delay, each component (*GenIntDelay*, *FwdIntDelay* and *RecIntDelay*) is obtained by using EWMA. The last delay and the all delay history values, for a packet n, are calculated using a β as follows:

$$Delay^n = \beta.Delay^{last} + (1 - \beta).Delay^{n-1} \quad (4)$$

Our previous EEDEM work estimates delays using a constant value for the β, defined *a priori*. Better estimation results are achieved when using different β values, adapted depending on the network load. Other research efforts use or adapt EWMA for estimation purposes. In [4] authors present an adaptive forecast method based on EWMA. In [5] authors propose the use of routing metrics that are obtained using EWMA.

3 Preliminary Experiments

Preliminary experiments were performed in order to better understand how the EED estimation error (EED_Error) changes in relation to different β values. The Cooja simulator [6] was used to setup a WSN of 16 nodes plus a sink node, all nodes were simulated as Tmote Sky [7] and configured with a transmission range of 30 m using the Unit Disk Graph Medium as physical channel model. The nodes ran the Contiki OS 2.5 and were deployed in a grid topology within an area of 100 m^2. The application layer used UDP and it generated packets of 100 Bytes in a constant rate here defined as Inter-packet Generation Intervals (IGI). Simulations were configured to stop whenever the sink node received 100 packets from each node, and were repeated 10 times using random seeds. The simulator was configured to output the instant of time when a packet was generated and when a packet reached the destination application. For each generated packet, the estimated EED (estEED) was collected and later compared with the real EED (realEED). Finally, when the simulation ended, the EED_Error for N samples was obtained using the difference between estEED and realEED calculated using the Symmetric Mean Absolute Percentage Error (SMAPE) according to Eq. 5, expressed in a value between 0 % and 200 %. SMAPE compares the difference

between estEED and realEED with the mean of these two values, thus treating over and under estimations equally, avoiding distortion on the average value.

$$EED\_Error_{(SMAPE)}(\%) = \frac{1}{N}\sum_{n=1}^{N}\frac{|estEED_n - realEED_n|}{(estEED_n + realEED_n)/2} \quad (5)$$

The results obtained for the EED_Error and its confidence interval are shown in Fig. 2. Different β values were used for IGIs of 1, 2.5, 5, and 10 s. The results show that for high network loads (lower IGIs) a high β provided the lowest EED_Error, while for low network loads (IGI above 2.5 s) a lower β should be used. Whenever a node is experiencing a high network load, the EED values will vary with a higher amplitude, thus, in order to enhance EED estimation, the last EED sample must have a higher weight than the EED history. In short, a high β value should be used in high network loads.

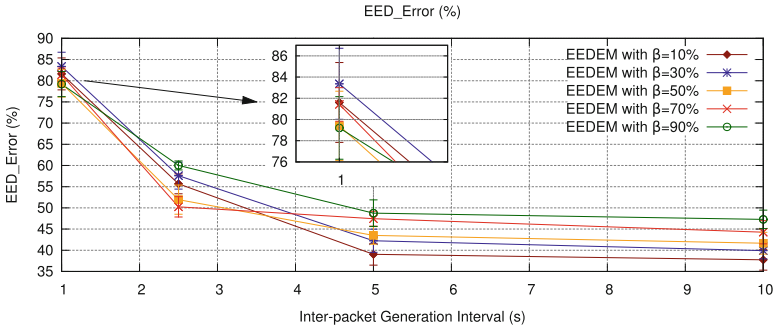

Fig. 2. EED_Error using β varying from 10 % to 90 %

4 Delay Accounting Optimization Procedure

The preliminary experiments demonstrated that, in order to minimize the EED_Error, each node must be aware of its network load. Our proposal Delay Accounting Optimization Procedure (DAOP) infers the network load by monitoring the real-time usage of the MAC queue and then, based on the size of the queue, selects the best β value and applies it in all internal timers. Figure 3 shows how the DAOP is integrated within the EEDEM. The DAOP assumes 4 intervals within the MAC-queueing block: i1, i2, i3, and i4. In interval i1 (from 0 up to 2 packets in the MAC queue) the DAOP assumes a low network load, in interval i2 (3 or 4 packets) and i3 (5 or 6 packets) the DAOP assumes a medium network load, and in interval i4 (from 7 up to the queue limit, i.e. 8 packets) it assumes a high network load. When a node sends a packet the queue usage is monitored and for intervals i1, i2, i3, or i4, a β value of 10 %, 30 %, 50 % or 70 % is applied, respectively, in all internal timers (β of 90 % was not used since it introduces higher EED_Error using DAOP). Since β values are calculated when packets are sent, the computational cost of DAOP will grow linearly with the sent packets, i.e., the procedure complexity is *O(n)*.

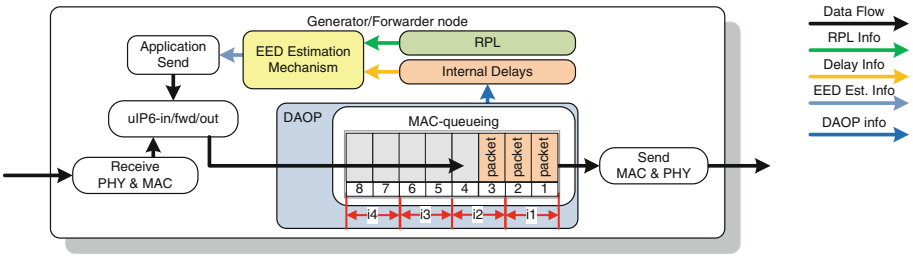

Fig. 3. DAOP integration in EEDEM

5 Results

The proposed solution monitors the MAC queue usage to infer the network load in real-time. Figure 4 shows the usage of the MAC queue for two cases: when the IGI is equal to 1, and when the IGI is equal to 5. The values were obtained in a node one hop away from to the sink, whenever a packet is to be sent. The results show that, for lower IGIs, the MAC queue has roughly 6 or

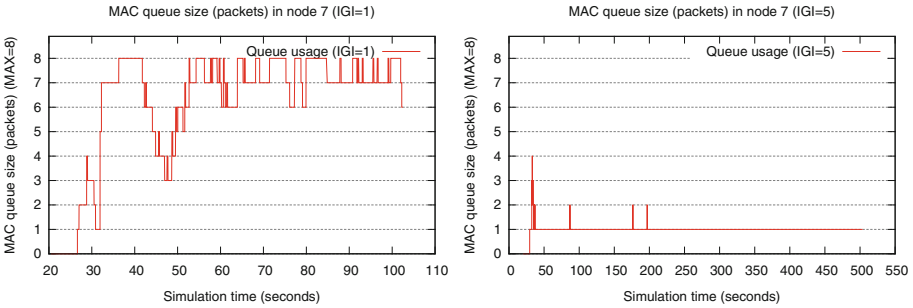

Fig. 4. MAC queue usage. Left: IGI=1 Right: IGI=5

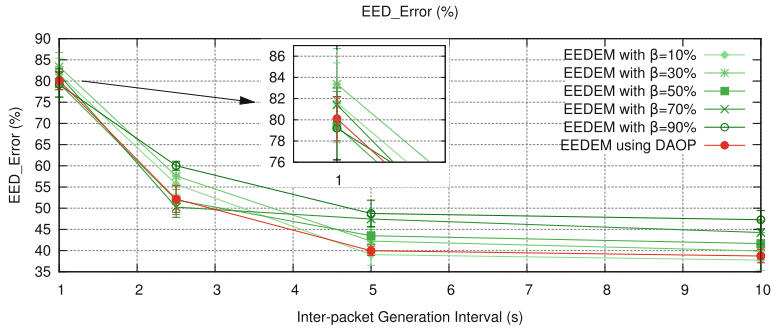

Fig. 5. EED_Error for EEDEM using different β and using DAOP

more packets, on average, and for an IGI equal to 5, the MAC queue has roughly 1 packet during all the simulated time.

Figure 5 compares the EED_Error obtained using the proposed solution with those obtained with constant β values of 10 %, 30 %, 50 %, 70 % and 90 %, for different IGI values. The results show that, by monitoring the MAC queue usage, the proposed DAOP dynamically infers network load and applies a β value that matches the best ones for each IGI in the preliminary experiments. Thus, DAOP presents the lowest EED_Error for all the different network loads.

6 Conclusions

Our previous proposal to estimate EED accounts for internal delays and uses RPL to feedback delays to the remaining nodes. The internal delays are accounted using an EWMA function, where the smoothing factor β is constant and defined *a priori*. Experimentation showed that the best EED estimation error results are obtained by varying the β value as a function of the network load.

This paper proposes a delay accounting procedure that dynamically adapts the β value inferring the network load by actively monitoring the node's MAC queue size. The results obtained show that the current solution provides a more accurate EED estimation for different network loads than our previous solution.

References

1. Pinto, P., Pinto, A., Ricardo, M.: End-to-end delay estimation using RPL metrics in WSN. In: IFIP Wireless Days (WD), pp. 1–6 (2013)
2. Winter, E.T., Thubert, E.P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Struik, R., Vasseur, E.J.P., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012
3. Pinto, P., Pinto, A., Ricardo, M.: RPL modifications to improve the end-to-end delay estimation in WSN, In: Proceedings of the 11th International Symposium on Wireless Communication Systems (ISWCS), Barcelona, Spain, August 2014
4. Nembharda, H.B., Kaoa, M.S.: Adaptive forecast-based monitoring for dynamic systems. Technometrics **45**(3), 208–219 (2003)
5. Li, H., Cheng, Y., Zhou, C., Zhuang, W.: Routing metrics for minimizing end-to-end delay in multiradio multichannel wireless networks. IEEE Trans. Parallel Distrib. Syst. **24**(11), 2293–2303 (2013)
6. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with COOJA. In: Proceedings of 31st IEEE Conference on Local Computer Networks, pp. 641–648 (2006)
7. Tmote Sky Project. http://www.snm.ethz.ch/Projects/TmoteSky
8. Contiki OS. http://www.contiki-os.org