MAP tele

DOCTORAL PROGRAMME IN TELECOMMUNICATIONS

Admission Control based on End-to-end Delay Estimation to Enhance the Support of Real-Time Traffic in Wireless Sensor Networks

Pedro Filipe Cruz Pinto

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy (PhD) in
Telecommunications

Supervisor: Manuel Alberto Pereira Ricardo (PhD)
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Co-Supervisor: António Alberto dos Santos Pinto (PhD)
Professor Adjunto do Instituto Politécnico do Porto

September, 2015

The Jury

—— President ——

Doutor Aurélio Joaquim de Castro Campilho
Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

—— Examiners Committee ——

Doutor Edmundo Heitor Silva Monteiro
Professor Catedrático da Faculdade de Ciências da Universidade de Coimbra

Doutor António Manuel Raminhos Cordeiro Grilo
Professor Auxiliar do Instituto Superior Técnico da Universidade de Lisboa

Doutor Rui Luís Andrade Aguiar
Professor Catedrático da Universidade de Aveiro

Doutor Manuel Alberto Pereira Ricardo
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Doutor Paulo José Lopes Machado Portugal
Professor Associado da Faculdade de Engenharia da Universidade do Porto

MAPtele

DOCTORAL PROGRAMME IN TELECOMMUNICATIONS

is a joint Doctoral Programme provided by

Universidade do Minho
Escola de Engenharia

universidade
de aveiro

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

This work has been done in context of the
SELF-PVP project CMU-PT/SIA/0005/2009 (FCOMP-01-0124-FEDER-013070) financed by the European Regional Development Fund (ERDF) through the COMPETE Programme and by national funds through the Fundação para a Ciência e a Tecnologia (FCT)
and the
BEST CASE project (NORTE-07-0124-FEDER-000056) financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the ERDF, and by national funds, through the FCT.

To Xana, Rita, David, and Henrique

Abstract

Wireless Sensor Networks (WSNs) are an essential enabler of the Internet of Things (IoT) concept that envisions a world of interactions between objects. In a WSN, the objects are small computers equipped with sensor, control and wireless communications capabilities. The WSN nodes specifications are driven by energy constraints since they use batteries and may be required to operate over long periods of time. As a consequence, these nodes' hardware use low-power electronics and their operation is often defined by limited processing and communications capabilities.

This thesis considers the case study of a solar smart grid, where each solar panel is equipped with a WSN node that may generate real-time streams towards a sink. Real-time monitoring or video surveillance are examples of such applications. The real-time traffic generated by WSN nodes demands from the network a service characterized by parameters such as delay, packet loss, and throughput. In particular, we focus this work on guaranteeing a maximum End-to-End Delay (EED) at the application layer for packets transported by the WSN. A packet will be considered useful if delivered at the destination within the expected maximum EED, and useless otherwise. The transmission of useless packets consumes processing and communications resources, and contributes negatively to the congestion of the system.

The thesis aims to enhance the WSN support for real-time applications and efficiently use the WSN resources, by exploring the hypothesis that potential useless data packets should not be transmitted by the source node. Therefore, the thesis provides two major contributions: 1) a real-time mechanism to estimate packet EED based on IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL); 2) a cross-layer admission control mechanism that decides if a packet should progress towards its destination, based on the EED estimation available in each network node.

The proposed EED estimation mechanism was evaluated and the results obtained reveal that internal processing delays of the nodes are significant and they should be considered in order to accurately forecast the packet EED; RPL was also found to be usable as the instrument for enabling the distributed estimation of EED. The packet admission control mechanism was also evaluated and the results obtained show that it actively contributes to decrease the number of the useless packets in transit in the WSN, consequently increasing the number of useful packets received at the destination, and improving the energy efficiency of each node, particularly under high network loads.

Keywords: WSN. Delay Estimation. End-to-End Delay. RPL. Admission Control.

Resumo

As Redes de Sensores Sem Fios (RSSF) potenciam o conceito da Internet das Coisas que prevê um mundo de interações entre objetos. Numa RSSF, os objetos são pequenos computadores equipados com sensores, controlos e recursos para comunicação sem fios. As especificações dos nós de uma RSSF são orientadas por restrições de energia uma vez que estes utilizam baterias e podem ter de operar por longos períodos de tempo. Como consequência, o hardware destes nós utiliza eletrónica de baixa potência e a sua operação é geralmente definida por capacidades limitadas de processamento e de comunicação.

Esta tese considera o caso de estudo de uma central solar inteligente, onde cada painel solar está equipado com um nó de uma RSSF e pode gerar fluxos em tempo real para um nó de destino. A monitorização ou a vídeo vigilância são exemplos de tais aplicações. O tráfego em tempo real, gerado pelos nós da RSSF, exige da rede um serviço caracterizado por parâmetros tais como atraso, perda de pacotes e taxa de transferência. Em particular, este trabalho foi focado em garantir um Atraso Extremo-a-Extremo (AEE) máximo ao nível da camada de aplicação para os pacotes transportados pela RSSF. Um pacote será considerado útil se for entregue no destino dentro de um AEE máximo esperado, caso contrário, será inútil. A transmissão dos pacotes inúteis consome recursos de processamento e de comunicação, e contribui negativamente para o congestionamento do sistema.

A tese tem como objectivo melhorar o suporte da RSSF para aplicações em tempo real, utilizando de forma eficiente os seus recursos, explorando a hipótese de que os pacotes potencialmente inúteis não devem ser transmitidos pelo nó fonte. Assim, a tese apresenta duas contribuições principais: 1) um mecanismo para estimar em tempo real o AEE de um pacote baseado no protocolo de encaminhamento RPL; 2) um mecanismo de controlo de admissão inter-camadas que decide se um pacote deve progredir para o seu destino, com base na estimativa do AEE disponível em cada nó da rede.

O mecanismo proposto para a estimativa do AEE foi avaliado e os resultados obtidos revelam que os atrasos de processamento internos aos nós são significativos e que devem ser considerados na previsão do AEE de um pacote; ao mesmo tempo, o RPL também se apresentou como um instrumento útil para permitir a distribuição da estimativa do AEE para todos os nós da rede. O mecanismo de controlo de admissão de pacotes foi também avaliado e os resultados obtidos mostram que este contribui ativamente para diminuir o número de pacotes inuteis em trânsito na RSSF, aumentando consequentemente o número de pacotes úteis recebidos no destino, e melhorando a eficiência energética de cada nó, particularmente quando a rede está sobrecarregada.

Keywords: RSSF. Estimativa de Atraso. Atraso Extremo-a-Extremo. RPL. Controlo de Admissão.

Acknowledgements

First, I would like to thank Prof. Manuel Ricardo for his precious guidance. His critical insights, decisive support, and valuable advices have made this work possible. I also would like to thank Prof. António Pinto for his continuous support and valuable suggestions that enabled me to overcome many obstacles found during this work.

I would like to thank INESC TEC for providing me with a leading research environment and, in particular, to all my colleagues in the Wireless Networks (WiN) research group at the Centre for Telecommunications and Multimedia (CTM), with whom many of the ideas of this work were discussed and improved due to their contribution.

I would like to thank Instituto Politécnico de Viana do Castelo (IPVC) for funding and supporting, in part, my involvement in the MAP-Tele doctoral programme; I would also like to thank my close professional colleagues that, from the start, encouraged me and provided an extra motivation to pursue this objective.

I would like to thank my parents for their life values and neverending support, and my extended family and close friends for their support and understanding during this journey.

Finally, I would like to specially thank and dedicate this thesis to Xana, Rita, David, and Henrique.

Pedro Pinto

"Ao trabalho corresponde o fruto que se colhe." in Cartas (1654)

Vieira, António

Contents

List of Figures

List of Tables

List of Abbreviations

#Nodes	Number of Nodes [90, 91]
#RcvdPkts	Number of Received Packets [59, 60, 71, 75, 82, 88, 89]
#SentPkts	Number of Sent Packets [59, 60, 89]
#UsefulPkts	Number of Useful Packets [82]
6LowPAN	IPv6 over Low Power Wireless Personal Network [39–41]
AC	Admission Control [xi, 9, 26–32, 49, 81, 82, 103, 105]
ACK	Acknowledge [xi, 16, 17, 42–45, 55, 106]
ACManager	Admission Control Manager [84–88]
API	Application Programming Interface [84, 85]
APP	Application [10, 54, 55]
AppAPI	Application API [84–86, 88]
AppID	Application ID [85]
ARP	Address Resolution Protocol [37]
ARQ	Automatic Repeat-reQuest [16, 106]
AS	Autonomous System [14, 26–29]
ASH	Auxiliary Security Header [39]
BPSK	Binary Phase-Shift Keying [35]
CAIDA	Center for Applied Internet Data Analysis [20]

CBR Constant Bit Rate [20, 58, 88]

CCA Clear Channel Assessment [36]

CDMA Code Division Multiple Access [20]

CLAC Cross-layer Admission Control [xii, 81–101, 104–106]

CP Candidate Parent [41, 44, 66, 67]

CSMA Carrier Sense Multiple Access [66]

CSMA/CA Carrier Sense Multiple Access / Collision Avoidance [36]

DAG Directed Acyclic Graph [xi, 41–47, 57, 58, 66, 67, 69, 70]

DAO Destination Advertisement Object [xi, xiii, 42–45, 66]

DAOP Dynamic Accounting Optimization Procedure [xii, xiii, 75–80, 104]

DHCP Dynamic Host Control Protocol [37]

Diffserv Differentiated Services [26, 30]

DIO DAG Information Object [xi, xiii, 42–46, 66, 67, 69, 70, 73, 88]

DIS DAG Information Solicitation [xi, xiii, 42, 44, 45, 66]

DNS Domain Name Service [37]

DODAG Destination-Oriented Directed Acyclic Graph [41]

DSSS Direct-Sequence Spread Spectrum [35]

DstID Destination ID [85]

DTSN Destination Advertisement Trigger Sequence Number [42]

EED End-to-End Delay [xii, 2, 4, 5, 7, 9, 11, 15, 18, 28, 31, 34, 51–53, 57–60, 62–66, 70, 71, 73–85, 88–91, 93, 99, 103–105]

EED_RE EED Rough Estimation [67–69]

EEDEM EED Estimation Mechanism [xii, xiii, 52–55, 57–67, 70, 71, 73, 74, 76–81, 88, 99, 103–106]

EEDError EED Estimation Error [xii, 59–61, 71, 72, 75–79, 88, 91, 92]

EstEED Estimated EED [85, 86, 88]

EstIF EED Estimation Interface [84–86, 88]

ETT Expected Transmission Time [23, 24, 59, 60, 62, 63, 65, 71, 73, 74, 78, 79]

ETX Expected Transmission Count [23, 47, 59, 63]

EWMA Exponential Weighted Moving Average [20–22, 26, 55, 74, 79]

FFD Full Function Device [35]

FIFO First-In-First-Out [32]

FwdD Forward Delay [55, 56]

FwdLinkD Forward Link Delay [56, 57]

FwdProcD Forward Processing Delay [56, 57]

GenD Generation Delay [55, 56, 86]

GenLinkD Generation Link Delay [56, 58]

GenProcD Generation Processing Delay [56, 58]

GPS Global Positioning System [15–17, 20, 25]

GTS Guaranteed Time Slot [36]

HC Header Compression [40]

HopMetric Hop Count Metric [67–70]

HystV Hysteresis Value [67, 69]

IANA Internet Assigned Numbers Authority [46, 47]

ICMP Internet Control Message Protocol [37, 40]

ID Identifier [35, 42, 43]

IEEE Institute of Electrical and Electronics Engineers [xi, 12, 18, 24, 30–37, 39, 40, 48, 106]

IETF Internet Engineering Task Force [xi, 19, 39, 41]

IGI Inter-packet Generation Interval [xii, 59, 60, 62, 63, 65, 70, 71, 73–75, 77, 78, 88, 91, 95, 99]

IGMP Internet Group Management Protocol [37]

Intserv Integrated Services [26]

IoT Internet of Things [1, 34, 37, 106]

IP Internet Protocol [xi, xiii, 1, 9–15, 18, 20, 25, 26, 32, 34, 37–43, 48, 49, 103]

IPPM IP Performance Metrics [19, 20]

IPR In-profile Packet Ratio [xii, 89, 90, 95, 96]

LLN Low-power and Lossy Network [41, 49]

LR-WPAN Low-rate Wireless Personal Area Network [34]

lwIP lightweight IP [xi, 37–39, 48]

MA Moving Average [20–22, 26]

MAC Media Access Control [xii, 11, 12, 24, 35, 39, 54, 55, 58, 68, 70, 76–79, 104]

MAE Mean Absolute Error [xii, 24, 59–61]

MAPE Mean Absolute Percentage Error [xii, 24, 25, 59–61, 71, 72, 88, 91, 92]

MaxEED Maximum EED [81, 82, 85, 86, 88, 91, 95, 99]

MBAC Measurement Based Admission Control [30, 31]

MCU MicroController Unit [33, 34, 90]

MEMS Micro-ElectroMechanical Systems [1]

MIC Message Integrity Code [39]

MinHystV Minimum Hysteresis Value [69]

MIPS Million Instructions Per Second [33]

MOP Mode of Operation [42, 66]

MP2P MultiPoint-to-Point [44]

MRHOF Minimum Rank with Hysteresis Objective Function [xi, 41]

MSE Mean Square Error [24]

MTU Maximum Transmission Unit [39]

NDP Neighbor Discovery Protocol [37]

NetAPI Network API [84, 85, 87, 88]

NTP Network Time Protocol [15–17, 25]

OCP Objective Code Point [46]

OF Objective Function [41, 46, 66, 79]

OF0 Objective Function Zero [41]

OPR Out-of-profile Packet Ratio [xii, 89, 90, 95, 96]

OS Operating System [xiii, 33, 34, 41, 47–49, 54, 88]

OSI Open Systems Interconnection [10]

OWAMP One-Way Active Measurement Protocol [20]

OWD One-Way Delay [14, 15, 17–20, 22, 23, 25, 26]

OWPP One-Way Probe Packet [16, 17]

P2MP Point-to-Multipoint [44]

P2P Point-to-Point [44]

PAN Personal Area Network [34, 35, 37]

PathD Path Delay [xii, 57–59, 62, 63]

PathDMetric Path Delay Metric [57, 58, 63, 66, 67, 70]

PBAC Parameter Based Admission Control [30, 31]

PCS Path Control Size [46]

PDU Protocol Data Unit [10, 39]

PHD Per-Hop Delay [xi, 14–18, 23, 25, 26]

PHY Physical [10, 11, 39, 49, 54, 58, 70]

PP Preferred Parent [41, 44, 66, 67, 69, 70]

pp percentage points [65, 71, 74, 95]

PPFC Packet Pair Flow Control [17, 23]

PPP Point-to-Point Protocol [37]

ProcD Processing Delay [xii, 11, 12, 25, 34, 57–59, 62, 63]

ProcDMetric Processing Delay Metric [57, 58, 63, 66, 67, 70]

PropD Propagation Delay [12, 16, 17, 25]

PRR Packet Reception Ratio [xii, 60, 65, 66, 71, 74, 78, 79, 82, 89–91, 94, 103, 104]

PSTN Public Switched Telephone Networks [26]

PUR Packet Usefulness Ratio [xii, 82, 89, 90, 95, 97]

QoS Quality of Service [xi, 2, 20, 26–32]

QPSK Quadrature Phase-Shift Keying [35]

QueueD Queue Delay [12, 25, 54–56, 88]

RA-EEDEM RPL Adaptation for EEDEM [xii, xiii, 65–74, 79, 80, 104]

RAM Random-Access Memory [33, 34, 37, 39, 48]

RcvD Receiver Delay [55, 56]

RcvLinkD Receiver Link Delay [56, 57]

RcvProcD Receiver Processing Delay [56, 57]

RFC Requests For Comments [19, 20, 39–42, 46, 47]

RFD Reduced Function Device [35]

RIPE NCC Réseaux IP Européens Network Coordination Centre [11]

RMSE Root Mean Square Error [24]

ROLL Routing Over Low-power and Lossy networks [xiii, 41, 47, 57, 67]

ROM Read-Only Memory [33, 34, 37, 39, 48]

RPL IPv6 Routing Protocol for Low-Power and Lossy Networks [xi–xiii, 5, 26, 34, 41–47, 49, 52, 57–59, 63–67, 70–72, 74, 78, 79, 83–85, 88, 104–106]

RPLIF RPL Interface [84, 85, 88]

RTD Round Trip Delay [14, 19]

RTT Round Trip Time [14, 18, 23]

SELF-PVP SELF-organizing power management for Photo-Voltaic Power plants [2]

SeqNr Sequence Number [85]

SICS Swedish Institute of Computer Science [34]

SMAPE Symmetric MAPE [xii, 24, 25, 75–79, 88]

SNMP Simple Network Management Protocol [37]

SrcID Source ID [85]

TCP Transmission Control Protocol [xi, 10, 32, 37, 38]

TotalDMetric Total Delay Metric [67, 69]

TransD Transmission Delay [12, 16, 17, 25, 34, 54–56, 88]

TUF Time Utility Function [xi, 31, 32]

TWAMP Two-Way Active Measurement Protocol [20]

TWD Two-Way Delay [14, 15, 17–19, 25]

TWPP Two-Way Probe Packet [17]

UDGM Unit Disk Graph Medium [58, 88]

UDP User Datagram Protocol [37, 40, 58, 70, 75, 88]

uIP micro IP [xi, 34, 37–39, 48, 66]

UP Usefulness Preview [xii, 82, 83]

WG Working Group [19]

WLAN Wireless Local Area Network [16]

WMA Weighted Moving Average [20–22, 26]

WMN Wireless Mesh Network [12, 18, 28]

WPAN Wireless Personal Area Network [34, 39]

WSN Wireless Sensor Network [xi, xii, 1–5, 7, 9, 12, 16, 18, 22, 25, 26, 30–34, 41, 47–49, 51–53, 58, 59, 74, 78, 80–83, 88, 99, 103–106]

Chapter 1

Introduction

Recent advances in the scaling of electronic circuits and in providing them with the ability to interact with the world around, enabled the appearance of Micro-ElectroMechanical Systems (MEMS). MEMS technology combines very small computers with sensor and control capabilities. MEMS mass commercialization and distribution made it very cost-effective and suitable for multiple uses. The deployment of communications capabilities in MEMS fostered the appearance of new network architectures and their interconnection to the existing global Internet Protocol (IP) network leaded to the birth of the Internet of Things (IoT) concept. The IoT envisions a world of interaction and coordination between objects, with or without human intervention, for the creation of smart environments.

Wireless Sensor Network (WSN) architectures extend the IoT concept by giving wireless communications capabilities to MEMS. A WSN is composed of a large number of sensor nodes, where each node can be characterized as a very small computer with a wireless interface. These nodes generate data from their sensors, such as temperature, humidity, moisture, and pressure, among others, and forward this data towards a gateway node. The gateway node, in turn, connects these networks to the Internet, as shown in Figure 1.1. WSN applications are multifold in areas such as smart metering, health care, environmental sensing, home automation, sports and wellness.

The hardware of the sensor nodes in a WSN is designed with processing and communications constraints since these nodes have limited energy resources. Even though these hardware limitations exist, more recently new and more complex applications and services (e.g. audio and video streaming) are pushed to be supported by the WSNs, in order to foster the concept of the IoT. These initiatives create new challenges in networking research areas such as routing, management, quality of service and energy efficiency.

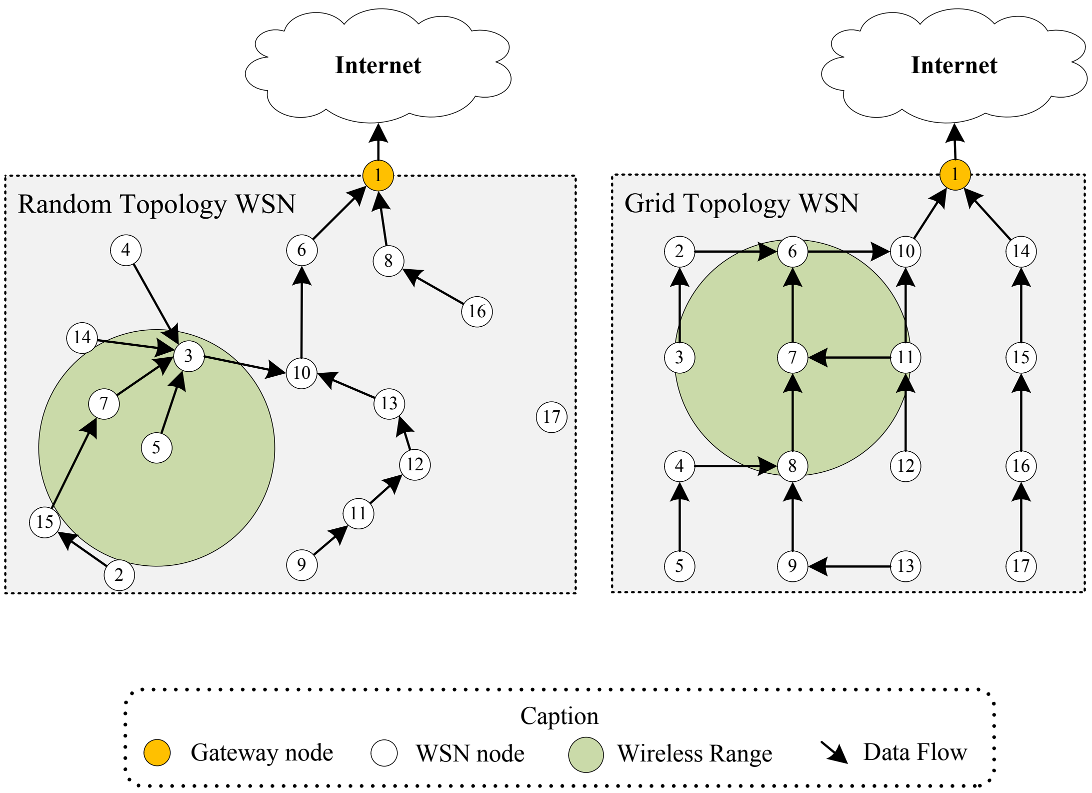


Figure 1.1: WSN with random and grid topologies

1.1 Scope and Motivation

This thesis was carried out in the scope of the SELF-organizing power management for Photo-Voltaic Power plants (SELF-PVP) project [1] that aimed to increase the efficiency of a photo voltaic power plant with approximately 200.000 solar panels distributed in an area of 250 hectares. The solar panels include sensor nodes that communicate with each other using a grid topology WSN as shown in Fig. 1.2. In this scenario, we aim to deploy real-time applications, such as monitoring or video surveillance, in a set of sensor nodes.

Real-time applications typically generate traffic flows with Quality of Service (QoS) requirements that can be defined in terms of delay, jitter or packet loss. In case these applications require strict delay boundaries from source to destination, their packets must be delivered to the destination application within an End-to-End Delay (EED) limit in order for the information to be considered useful. The packets delivered outside the defined EED limit will be considered useless and discarded by these applications at the destination.

In order to enhance the operation of these applications, and since WSN nodes have relevant

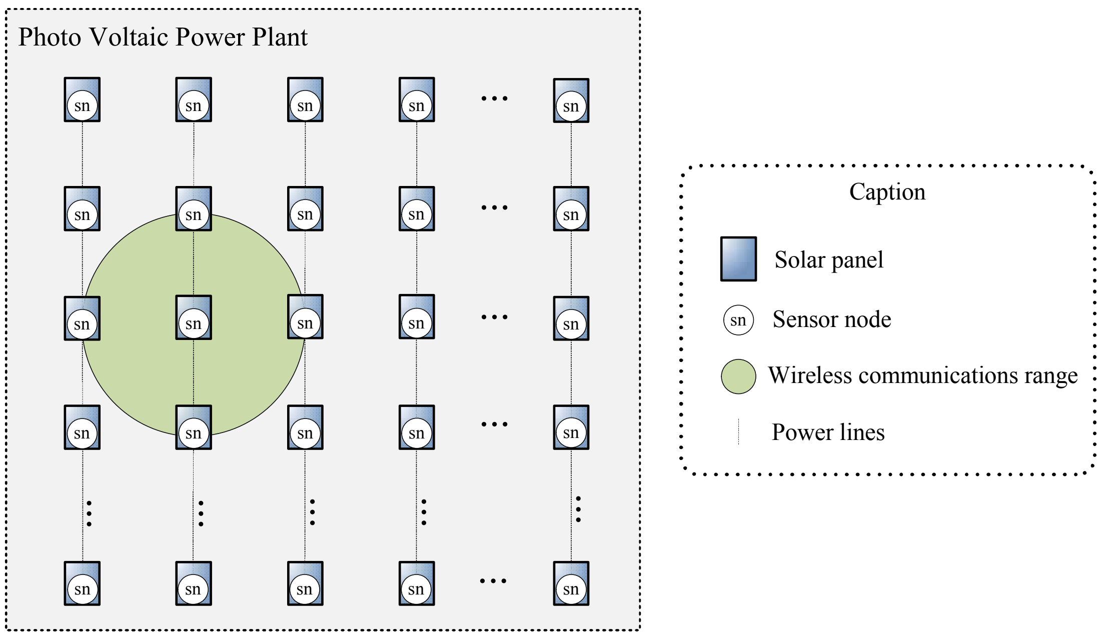


Figure 1.2: Photo Voltaic Power Plant

processing and communications constraints, we explore the idea that the WSN should avoid processing and transporting useless packets and use its full potential to maximize the number of delivered useful packets. Therefore, our research is oriented towards the enhancement of the performance of a grid WSN considering the application's viewpoint, while taking into consideration the efficient use of the available resources.

1.2 Problem Statement

A real-time application is to be deployed on a grid WSN where each node has limited resources in terms of processing, communications and energy. The real-time application generates delay sensitive flows with data that is assumed to be useful for the destination only if it is received within a strict delay boundary, and useless otherwise. In order to enhance the support for this application, the WSN performance can be oriented to maximize the number of delivered useful packets. At the same time, since WSN nodes have relevant processing, transmission and energy constraints, they should avoid to process and transport the useless packets.

The main problem to address is that the usefulness of a packet is determined at its destination, and processing, transmission and energy resources have already been expended to transport the packet. Since the destination application may not consider all received packets as useful, if we are able to identify, as soon as possible, which packets will likely miss the application delay deadlines and avoid their transmission to the network, an increase in network performance and energy efficiency is expected to be achieved.

1.3 Objective

The main objective of this thesis is to enhance the support of real-time applications in a grid WSN topology, as shown in Fig. 1.3. In this topology, each source node generates a delay sensitive data flow directed towards a central destination node.

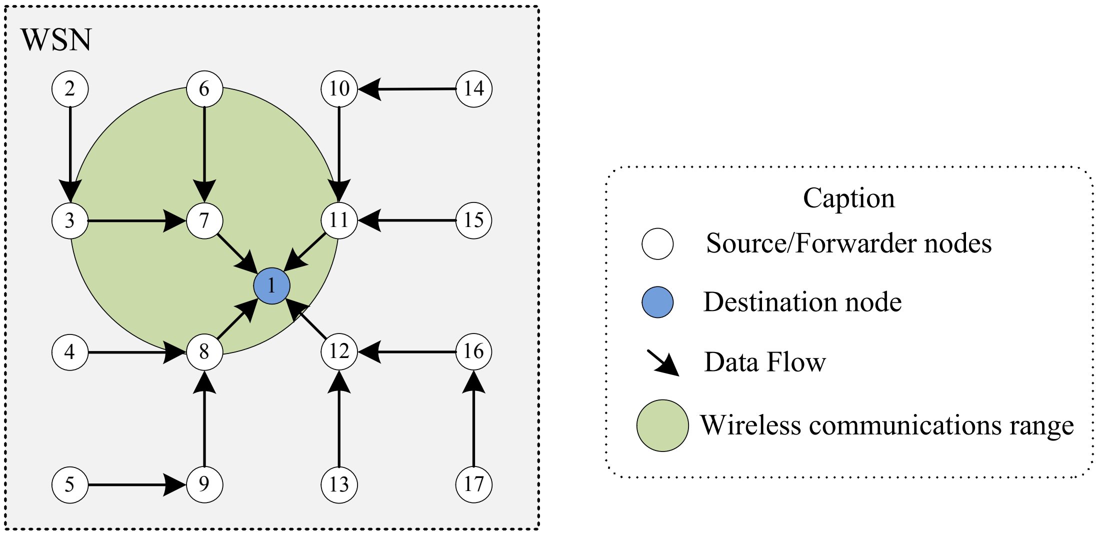


Figure 1.3: WSN topology

In each source node, the chosen strategy is to preview the EED of each packet and, as earlier as possible, avoid packet transmissions when these are expected to not comply with the limits given by the application. In order to pursue this strategy, the research efforts are divided in two particular objectives:

- Provide an EED estimation mechanism to be deployed in a WSN with minimal impact on network performance;
- Provide a WSN admission control mechanism based on the EED estimation and intended to enhance network performance and foster energy efficiency.

1.4 Contributions

This thesis provides two main original contributions:

- **Novel mechanism to estimate EED based on RPL routing protocol**

    ∗ In order to preview if a packet will be delivered within the EED limit defined by the application, a novel EED estimation mechanism is proposed. Other delay estimation mechanisms are proposed in literature but some of them do not provide a real-time and per-packet delay estimation, while others introduce additional traffic in the WSN to provide estimations. The proposed EED estimation mechanism provides a real-time and per packet EED estimation using IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL). RPL packets are used to feedback the EED delay of the previously sent packets to the source nodes, thus avoiding extra traffic in the WSN. Also, to enhance EED estimation accuracy, this proposal accounts not only with transmission delays but also with the in-node processing delays which are relevant in the context of the limited processing resources of the nodes. This contribution has been published in [2]. Also, a set of RPL modifications to enhance the accuracy of EED estimation were proposed, and published in [3]. In the context of the EED estimation mechanism and in order to enhance EED estimation when using multiple network loads, a delay accounting optimization procedure was also proposed, and published in [4].

- **Novel cross-layer admission control mechanism based on the EED estimation**

    ∗ In order to decide if a packet should be transmitted accordingly to their usefulness to the destination application, a novel cross-layer packet admission control mechanism is proposed. The proposed admission control mechanism is distributed by the WSN nodes and it is responsible for the decisions to transmit or drop a packet according to the requirements defined by the application. Other admission control mechanisms are proposed in the literature but the novelty of the proposed mechanism is that it runs in a cross-layer operation mode involving the application and network layers, while implementing interfaces with the EED estimation mechanism and RPL routing protocol. This contribution has been accepted for publishing in [5].

1.5 Publications

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“Data and Path Aggregation in Large Scale and Cluster-based Wireless Sensor Networks”**, in MAPTele Workshop, Aveiro, Portugal, May 2011.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“Secure Data and Path Aggregation in WSN (Poster)”**, in MAPTele Workshop, Porto, Portugal, Jun. 2012.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“End-to-end Delay Estimation using RPL Metrics in WSN”**, in Proceedings of the Wireless Days (WD’2013), IFIP, Valência, Spain, Nov. 13-15, 2013, pp. 1–6.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“RPL Modifications to Improve the End-to-end Delay Estimation in WSN”**, in Proceedings of the 11th International Symposium on Wireless Communications Systems (ISWCS), IEEE, Barcelona, Spain, Aug. 2014, pp. 868–872.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“Reducing WSN Simulation Runtime by using Multiple Simultaneous Instances”**, in Symposium on Modelling and Simulation in Computer Sciences and Engineering (ICNAAM 2014), Rhodes, Greece, Sep. 2014.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“Delay Accounting Optimization Procedure to Enhance End-to-End Delay Estimation in WSNs”**, in Proceedings of the 8th International Wireless Internet Conference (WICON 2014) - Symposium on Wireless and Vehicular Communication, Lisbon, 2014.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **“Reducing Simulation Runtime in Wireless Sensor Networks: A Simulation Framework to Reduce WSN Simulation Runtime by Using Multiple Simultaneous Instances (Book Chapter),”** in Handbook of Research on Computational Simulation and Modeling in Engineering, IGI Global, 2016. 726-741. 8 Sep. 2015. ISBN: 978-1-4666-8823-0.

- Pedro Pinto, António Pinto, Manuel Ricardo, **“Delay Accounting Optimization Procedure to Enhance End-to-End Delay Estimation in WSNs”** (Book Chapter), Wireless Internet Book - Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Revised Selected Papers of the 8th International

Conference, WICON 2014, Lisbon, Portugal, Nov. 13-14, 2014. Volume 146. May 21th, 2015. ISBN: 978-3-319-18801-0.

- Pedro Pinto, António Pinto, and Manuel Ricardo, **"Cross-layer Admission Control Mechanism to Enhance the Support of Real-time Applications"**, Sensors Journal, IEEE, vol. 15, no. 12, pp. 6945–6953, Dec. 2015.

1.6 Document Structure

The structure of this thesis is as follows. Chapter 2 reviews the related work regarding the EED estimation and admission control research areas. It also describes specific operations and constraints present in the WSNs. Chapter 3 details and evaluates the EED estimation mechanism, and the implemented improvements that enhance the EED estimation. Chapter 4 details and evaluates the proposed packet admission control mechanism. Finally, Chapter 5 concludes the thesis and discusses future work.

Chapter 2

Delay Estimation and Admission Control in IP Networks

Real-time applications require specific EED boundaries (between source and destinations nodes) for all its packets. At the destination, if packets are delivered within the defined EED boundary they are considered useful; by contrary, if the packets are delivered outside this time boundary they will be considered useless and will be discarded at destination.

In order to preview the usefulness of a packet, the delay from source to destination must be estimated in each node. Section 2.1 presents the state of the art on delay measurement and estimation. In order to actively control the admission of new traffic and avoid transmitting packets that potentially will miss the defined EED limit, an Admission Control (AC) mechanism is necessary. Section 2.2 presents the state of the art on AC in IP networks.

Since the envisioned application must be supported by a WSN, the particular operation procedures and relevant constraints of these networks are also described in Section 2.3. This Chapter is summarized in Section 2.4.

2.1 Delay Estimation in IP Networks

The expansion of the packet-switched networks in early 90s facilitated the interconnection of different network architectures. However these networks provide little control over the packet delay at the forwarding nodes [6]. Since then, several research efforts were made in order to characterize, measure and estimate packet delays in the Internet, or global IP network.

2.1.1 Delays Definition

To be a part of an IP network, a node must implement an IP network stack. Fig. 2.1 presents the Open Systems Interconnection (OSI) and TCP/IP models, and a TCP/IP stack with a common combination of protocols which use IP as the center protocol. These protocols perform functions specified in the depicted models from Physical (PHY) layer to Application (APP) layer implemented in hardware, or software, or both. The TCP/IP model is used as reference in this thesis.

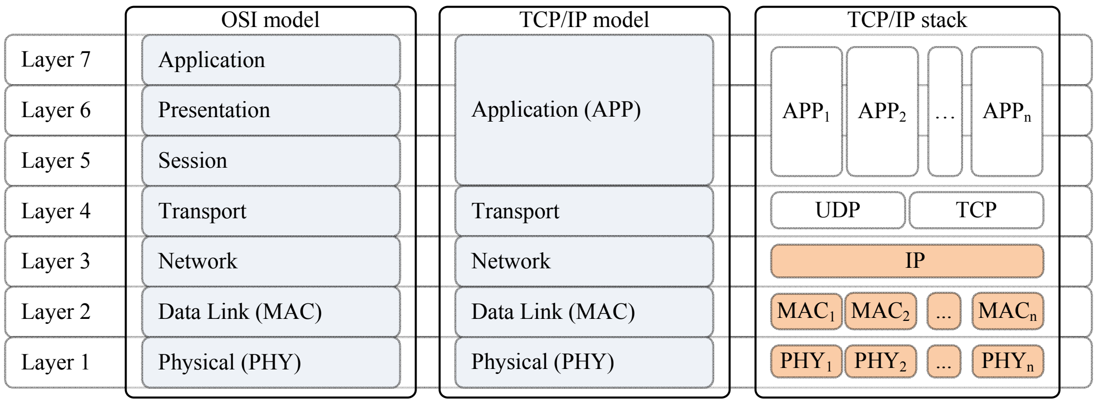


Figure 2.1: IP Stack

When an application sends data, it uses the set of protocols of the stack to transmit it from the source to the destination nodes according to Fig. 2.2. In this case the application APP$_1$ passes data to lower layers which in turn send it to the PHY layer. The router de-encapsulates the received information up to IP protocol an forwards it through another interface. The information eventually reaches the destination node. Since this information is successively encapsulated and de-encapsulated, different types of Protocol Data Unit (PDU) are formed (e.g. segments, datagrams, packets, or frames). For simplicity, these different PDUs are referred as packets for the rest of this thesis.

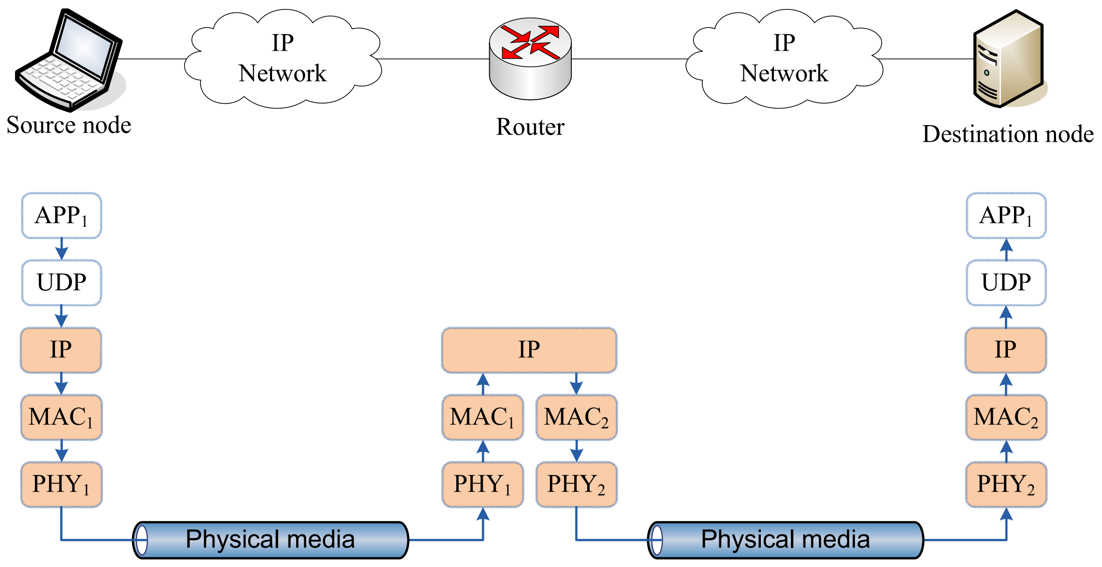


Figure 2.2: Typical communication between a source and a destination node

By definition, a delay is a time interval which is obtained by the difference of two time instants t_A and t_B, where $t_A < t_B$, as follows:

$$\text{Delay} = \Delta_t = t_B - t_A \tag{2.1}$$

In the IP networks, the time instants are collected at reference points. When a packet progresses from source to destination, different delay components can be accounted in each node. Fig. 2.3 presents a source node sending a packet to a destination node where the following four delay components can be depicted:

- Processing Delay (ProcD): is the time required to process a packet within a node. This delay includes not only the time elapsed while performing layer 3 and above layers tasks, but also the tasks within PHY and Media Access Control (MAC) layers when packet is received. In the IP networks this delay depends on the number of tasks and on the computational power available. In [7] the authors present an analysis of EED measurements in IP networks performed by Réseaux IP Européens Network Coordination Centre (RIPE NCC) by using probe packets where it is assumed that the ProcD is dependent of three factors: the protocol stack, the the computational power available at each node and the link driver. Also, the authors verified that the processing delays are not the same for different probe-packets due to the variability of tasks performed in the router and they split the processing delay in two parts: the stochastic and the deterministic. The processing delay component is often neglected by many research efforts when accounting delays in a IP network; however, this delay component may be significant on scenarios characterized by wireless devices with limited processing resources, such as those used in WSNs. As example, in [8] the authors present a delay analysis for a Wireless Mesh Network (WMN) using IEEE 802.11 devices in which each mesh node accounts with intra-node processing delays. In [9] and in [10] the authors account the ProcD to provide accuracy to their delay estimation.

- Queue Delay (QueueD): is the time elapsed since the packet enters the MAC queue, waiting for transmission, until it leaves this queue to be transmitted. This delay is essentially dependent of the network load. Queue theory [11] can provide an estimation for this delay component.

- Transmission Delay (TransD): is the time required to push all the packet into the physical media. This delay is proportional to the packet's length and can be obtained using:

$$\text{TransD (s)} = \frac{\text{length of packet (bit)}}{\text{rate of transmission (b/s)}} \tag{2.2}$$

- Propagation Delay (PropD): is the amount of time required to move a bit of the packet from one node to the next node. This delay is proportional to the distance between the nodes and can be obtained using:

$$\text{PropD (s)} = \frac{\text{distance between nodes (m)}}{\text{signal propagation speed (m/s)}} \tag{2.3}$$

  The signal propagation speed is dependent of the physical media used. For instance, in wireless networks the propagation speed is approximately the speed of light, denoted by the constant c, which is approximately 3.00×10^8 m/s; when using a copper wire the signal propagation is around 2/3 of c, i.e. approximately 2.00×10^8 m/s. For a distance of 10 m between nodes and using wireless medium, the PropD assume values around 33.3 ns and thus, in some scenarios, the PropD is neglected.

Using the delay definitions above, the total delay in a node can be obtained using the following equation:

$$\text{Total Delay (node)} = \text{ProcD} + \text{QueueD} + \text{TransD} + \text{PropD} \tag{2.4}$$

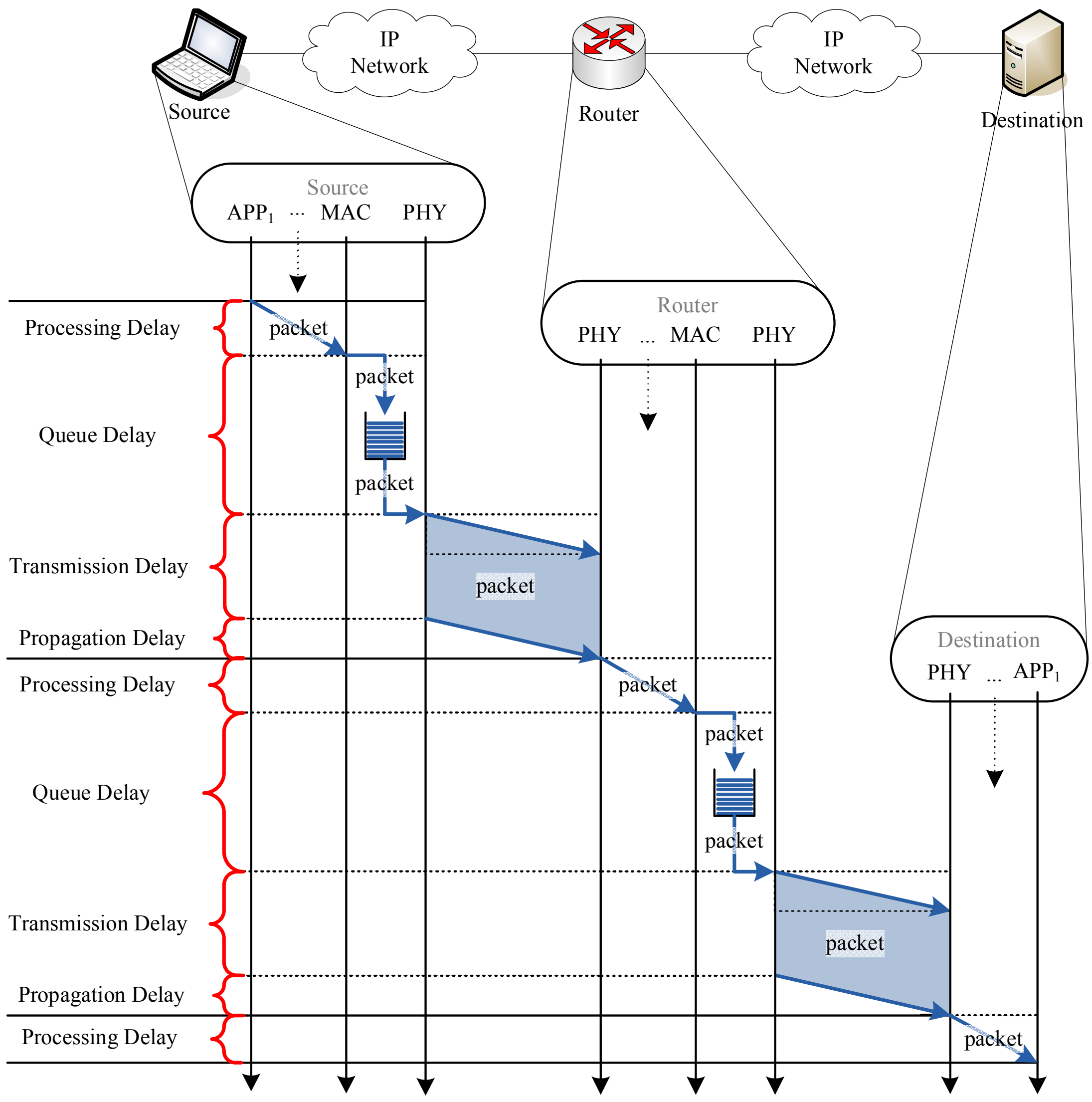


Figure 2.3: Delays within nodes in IP networks

2.1.2 Delays Measurement

In order to measure packet delays in IP networks two reference points are required. Timestamps must be collected when packets pass through these reference points and delays are accounted in a unique location by using these timestamps. Different scales can be used for these reference points; in a node scale these reference points may refer to code execution points; in a macro scale these reference points may refer to two routers located in distant geographical positions.

Measuring delays using Layer 3 reference points

Fig. 2.4 presents a set of layer 3 reference points and, using them, three types of delays can be defined: Per-Hop Delay (PHD), One-Way Delay (OWD), Two-Way Delay (TWD), also named Round Trip Delay (RTD) or Round Trip Time (RTT). The same definition is used in [12].

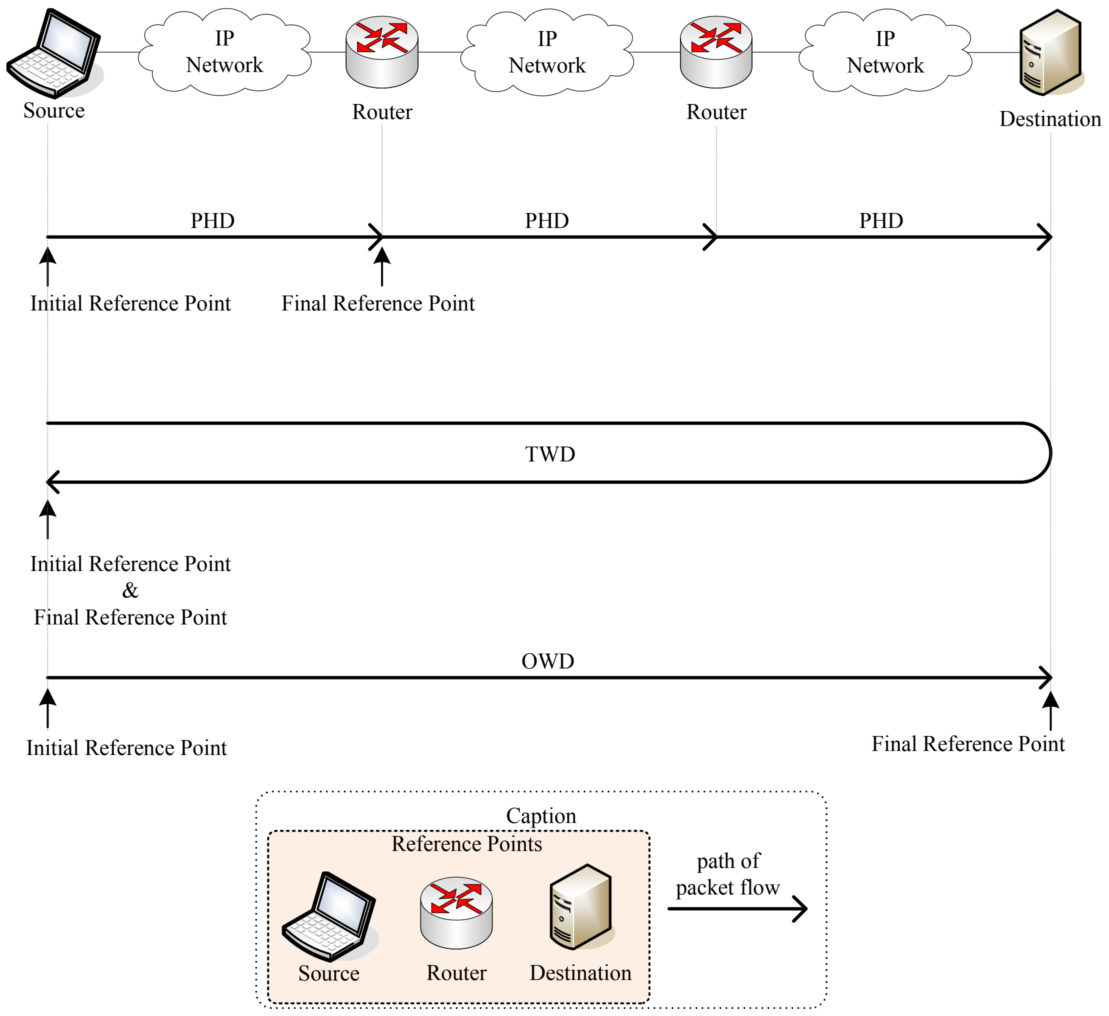


Figure 2.4: Reference points and delay measurement in IP networks

PHD accounts the delay experienced by a packet between two network reference points distant one hop from each other, named initial reference point and final reference point. The OWD is the delay of a one way packet between initial and final reference points involving multiple intermediate hops and IP networks, e.g. a source and a remote destination node or two edges of an Autonomous System (AS). In case the reference points are the packet source and destination, the OWD can be assumed as the packet EED. The TWD is the round trip delay of a packet, i.e, both initial and final reference points are in the same node or reference point.

Fig. 2.5 proposes a taxonomy regarding methods for measuring these delays. For each method it is indicated if the measurement is active or passive, i.e. if the measurement procedure generates additional traffic or not. Also, for each method it is indicated if a high or a low overhead is introduced when measuring the delay.

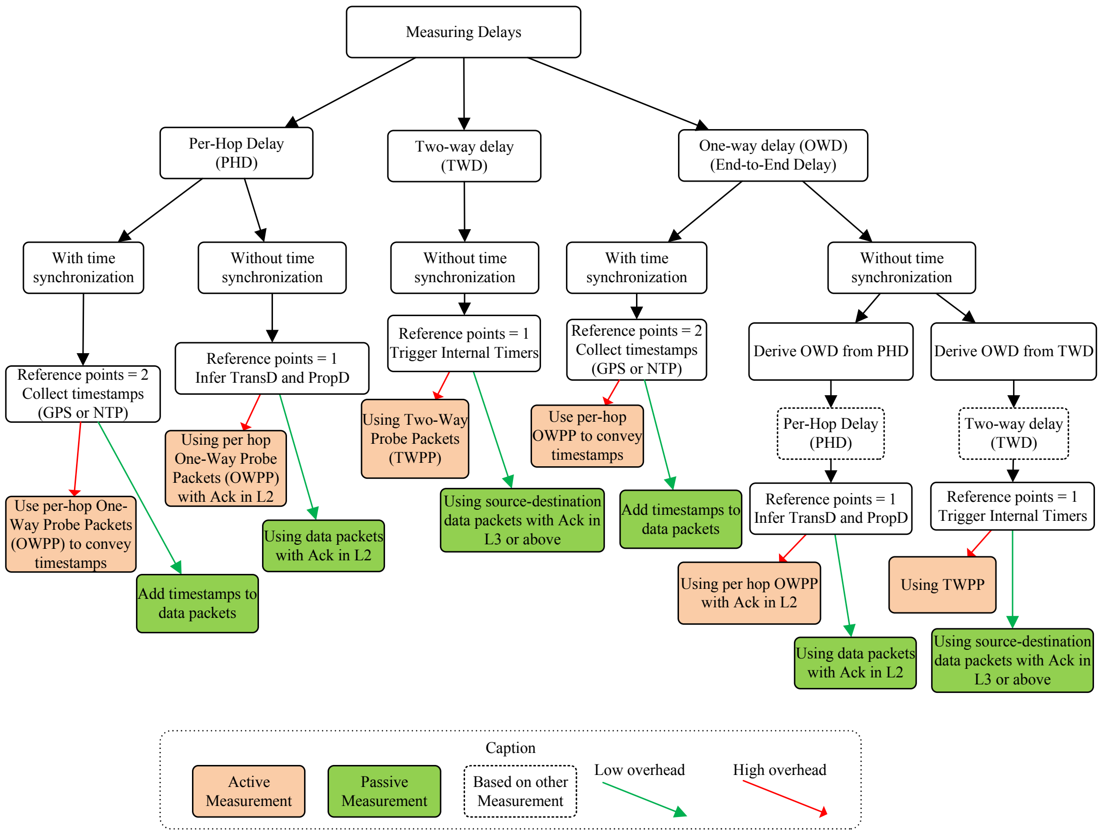

Measuring Delays
Per-Hop Delay (PHD)
Two-way delay (TWD)
One-way delay (OWD) (End-to-End Delay)
With time synchronization
Without time synchronization
Without time synchronization
With time synchronization
Without time synchronization
Reference points = 2 Collect timestamps (GPS or NTP)
Reference points = 1 Infer TransD and PropD
Reference points = 1 Trigger Internal Timers
Reference points = 2 Collect timestamps (GPS or NTP)
Derive OWD from PHD
Derive OWD from TWD
Use per-hop One-Way Probe Packets (OWPP) to convey timestamps
Add timestamps to data packets
Using per hop One-Way Probe Packets (OWPP) with Ack in L2
Using data packets with Ack in L2
Using Two-Way Probe Packets (TWPP)
Using source-destination data packets with Ack in L3 or above
Use per-hop OWPP to convey timestamps
Add timestamps to data packets
Per-Hop Delay (PHD)
Two-way delay (TWD)
Reference points = 1 Infer TransD and PropD
Reference points = 1 Trigger Internal Timers
Using per hop OWPP with Ack in L2
Using TWPP
Using data packets with Ack in L2
Using source-destination data packets with Ack in L3 or above
Caption
Active Measurement
Passive Measurement
Based on other Measurement
Low overhead
High overhead

Figure 2.5: Taxonomy of delay measurement in IP networks

To measure PHD, when Global Positioning System (GPS) or Network Time Protocol (NTP) is available, nodes can be synchronized so that the hardware clocks of source and destination have the same reference time. In the case of using NTP, specific probe packets are used to transport the timestamps. The NTP is widely used in the Internet for clock synchronization and it provides an accuracy to the order of milliseconds over time scales of hours to days; however systematic errors can be verified as show in [13]. If GPS is used, the timestamps can be inserted directly in the data packets which enables to minimize the measurement overhead. Therefore, in this measurement method, a timestamp is collected in the initial reference point, this timestamp is transmitted to the next hop node, and since the next node is the final reference point, it collects another timestamp, and calculates the difference between the two collected timestamps to obtain the PHD. Two options are available to transport the first timestamp to the next node: to use a per hop One-Way Probe Packet (OWPP) that includes the timestamp, or add the timestamp to a data packet. The former uses a specific packet and the latter uses normal data packets which in general introduce less overhead in the measurement procedure.

GPS or NTP may not be available or may not work under some situations as indoor areas. In this case, the nodes are not synchronized and the PHD must be measured within each node, i.e. with initial and final reference points in the same reference point. This is accomplished by measuring the delays as presented in Eq. 2.4 by using internal timers, except for the TransD and the PropD (see Fig. 2.3) for which the delays must be inferred.

Fig. 2.6 presents a node and its next hop where PHD is to be obtained using per hop OWPP or data packets and using a L2 stop and wait Automatic Repeat-reQuest (ARQ) procedure, which consist in an automatic Acknowledge (ACK) in L2 when the packet is correctly received, which is commonly used in Wireless Local Area Networks (WLANs) and WSNs.

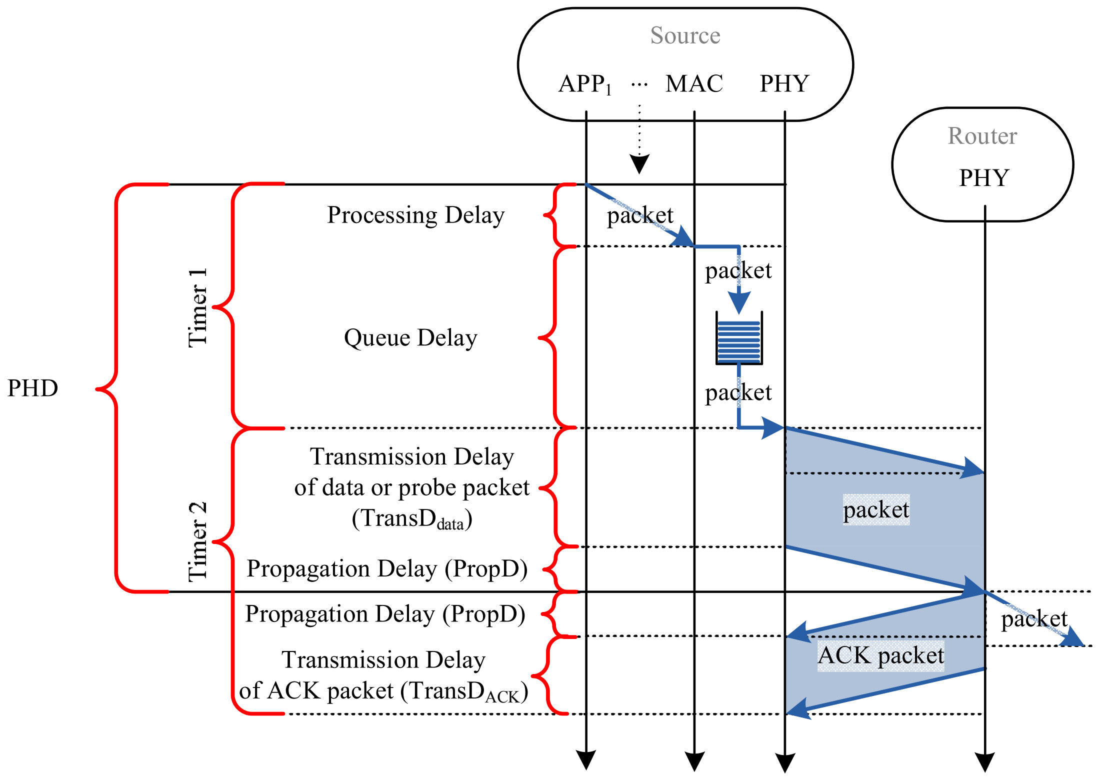

Source
APP$_1$ ··· MAC PHY
Router
PHY
PHD
Timer 1
Timer 2
Processing Delay
Queue Delay
Transmission Delay of data or probe packet (TransD$_{data}$)
Propagation Delay (PropD)
Propagation Delay (PropD)
Transmission Delay of ACK packet (TransD$_{ACK}$)
packet
packet
packet
packet
packet
ACK packet

Figure 2.6: Obtaining PHD without synchronization timer

From the Fig. 2.6, the PHD can be obtained using:

$$\text{PHD} = (\text{Result of Timer 1}) + \text{TransD}_{\text{data}} + \text{PropD} \tag{2.5}$$

where $\text{TransD}_{\text{data}}$ can be obtained using:

$$\text{TransD}_{\text{data}} = (\text{Result of Timer 2}) - (2 \times \text{PropD} + \text{TransD}_{\text{ACK}}) \tag{2.6}$$

Thus, PHD can be obtained as:

$$\text{PHD} = (\text{Result of Timer 1}) + (\text{Result of Timer 2}) - \text{PropD} - \text{TransD}_{\text{ACK}} \tag{2.7}$$

Assuming a specific physical media the PropD can be approximated using the Eq. 2.3, and the $\text{TransD}_{\text{ACK}}$ can be approximated using the Eq. 2.2. This procedure is possible if ACK is enabled in L2. An alternative approach to measure the TransD using probe packets in one reference point is named Packet Pair Flow Control (PPFC) and is proposed in [14]. PPFC is intended to be used in steady networks and it is based on the idea that if two probe packets are sent directly after each other, they are also queued one after the other and the time which lies between the end of the reception of the first packet and the start of the reception of the second packet, can be inferred as the transmission time.

When measuring TWD, both initial and final reference points are defined in the same node, i.e. there is a single reference point. Thus, time synchronization is not required and this type of delay is easily measured by triggering internal timers that account for the delay since the packet is sent, until an answer or echo is received. A Two-Way Probe Packet (TWPP), e.g. generated by ping or traceroute utilities, or any source to destination packet with ACK in L3 or above can be used to trigger the timers and collect the TWD.

In order to measure the OWD, the network nodes can be synchronized (using GPS or NTP) or not. If the nodes are synchronized, the timestamps are collected at two different reference points and OWD is measured. To transport the timestamp from one reference point to the other, OWPPs can be used or the timestamps can be added to normal data packets. If nodes are not synchronized, two options are available: to derive OWD from PHD[1] or to derive OWD from

[1]Network delay tomography operates in reverse, i.e. measures OWD and infers PHD. Further reading regarding network delay tomography can be found in [15, 16, 17, 18, 19].

TWD. If OWD is derived from PHD, Eq. 2.8 or Eq. 2.9 can be used.

$$\mathrm{OWD} = \mathrm{PHD} \times N \tag{2.8}$$

$$\mathrm{OWD} = \sum_{i=1}^{N} \mathrm{PHD}_i \tag{2.9}$$

where PHD_i is the PHD obtained in node i and N is the number of nodes up to the final reference point. The accuracy provided by the Eq. 2.8 to derive OWD from PHD is highly questionable since, in the real scenarios, individual PHDs may be different and even uncorrelated. Eq. 2.9 should provide more accurate results but individual PHD must be obtained in all the nodes transversed by packets.

Authors in [20] present a proposal to estimate OWDs in IP networks by conducting measurements of transmission, propagation and queuing delays in each node. In the context of wireless networks, in [21] the authors provide an analysis on the minimum delay in a WSN using unslotted mode of IEEE 802.15.4, taking into account the transmission related times, such as back off periods and inter-frame spacing. In [22] the authors propose an EED-based routing protocol for a WMN intended to to minimize EED accounting with queuing and transmission delays. In [9] the authors present a cross-layer mechanism to guarantee a defined EED for time sensitive applications that uses the IP-header option field to accumulate the PHD estimate that is used by a forwarder node to select an output priority queue.

In the case of the OWD being derived from TWD, Eq. 2.10 can be used.

$$\mathrm{OWD} = \frac{\mathrm{TWD}}{2} \tag{2.10}$$

In [23] authors provide a procedure to estimate TWD in multicast scenarios by using probe packets. Authors in [24] present a model focused on the prediction of the TWD obtained using statistical functions over previous measurements of TWD. In [25] the authors observed that RTT is a poor approximation of the OWD and proposed a scheme that analytically derives the OWD, forward and reverse delay for asymmetric networks. Also, the analysis made in [26, 27] reveals that the Internet paths have large delay asymmetries, raising doubts about the accuracy of this method when used with real traffic.

IETF Efforts to Measure Delay in IP networks

In order to provide common understanding regarding performance and reliability metrics that could be adopted in the Internet, Internet Engineering Task Force (IETF) has defined frameworks and methodologies to measure metrics in the Internet such as delay, bandwidth, throughput and packet loss. Fig. 2.7 presents the major IETF contributions in this area. They come mainly from IP Performance Metrics (IPPM) Working Group (WG).

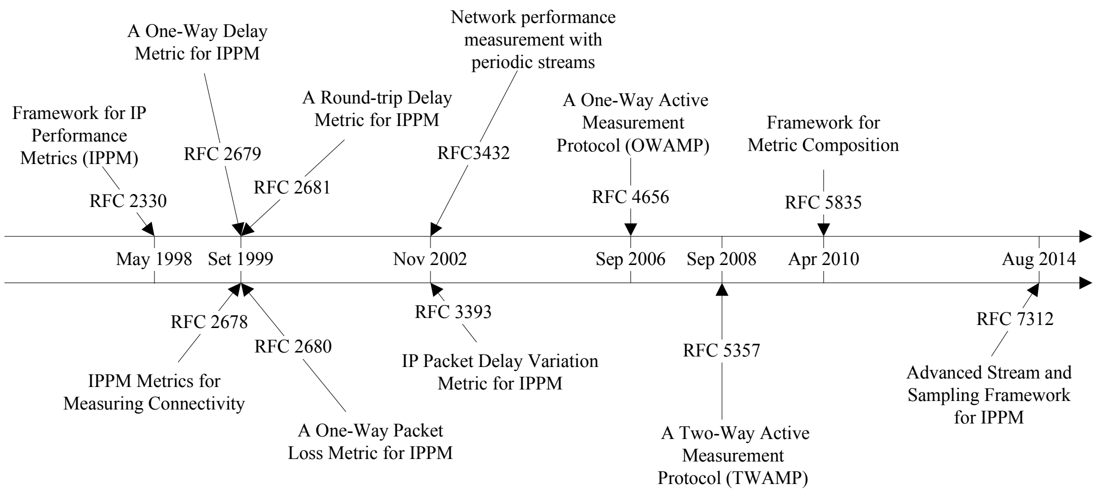

A One-Way Delay Metric for IPPM
Network performance measurement with periodic streams
Framework for IP Performance Metrics (IPPM)
RFC 2330
RFC 2679
A Round-trip Delay Metric for IPPM
RFC 2681
RFC3432
A One-Way Active Measurement Protocol (OWAMP)
RFC 4656
Framework for Metric Composition
RFC 5835
May 1998
Set 1999
Nov 2002
Sep 2006
Sep 2008
Apr 2010
Aug 2014
RFC 2678
IPPM Metrics for Measuring Connectivity
RFC 2680
A One-Way Packet Loss Metric for IPPM
RFC 3393
IP Packet Delay Variation Metric for IPPM
RFC 5357
A Two-Way Active Measurement Protocol (TWAMP)
RFC 7312
Advanced Stream and Sampling Framework for IPPM

Figure 2.7: IETF standards initiatives regarding performance metrics measurement in the Internet

In RFC 2330 [28] the IPPM WG defines a general framework and concepts to which performance metrics should comply to, and possible measurement methodologies. These metrics can be derived from other metrics that exhibit spatial, or temporal composition. The methodologies highlighted to measure these metrics fall in three categories: 1) direct measurement by injecting test traffic; 2) project end-to-end metrics from measured hop metrics; 3) estimate a metric from other sets of metrics.

The RFC 2678 [29] defines a set of metrics for connectivity between a pair of Internet nodes over a time interval and the RFC 2680 [30] defines metrics for one-way packet loss across Internet paths. The RFC 2679 [31] defines a metric for OWD and the RFC 2681 [32] defines a metric for TWD of packets in context of IPPM across Internet paths. Both contributions highlight that there are scenarios where OWD measurement should be performed instead of the RTD measurement since the path from a source to a destination can be different than the reverse path and even when two paths are symmetric, they may have different performance characteristics due to asymmetric queuing. Also, the performance of an application may depend mostly on the performance in one direction and in the use of QoS enabled networks, so provisioning in one direction maybe different than provisioning in the reverse direction, and thus the QoS guarantees differ.

The RFC 3393 [33] defines a metric for characterizing the variation of packets delays across Internet, which is based on the difference between OWD of selected packets. RFC 3432 [34] describes a periodic sampling method and relevant metrics for assessing the performance of IP networks using active and passive measurements and simulating applications generating Constant Bit Rate (CBR) traffic, typically multimedia applications.

RFC 4656 [35] presents the One-Way Active Measurement Protocol (OWAMP) for unidirectional metrics such as OWD and one-way loss using time sources such as GPS and Code Division Multiple Access (CDMA)-based systems (e.g. cellular networks) by using probe packets. The Center for Applied Internet Data Analysis (CAIDA) [36] uses OWAMP to measure one-way latency. Since OWAMP does not accommodate round-trip or two-way measurements, RFC 5357 [37] proposed the Two-Way Active Measurement Protocol (TWAMP) that can be used to measure one-way metrics in both directions between two network elements.

RFC 5835 [38] provides a framework for classes of metrics such as temporal aggregation, spatial aggregation, and spatial composition, that were described in the original IPPM framework (RFC 2330). RFC 7312 [39] updates the RFC 2330 with advanced considerations for measurement methodology and testing.

2.1.3 Delays Estimation

Two types of strategies can be used to estimate delays: offline and real-time. The offline strategy, takes advantage of theoretical models such as queuing theory or network calculus. The real-time strategy, estimates delays using samples of packets delays.

In an IP network, the sequence of packet delays measured can be described as a time series. The forecast methods based on time series use past data to estimate future data items and these methods are used in areas such as business planning or weather forecast [40, 41]. Therefore, real-time packet delay forecast or estimation comprehends two steps: 1) collect previous packet delays and 2) based on the previous packet delays provide an estimate for future packet delays. If the time series can be defined as stationary, i.e. no systematic change in key statistical moments such as the mean or variance, the following estimation methods can be used: naive, Moving Average (MA), Weighted Moving Average (WMA) and Exponential Weighted Moving Average (EWMA).

Assuming that D_n is the delay of the n-th sample, its estimate defined as $\widehat{D_n}$ can be obtained using naive method as follows:

$$\widehat{D_n} = D_{n-1} \tag{2.11}$$

Using the MA method, the $\widehat{D_n}$ is obtained as follows:

$$\widehat{D_n} = \frac{1}{N} \sum_{i=0}^{N-1} D_{n-1-i} \tag{2.12}$$

where N is the number of previous delays to consider.

Using the WMA method, the $\widehat{D_n}$ is obtained as follows:

$$\widehat{D_n} = \frac{1}{N} \sum_{i=0}^{N-1} (W_i \times D_{n-1-i}) \tag{2.13}$$

where:

W_i is the weight for the item i, where $\sum_{i=0}^{N-1} W_i = 1$

N is the number of previous delays to consider

Using the EWMA, the $\widehat{D_n}$ is obtained as follows:

$$\widehat{D_n} = \beta . D_{n-1} + (1-\beta) . \widehat{D_{n-1}} \tag{2.14}$$

where β is the smoothing factor and $0 < \beta < 1$

The naive method simply assumes that the next delay will be equal to the last one obtained, ignoring any historical data.

The MA method is often used as it is easy to understand and compute, but the estimation is only available when data series are equal or greater than N. The MA method also tracks the actual data but presents it with a lag and weights the data equally, which means it is not able to adequately represent data outliers (i.e. data points distant from other observations)[42].

In order to cope with some of this issues, WMA method can be used. WMA method counts differently the recent data and the other periods data in order to better adjust to the properties of the time series. Even though the result of WMA method also presents lag when compared to the time series and, similarly to the MA method, it is highly dependent on the value of N.

The EWMA method is easy to compute in real-time. Older data points never leave the estimation results but their impact is reduced for each new item in data series. Another feature regarding the EWMA is that it does not use many memory resources. While MA and WMA methods require the entire data set to be stored in memory, EWMA only needs the last estimation and the last sampled value. Due to these advantages many techniques use EWMA for forecasting. Authors in [43] propose an adaptive scheme based on EWMA to provide fault tolerant sensor networks and balance between traffic overhead and transmission failure, using multipath routing. In [44] authors propose a link quality monitoring mechanism where EWMA is employed to smooth the monitoring results. In [45] and [46] the authors propose prediction algorithms based on a moving average to be used in solar panels, and compare these methods with EWMA. In [47] the authors propose a link quality evaluation algorithm which employ EWMA method to adjust its sensitivity. Authors in [48] propose a prediction model to forecast the expected energy intake in a wireless sensor node, whose performance is compared with EWMA-based solutions. In [49] the authors propose the use of routing metrics accounting with average queuing and transmission delays obtained using EWMA. In [50] the authors use a data aggregation mechanism to reduce redundant packets by taking into account their current data smoothed by a EWMA function.

Although the methods described above can be used to forecast delay values based on the previous experienced delays, the estimate can be requested in a different network node from that in which the estimate was obtained. In this case, after providing a delay estimate, it is necessary to transport it to the nodes where the estimation is required. As example, when measuring OWD the second timestamp is the final reference point (see Fig. 2.4), and thus the delay estimate will be obtained in final node. If the source node is required to have this estimate it is necessary to transport the estimate up to the source node.

The Fig. 2.8 presents three methods that can be used to feedback delay information to where it is needed, here named as the estimation points. The terms active and passive feedback are used similarly to active and passive measurement, i.e. to indicate if the method influences more or less the traffic that is already transported in the network. The methods that use specific messages, e.g. using their own packets, have the undesired effect of introducing additional traffic, which in turn contributes to consume energy and processing resources, which is highly undesirable in WSN scenarios. In order to avoid extra traffic, the delay information can be conveyed in packets that are already transported in the network, i.e. conveyed in data plane messages or in control plane messages. In order to feedback delays using the data plane messages it is necessary that data packets are transported in reverse direction from source to destination node. In case of real-time applications the traffic can be highly asymmetric or not provide any data packets in reverse direction at all. The feedback can also be provided by using control plane messages, more particularly using the routing protocol messages. In [51] the authors survey routing metrics related to delays accounting that can be used in this context.

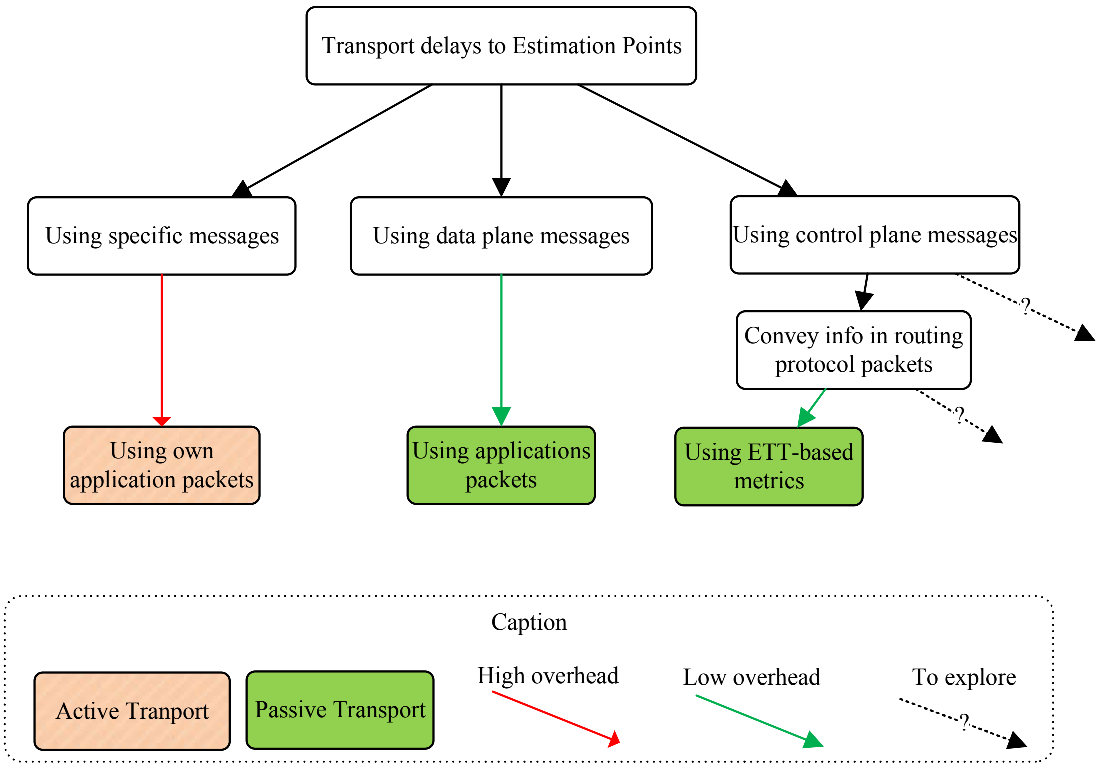

Transport delays to Estimation Points
Using specific messages
Using data plane messages
Using control plane messages
Convey info in routing protocol packets
?
?
Using own application packets
Using applications packets
Using ETT-based metrics
Caption
Active Tranport
Passive Transport
High overhead
Low overhead
To explore
?

Figure 2.8: Transporting delays to the estimation points

The research proposals that use routing protocol messages to feedback previous delays, use Expected Transmission Time (ETT) or ETT-related metrics, which can be derived from metrics such as the Expected Transmission Count (ETX) as shown in [52]. The ETT can be derived from ETX by using Eq. 2.15, where ETX is the expected number of transmission attempts required for successfully transmitting a packet, S is the packet size, and D is the data rate of the link.

$$\mathrm{ETT} = \mathrm{ETX} \times \frac{S}{D} \tag{2.15}$$

Authors in [53] provide studies to derive the OWD from the PHD or from the RTT. This work also used a wireless testbed to compare the performance of RTT and PPFC when used as link quality routing metrics against the performance of the ETX and hop count. The results provided showed that only ETX was able to outperform the hop count metric, whereas the two delay based metrics performed poorly. In [54] the authors propose a ETT-based metric intended to provide routing efficiency under various link conditions. In the proposed metric the MAC layer overheads are taken into account for calculating the data transmission time, instead of simply using packet/bandwidth. Authors claim the new metric outperforms the normal ETT metric in terms of network throughput and average packet delay. In [55] the authors present a novel ETT derived metric which takes into account the time between transmissions in each node in order to increase average network throughput in Wireless Mesh Networks. In [56] the ETT metric is adapted to improve the estimation of transmission time by including the actual load of different nodes. In [57] the authors introduce a new routing metric based on ETT metric to incorporate bandwidth adaptability in IEEE 802.11a networks.

After estimating or forecasting delays, and then obtaining the real delay values for those estimates, an accuracy evaluation could be conducted. The accuracy evaluation is based on the comparison of the estimated value with the real value. Different methods to evaluate the accuracy of an estimation have been proposed in literature. These methods can be [58] scale-dependent, based on percentage errors, or based on relative errors. The latter method is out of scope and thus, it will not be addressed.

In the scale-dependent errors, the result has the same scale of the data and the following examples can be depicted: Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE). For a real delay item D_i and its estimate $\widehat{D_i}$ (both expressed in seconds), the MAE, MSE and RMSE are obtained as:

$$\text{MAE (s)} = \frac{1}{N}\sum_{i=1}^{N}\left|\widehat{D_i} - D_i\right| \tag{2.16}$$

$$\text{MSE (s)} = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{D_i} - D_i\right)^2 \tag{2.17}$$

$$\text{RMSE (s)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{D_i} - D_i\right)^2} \tag{2.18}$$

where N is the number of data items in data series

The percentage errors methods have the advantage of being scale independent. Two of the most common methods are Mean Absolute Percentage Error (MAPE) and Symmetric MAPE (SMAPE) (or Adjusted MAPE). The percentage error between D_i and $\widehat{D_i}$ using MAPE can be obtained as:

$$\text{MAPE}\ (\%) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\widehat{D}_i - D_i\right|}{D_i}(\times 100) \tag{2.19}$$

where N is the number of data items in data series

MAPE compares the difference between $\widehat{D}_i$ and D_i with the D_i and thus, the results are expressed in a percentage from 0 to $+\infty$, and it implies a different error representation if the estimate is under or over the real value. In order to tackle this effect, SMAPE can be used. SMAPE compares the difference between $\widehat{D}$ and D with the mean of these two values, and it is obtained as:

$$\text{SMAPE}\ (\%) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\widehat{D}_i - D_i\right|}{(\widehat{D}_i + D_i)/2}(\times 100) \tag{2.20}$$

where N is the number of data items in data series

SMAPE has a lower bound of 0% and an upper bound of 200% and it intends to treat over and under estimations equally, avoiding distortion on the average value. Indeed, some contributions, e.g. [59], argue that SMAPE is not as symmetric as it may suggest, and variations of SMAPE should be used. In [58] and in [59] the methods above are compared and discussed around real scenarios.

2.1.4 Discussion

Section 2.1.1 started with a characterization of the delays components observed within an IP network node. QueueD is a component that depends on the network and traffic conditions; ProcD depends on the processing power available and on the stack implemented at each node; TransD and PropD are components that depend respectively on the transmission characteristics and physical media. In some research works, delays such as the ProcD and PropD are neglected since they are said to represent a small part of the total delay. However, ProcD should be accounted particularly in scenarios where nodes have limited processing resources as those employed in WSNs where ProcD can represent a relevant part of the total delay.

Section 2.1.2 has provided an overview on methodologies to measure delays in IP networks, and three types of delays were defined: PHD, OWD and TWD. These three types of delays can be measured using GPS or NTP. However, if these options are unavailable, the nodes should measure these delays using a single reference point, that is, use TWD to infer PHD or OWD. The OWD estimations inferred from TWD may not be credible when in the presence of asymmetric traffic. If OWD is derived from PHD, multiple delays components should be accounted. In WSNs that have limited processing resources, the processing delay can be highly relevant and should be accounted.

Finally, Section 2.1.3 has provided an overview on methods to estimate delays based on previous measurements. From the methods discussed, it was observed that EWMA considers the data series history and requires lower memory resources than MA or WMA. The delay estimation process is performed on a particular node but since the delay estimates may be required in other nodes, they need to be transported through the network. A possible solution is to use control plane messages, for instance using routing packets. In a WSN, the RPL can be used and adapted for that purpose.

2.2 Admission Control in IP Networks

In order to provide QoS to an application or flow in a network, a group of mechanisms such as admission control, resource reservation, scheduling, classification, policing, or shaping may be used. AC, in particular, can be used to control the amount of traffic entering in a network. AC mechanisms were used in the Public Switched Telephone Networks (PSTN). When a call is to be established, the AC mechanism is performed and decides if the call is accepted (resources available in the network), or if the call is rejected (no resources available in the network for the call). In these networks the main objective of the AC is to help determine if the network has enough resources for the incoming request.

In packet switched IP networks, the AC is also used to provide QoS through Integrated Services (Intserv) [60] and Differentiated Services (Diffserv) [61] architectures. Intserv is based on per-flow reservations in the network to provide per-flow QoS guarantees. This approach requires maintenance of individual flow states in the routers, and its signaling complexity grows with the number of flows; here the AC is used to decide if new flows are or not accepted. Diffserv relies on packet markers, policing functions at the edge routers, and different per-hop behaviors at core routers to provide QoS to aggregated traffic; here the AC may not be used, but its deployment is recommended to control real-time traffic at the ingress node [62] (for instance, for traffic classified with an Expedited Forwarding Per-Hop-Behavior). Intserv and Diffserv implementations in the IP global network are not used since the Internet is composed of multiple AS with different network administrations. Thus, these implementations are only applied in a set of IP networks under a unique administration.

The desired levels of QoS can also be provided using the over provisioning technique which consists in the deployment of enough resources to handle all the estimated offered traffic. Although over provisioning strategy maintains network simplicity, it does not provides the desired levels of QoS in scenarios such as network congestion or in scenarios with “greedy applications” (applications that consume always all the available network resources). In general, the over provisioning technique cannot provide any QoS guarantees.

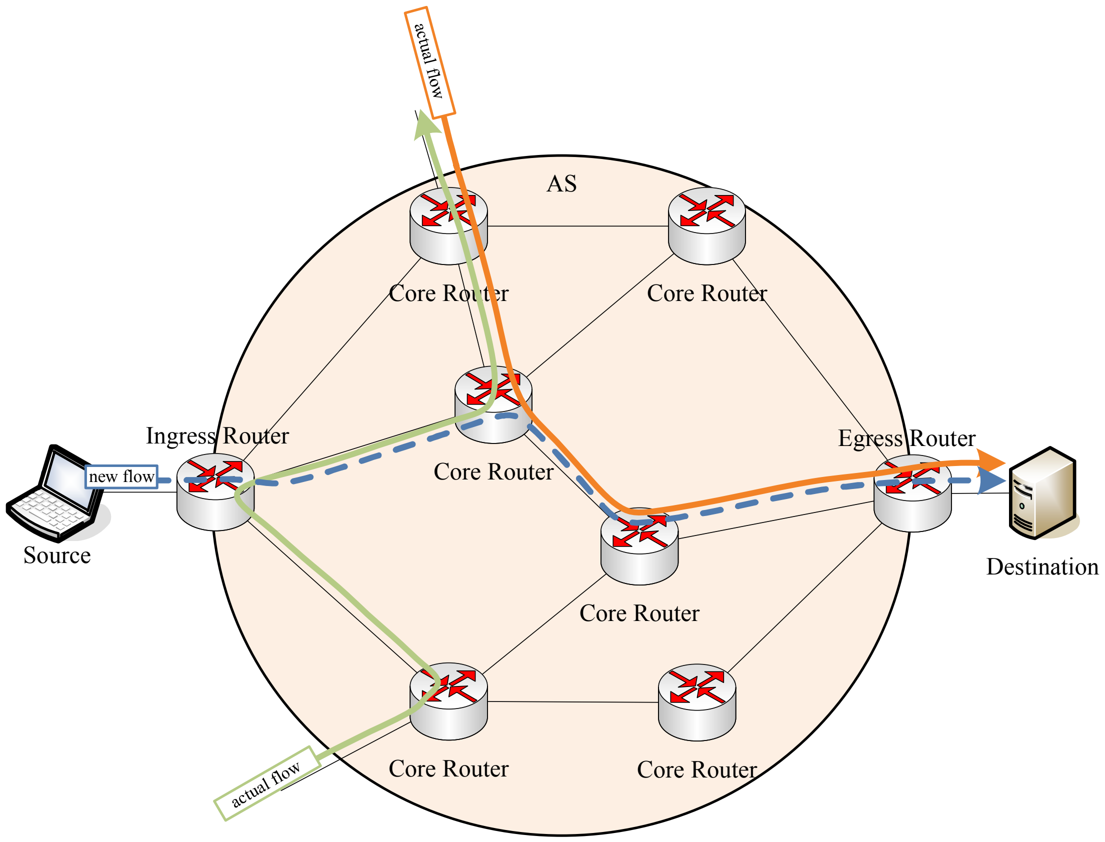


Figure 2.9: Network topology providing flow Admission Control

In Figure 2.9 is presented an example of a network topology where an AC mechanism is implemented within an AS area. Here, a source node generates a new flow intended to be delivered at a destination node. The new flow enters the AS area through an ingress router and in the AS exists one AC mechanism. This AC mechanism evaluates if the network can admit this new flow without affecting the level of service already assured to accepted flow(s). In order to make this decision, AC considers the traffic characteristics and the QoS requirements of the new flow and of the flows for the path to destination. This decision is valid through all the path from ingress router to the egress router; it is then communicated to the ingress router to admit or deny the new flow towards destination. Wrong decisions can be made by AC mechanism, i.e. to accept a flow without having enough resources to comply its or other flows' requirements, which is commonly named as a false positive, or to deny a flow that would have enough resources to be accepted, which is commonly named as a false negative.

A general and common criteria used to distinguish the wide range of AC proposals in the literature is given by the location where the AC decision is made. Two categories can be distinguished: the centralized AC which assumes a unique entity (e.g. one of the core routers of Fig. 2.9) that performs the AC decisions and exchanges signaling packets with the ingress nodes when new flows arrive; and the distributed AC which assumes that the decision is performed in multiple points within the network in a distributed manner (e.g. all the routers in Fig. 2.9 implement the AC mechanism).

The centralized AC mechanism assumes that the unique entity that has the complete and up-to-date knowledge of entire network topology and the usage of its resources. An example of a centralized AC mechanism is proposed in [63] where decisions are taken based on the measurement of the EED in a WMN. However, a centralized AC mechanism may not adequate for large and highly dynamic networks since the unique entity may have to process high volumes of information, what may imply bottlenecks, and in some cases, it may stands for a single point of failure.

The distributed AC mechanisms avoid the single point of failure and the scalability concerns of the centralized approach. However, since they have multiple AC decision points, they may not have the same view of resources occupancy and different decisions may be taken for flows competing for the same resources. These decisions may lead to violations of QoS and inefficiency of resources usage.

Considering the characteristics and limitations of the centralized and distributed AC mechanisms, the latter seems more adequate for the scenario of this thesis and thus, the focus of the next section will be directed to distributed AC mechanisms.

2.2.1 Distributed Admission Control

Figure 2.10 presents a classification for the different types of distributed AC mechanisms proposals in the literature. These proposals are organized in two groups according to their operation in the AS: the Edge-to-Edge and Hop-by-Hop operation. Similar classification can be found in [64].

In Edge-to-Edge operation only the ingress and egress nodes participate in the AC mechanism. The AC decision is taken on the egress node based on measurements and the decision

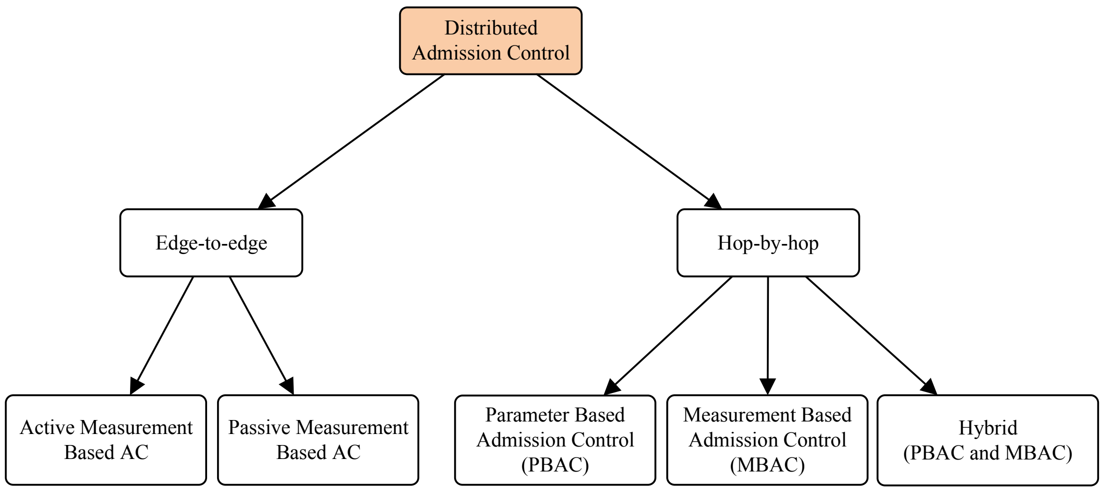

Distributed Admission Control
Edge-to-edge
Hop-by-hop
Active Measurement Based AC
Passive Measurement Based AC
Parameter Based Admission Control (PBAC)
Measurement Based Admission Control (MBAC)
Hybrid (PBAC and MBAC)

Figure 2.10: Admission Control mechanisms overview

is transported back to the ingress (see Fig 2.9). So, only these routers exchange control plane (signaling or measurement) packets and the decision taken is valid to the entire path from the ingress router to egress router. The advantage of Edge-to-Edge operation is that intermediate nodes (core routers in the Fig. 2.9) do not have to maintain any reservation state, since they are not participating in the AC mechanism. Two types of proposals can be found in this operation mode: the active measurement-based AC and the passive measurement-based AC proposals. In the active measurement-based proposals a probing flow is used to test entire path and to provide means for an AC decision on the egress node. In the passive measurement-based proposals, the QoS of the aggregate of accepted flows is continuously measured at the egress and used to provide an decision in the egress node.

In Hop-by-Hop operation all the nodes participate in the AC mechanism. Each of the routers in an AS (see Fig. 2.9) implement an independent AC mechanism that take a local decision about a new flow. The local decision is not valid for all the path from source to destination, and if a flow is accepted in a specific node it will progress to the next node and it will be evaluated again. Thus, the complete decision happens only in the last node (egress router) if the flow is accepted in all nodes up to this point. In case a router rejects a flow, this decision is propagated to the other nodes, ideally up to the ingress router in order to reject the flow prior is entrance in the AS area. Thus, in the Hop-by-Hop operation all the nodes of the path communicate with each other and each node has to maintain the state for the actual aggregated reservation. AC decisions are taken simultaneously in different nodes, which may lead to concurrency problems such as Thrashing [65]. The Trashing occurs when one flow is accepted in a node and the respective resource reservation is performed only in that node; if this flow is rejected later, other flows in the previous nodes may have been false rejected, since the resources were, in fact, available.

Using a Hop-by-Hop operation three types of proposals can be found in literature: the Parameter Based Admission Control (PBAC), Measurement Based Admission Control (MBAC), or a hybrid of both. PBAC (also known as Traffic Descriptor-based AC) proposals are based on the assumption that the traffic characteristics of the new flows are known prior to their establishment. There are no measurements and no resource estimation, and the traffic characteristics are conveyed by traffic descriptors as the only input for the AC mechanism to provide a decision. Also, in these proposals it is assumed that each node has a complete knowledge of currently admitted requests and current available network resources. The major disadvantage of these mechanisms is that it is difficult to have an accurate knowledge of each flow service request before it is established. In [66] the authors present a PBAC to provide hard QoS guarantees using a Diffserv architecture. In [67] the authors present a PBAC mechanism implemented in a peer-to-peer network for real-time video streaming applications; the decision of the AC mechanism is performed by a service provider, and is based on traffic descriptors that characterize the applications and their contract with service provider and the network resources.

MBAC proposals make the AC decision based on real-time measurements on the network. The AC mechanism attempts to capture the characteristics and requirements of flows admitted and bases its decisions on this knowledge. When compared with PBAC, the MBAC has the advantage to dispense the *a priori* knowledge about the flow characteristics and to predict characteristics of aggregate flows is usually easier that to predict in a per-flow basis. The major disadvantage of MBAC is that its decision to accept or deny a flow depends on measurements which have always associated errors that could lead to false negatives or false positives. In [68] the authors propose a MBAC that uses only measurements of aggregate bandwidth and does not need to keep the flow state in each node. Authors in [69] present an MBAC mechanism for WSNs based on direct measures of packet loss ratio, inter-arrival jitter and throughput, to be used by real-time applications; the authors estimate these performance parameters by using probing packets. In [70] the authors implemented two AC mechanisms, one using on PBAC and the other using MBAC and they evaluated the efficiency of their network utilization. When tested bursty traffic patterns, the authors concluded that MBAC provided a more efficient network utilization than PBAC. In [71] authors proposed a analytical model for node delay distribution in IEEE 802.11 wireless networks and developed an admission control mechanism scheme for traffic with stochastic QoS guarantees to be applied in a source node.

The hybrid proposals basically have been developed to address the problems highlighted with PBAC and MBAC approaches. These hybrid proposals use both knowledge of submitted traffic descriptors and measurements taken from the network to predict future service levels required by a flow. In [72] authors propose a hybrid approach using bandwidth measurements that, when compared to PBAC and MBAC proposals, provides better network utilization efficiency. In [73] the authors present a hybrid proposal that directly estimates effective bandwidth from available traces, and use these estimates in conjunction with peak rate values (given by a traffic descriptor) to take a AC decision about a new flow. In [74] authors propose an hybrid admission control mechanism for real-time traffic that takes both delay and reliability into account, and a fairness-aware rate control algorithm for non-real-time traffic, both to use in WSNs and tested in IEEE 802.11. The admission control mechanism is deployed in the source node and the delay estimation is not addressed.

2.2.2 On the Implementation of Admission Control

The type of AC mechanism to implement can also be defined according to the QoS guarantees requested by the application flows that will be transported in the network. The flows requesting a defined service level to an AC mechanism can have very diverse QoS requirements in terms of data rates, delay bounds, or maximum loss ratios. Particularly regarding delay, the authors in [75] provide an helpful study to define different types of QoS guarantees. Critical control applications demand some degree of reliability and timely delivery of control commands, thus, they require deterministic or hard QoS guarantees. Multimedia applications can tolerate some degree of QoS violation, so probabilistic or soft QoS guarantees should be provided [76][77]. These type of constraints can be defined as a Time Utility Function (TUF) [75] as shown in Fig. 2.11, where EED_p is the time elapsed by packet p since its generation at the application in the source node, until it arrives at application in the destination node.

According to [78], the PBAC mechanisms are used to provide hard real-time services that are based on worst case bounds derived from the parameters describing the flow; these algorithms typically result in low network utilization in the face of bursty network traffic. The MBAC mechanisms can use less stringent admission control algorithms and thus, they are used to provide soft real-time services. More generally, the type of AC mechanism should always be adequate to network and applications specifications, and also to the trade-off between network resource utilization and the conflicting requirement to maintain the QoS of current flows. Although commonly the AC mechanisms take decision in a per-flow basis, other granularities can be defined for the unit which is the target of an AC mechanism decision such as per-packet,

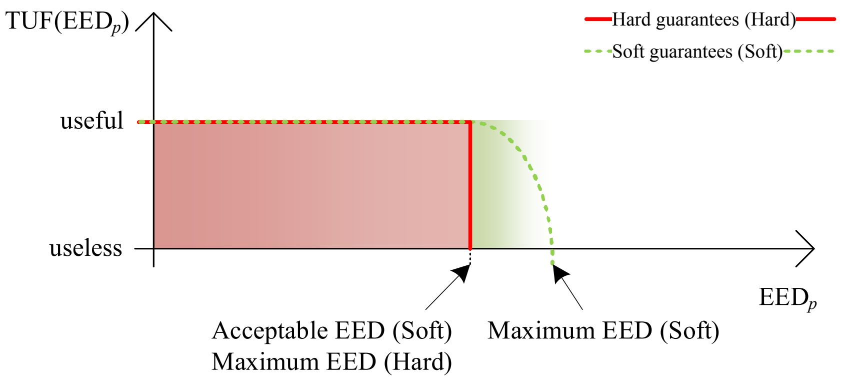


Figure 2.11: Time Utility Function for an application requiring soft QoS guarantees

per-TCP-session, or per-user AC [79]. In [80] an example of a per-packet admission control where its decision is based on resource tokens instead of bandwidth measurements.

The operation of single traditional IP networks are based in First-In-First-Out (FIFO) queues with tail drop, which means that the implementation of AC mechanisms could turn the network administration and operation more complex and costly. Thus, the deployment of AC has to be performed maintaining network simple and efficient. In particular, the deployment of an AC mechanism in a wireless network such as a WSN implies tackling specific challenges: when compared to structured networks, the wireless networks usually has less usable spectrum, less reliability, and typical wireless medium phenomena such as interference or multipath fading. Thus, the effects of a congestion can be more severe in these type of networks and an AC mechanism may be helpful. Nevertheless, the AC mechanism must also be carefully designed in performance and efficiency in order to cope to these networks' limited energy and communications resources.

2.2.3 Discussion

This section provided an overview on the state-of-art regarding the AC with a special focus on the actual distributed AC proposals in the literature. Also, considerations regarding the implementation of the AC mechanisms are provided.

When compared to the centralized AC mechanisms, the distributed AC mechanisms are more adequate to large and highly dynamic networks. Focusing on the actual distributed AC proposals it can be remarked that they perform per flow decisions and most of them imply sending extra control plane messages to distribute their decisions. Thus, they are not optimized to the constraints of a WSN where nodes use IEEE 802.15.4 standard and have limited energy resources. Also, in order to prevent processing useless packets as soon as possible, the admission control should be deployed in source and forwarding nodes and operating in a cross layer approach.

2.3 WSN Operation and Constraints

In a WSN, sensor nodes have limited energy, processing and communications resources. Energy constraints influence the hardware and software they use for operation. This section describes hardware, Operating System (OS), and standards and protocols, from the point of view of key constraints and behavior.

2.3.1 Hardware and Operating Systems

The WSNs can be deployed in a range of hardware products including the Seed-Eye [81], the WiSMote [82], the Z1 [83], the MICAz [84], the Telos [85, 86], and the Tmote Sky [87]. All these hardware platforms are compatible with IEEE 802.15.4 standard. This overview is focused on the Tmote Sky [88] platform which is the successor of the Telos motes[2].

The Tmote Sky [88] comprises a TI MSP430F1611 MicroController Unit (MCU), a TI CC2420 radio chip, 10 *k*Bytes of RAM, 48 *k*Bytes of ROM (flash), and an external flash of 1024 *k*Bytes. The MSP430 MCU family used in these motes is announced for their ultra low power consumption. For instance, the MSP430F1611 [90] is a 16 bit MCU which is announced to have a power consumption of 330 μA (at 1 *M*Hz using 2.2 V), 1.1 μA in standby mode and 0.2 μA in Off Mode (RAM retention). Also, according to [91], the MSP430F1611 MCU processor runs at 8 *M*Hz, and when using 3 V as supply voltage it executes 0.33 Million Instructions Per Second (MIPS) which is much less than the 1186 MIPS per core executed by a Raspberry Pi [92] (according to [93]). Tmote Sky uses the CC2420 [94] radio chip which is compliant with the IEEE 802.15.4 standard. This radio chip is designed for low power and low voltage wireless applications and has power consumptions of 18.8 *m*A when receiving, and of 17.4 *m*A when transmitting. It operates at 2.4 *G*Hz, and achieves data rates of 250 *k*bit/s. When CC2420 is installed in a Tmote Sky, an integrated onboard antenna enables an indoor range of approximately 50 m and 125 m for outdoors, according to [88]. When comparing

[2]The Telos motes are available in two versions: the Telos Revision A (or Telos RevA) [85] and the Telos Revision B (also known as Telos RevB or TelosB) [86]. The RevA comprises a MSP430F149 MCU, a TI CC2420 radio chip, 2 *k*Bytes of RAM, and 60 *k*Bytes of flash. The RevB comprises the TI MSP430F1611, the same radio chip of RevA, 10 *k*Bytes of RAM, and 48 *k*Bytes of ROM (flash). Both Telos RevA and RevB include sensors for light, temperature, and humidity. Other minor differences between Telos RevA and RevB are detailed in [89].

CC2420 to CC3200 (single-chip MCU with built in Wi-Fi connectivity) pointed as an IoT solution, the latter enable data rates of 54 *M*bit/s when in mode 802.11g and 72 *M*bit/s when in mode 802.11n. These limitations in processing and communication capabilities are relevant when evaluating the EED in a sensor node. The reduced processing power, increases the time to execute programming routines, i.e. the ProcD, and the reduced data rate of the radio chip increases the time to transmit a packet, i.e. the TransD.

Multiple OSs specific for the WSNs are available. Examples of open source are the Tiny OS [95], the Contiki OS [96, 97], the RIOT OS [98, 99, 100], and the Lite OS [101]. From these, an overview on Contiki OS is provided. The Contiki OS has been developed at the Swedish Institute of Computer Science (SICS) and counts with a wide number of developers that provide support, new features and fixes. It was the first operating system for wireless sensor nodes to implement the micro IP (uIP) stack [102, 103, 104], and, in 2008, it also incorporated the uIPv6 [105]. Contiki uses the protothread programming abstraction [106] and both the Contiki OS and its applications are implemented in the C programming language. Contiki OS has been ported to multiple microcontroller architectures, including the Texas Instruments MSP430 and the Atmel AVR. According to [87], the typical contiki OS memory footprint using full IPv6 networking and RPL requires 10 *k*Bytes of RAM and 30 *k*Bytes of ROM, which represents 100% of available RAM and 62.5% of available ROM of Tmote Sky sensor node.

2.3.2 IEEE 802.15.4 Standard - Physical and MAC Layers

WSNs can be deployed using different physical and data link layer standards such as IEEE 802.15.4 or IEEE 802.11[3]. Since WSNs are usually limited in transmission and energy resources, they are deployed as a Personal Area Network (PAN), i.e, a computer network organized around a personal area, using short range communications to interconnect with other devices. The IEEE 802.15 Working Group develops standards for the Wireless Personal Area Network (WPAN) and namely the IEEE 802.15.4 standard defines both physical and data link layers in the context of a Low-rate Wireless Personal Area Network (LR-WPAN). A LR-WPAN is a WPAN that is characterized by using low-cost devices, with low data rates that have a low-power operation. The IEEE 802.15.4 standard has been developed since 2003, with relevant updates in 2006 [110] and in 2011 [111].

[3]Recent advances in IEEE 802.11 based standards, namely in the IEEE 802.11ah [107] standard, are being developed to adapt 802.11 standard to the IoT concept and to the WSN requirements assuming low power and wide range sensor nodes. The IEEE 802.11ah Draft4.0 was released in February 2015 and the IEEE 802.11ah Draft5.0 was released in April 2015. Further reading providing an overview on 802.11ah can be found in [108] and [109].

The IEEE 802.15.4 standard defines the networks as PANs, where each PAN is composed of one coordinator and one or more members. The PANs can be interconnected, in this case the PAN coordinator will also be a member of another PAN. The packets may use a 16 bit PAN identifier in order to identify its own PAN and the destination PAN.

The standard specifies two types of devices: the Full Function Device (FFD), that can communicate with every other node and support the full protocol, and the Reduced Function Device (RFD) that can only communicate with the FFDs. A PAN coordinator must first be defined from the list of FFDs devices present in the PAN. Each node will use two address: a long address (64 bit length), the global Identifier (ID); a short address (16 bit length), the PAN specific address that is assigned by the PAN coordinator when the device joins the PAN.

Regarding the physical layer, the IEEE 802.15.4 standard defines three, license-free, frequency bands using Direct-Sequence Spread Spectrum (DSSS) modulation, each one containing a set of channels as follows [112, 110, 111]:

- 868.0-868.6 MHz: usable in Europe with one channel using Binary Phase-Shift Keying (BPSK) and providing a bit rate of 20 $k$bit/s;
- 902-928 MHz: usable in North America initially with up to ten channels, later extended to thirty, using 2 MHz of channel spacing, BPSK and providing a bit rate of 40 $k$bit/s;
- 2400-2483.5 MHz: usable Worldwide with up to sixteen channels using 5 MHz of channel spacing, Quadrature Phase-Shift Keying (QPSK) and providing a bit rate of 250 $k$bit/s.

The latter frequency band is shared with the IEEE 802.11 radio frequency at the 2.4 GHz band and significant interference is expected.

In the MAC layer, two modes of operation are defined in the IEEE 802.15.4 standard: the beacon-enabled mode and the beacon-less mode. When in the beacon-enabled mode, the PAN coordinator assigns time slots to each receiver and enforces a transmission schedule using explicit beacon messages. The access to the channel is slotted and, thus, this mode enables devices to consume less power because the receivers can be switched off. In this mode, the PAN coordinator broadcasts a periodic beacon message containing information about the PAN. The period between two consecutive beacons is defined as a *superframe*, which is divided into an active and an inactive part, during which the coordinator may enter power saving mode.

In the beacon-less mode no beacon messages are transmitted by the coordinator and the receivers must be listening all the time. This mode uses more battery, but it is easier to configure. In this mode, if the node wants to send a frame it checks if the channel is free, and, if so, it sends the frame. If the channel is busy the node waits for a random period of time before trying to access the channel again. Medium access control is performed with Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) that may be combined with Clear Channel Assessment (CCA), depending on the selected mode. The available access control modes are threefold: 1) beacon-less mode with unslotted CSMA/CA 2) beacon-enabled mode with slotted CSMA/CA 3) beacon-enabled mode with slotted CSMA/CA integrated with Guaranteed Time Slot (GTS).

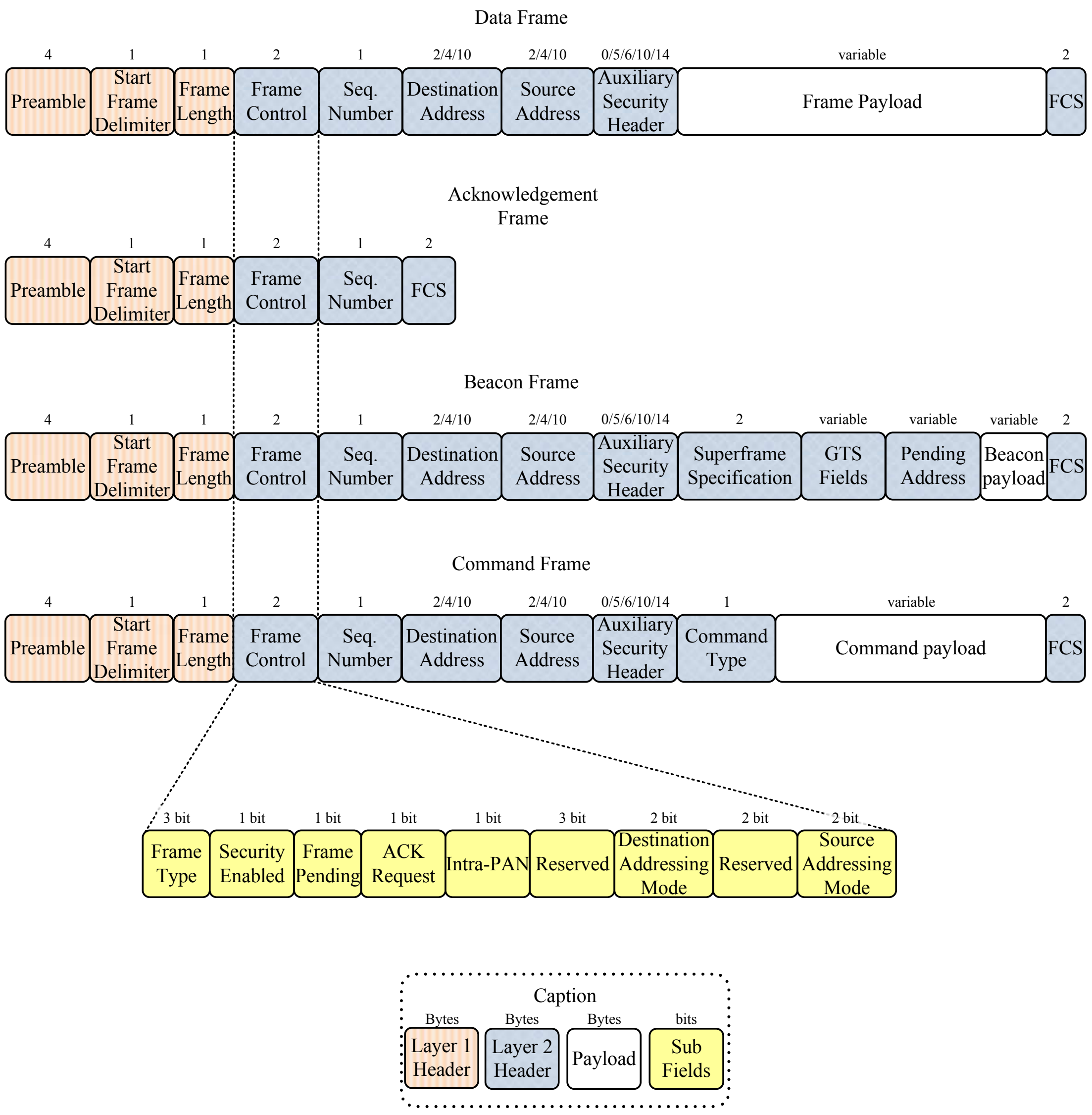


Figure 2.12: Frames structures defined in the IEEE 802.15.4 standard

The IEEE 802.15.4 standard defines four types of frames, shown in Fig. 2.12. The Data frame is the one that conveys the data payload. The Acknowledgment frame is used to acknowledge the correct reception of another frame. The Beacon frame is used by the coordinator to transmit the beacons messages and organize the PAN. The Command frame is used for association, disassociation, data and beacon requests, conflict notification, among others. Further information about frame types and formats are available in IEEE 802.15.4 standard [112, 110, 111].

2.3.3 IP-based stacks

Fig 2.13 shows the standard TCP/IP stack and the lightweight IP (lwIP) [113] stack implementations. The standard TCP/IP the stack implementation usually includes multiple options for layer 1 and 2, can use both IPv4 and IPv6, supports protocols such as the Internet Control Message Protocol (ICMP), and implements both the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) transport modes. lwIP is an open source and lightweight implementation of the TCP/IP stack, developed to be used in resource constrained equipment. Regardless of its low memory and code footprints, it still implements all the functions of the main protocols used in a typical TCP/IP stack, such as the Address Resolution Protocol (ARP) the Point-to-Point Protocol (PPP), the ICMP, the Internet Group Management Protocol (IGMP) the IP, the UDP, the TCP, the Domain Name Service (DNS), the Simple Network Management Protocol (SNMP) and the Dynamic Host Control Protocol (DHCP) protocols. A lwIP installation typically requires about 20 kBytes of ROM and 40 kBytes of RAM. An implementation of lwIP over the Ethernet II standard is described in [114].

Fig 2.14 shows the uIP and the uIPv6 stack implementations. The uIP stack [102, 103] implements the main protocols in TCP/IP, using IPv4, and was developed for embedded systems with even more restricted ROM and RAM specifications than those targeted by the lwIP stack. uIP implementation includes the ARP, the ICMP, the IP, the UDP and the TCP protocols. With the advent of the IoT, where multiple sensors are meant to communicate with each other while also being connected to the Internet, the IPv6 was considered as crucial. The uIPv6 [105] stack was developed with this scenario in mind. The uIPv6 stack implementation reduced the TCP/IP stack to only implement the essential IPv6 protocols. It is claimed to be the world's smallest IPv6 stack and it implements the IPv6, the ICMPv6, the Neighbor Discovery Protocol (NDP), the TCP, and the UDP protocols.

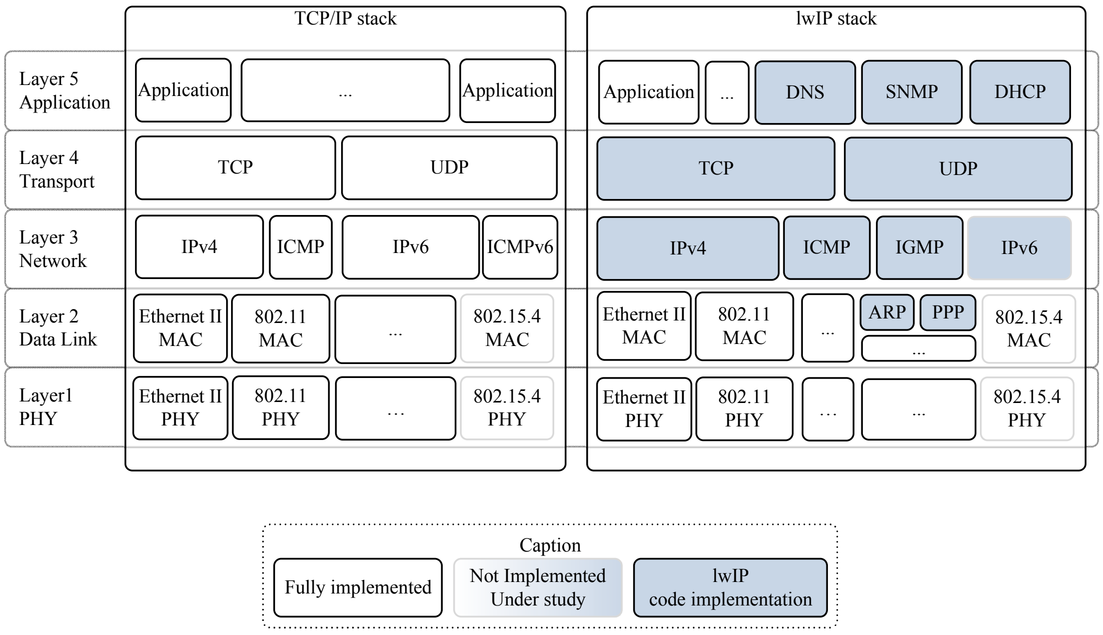


Figure 2.13: TCP/IP and lwIP stacks

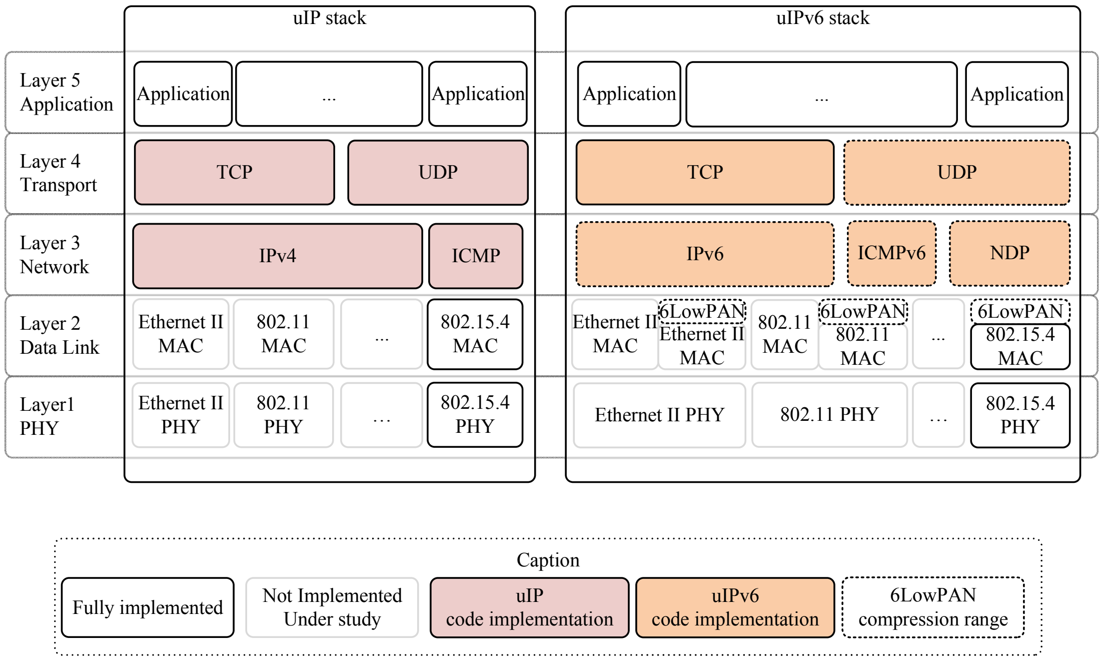


Figure 2.14: uIP and uIPv6 stacks

Table 2.1 compares the discussed IP-based stacks implementations in terms of release date, code size and of required RAM. These stacks, particularly the uIPv6, are extending their support for new features while still being able to meet the requirements of new low cost motes. Research efforts have been made to extend the operation of the uIPv6 to other layer 2 technologies. For instance, in [115] the uIPv6 stack is implemented over IEEE 802.15.4, IEEE 802.11 and IEEE 802.3 PHY and MAC layers. In [116] the uIPv6 stack is implemented over IEEE 802.11 PHY and MAC layers.

Table 2.1: IP-based stacks comparison

	Release date	**Code ROM size (*k*Bytes)**	**Required RAM (*k*Bytes)**
lwIP	2000	40	20
uIP	2001	4	1
uIPv6	2008	11.5	2

The uIPv6 implementation may not cope with the IPv6 functionality required in the current platforms. The IEEE 802.15.4 standard can only handle a Maximum Transmission Unit (MTU) of 127 Bytes, forcing nodes to fragment and encapsulate their IPv6 PDUs or IPv6 packets, which can have up to 1280 Bytes, into the IEEE 802.15.4 small frames. The RFC 2460 [117] defines that the IPv6 requires that every link must have a MTU of 1280 Bytes or greater. If the link cannot convey packets with 1280 Bytes in length in one piece, it must provide link-specific fragmentation and reassembly (below IPv6). Fig. 2.15 compares the protocol overhead while using IPv6 over IEEE 802.15.4, spanning from the best case scenario to the worst case scenario. In the best case scenario, the use of protocol headers is lower and these leave 70 Bytes available to be used to convey the data payload. In contrast, with higher protocol overheads the space available for data payload is only 28 Bytes. According to experimental evaluation in [118], if the CBC_MAC_16 security mode is enabled, the Auxiliary Security Header (ASH) and Message Integrity Code (MIC) fields must be inserted, thus leaving only 2 Bytes for the payload. A detailed analysis about MAC security overhead in the IEEE 802.15.4 standard can be found in [119].

This leaded IETF to form the IPv6 over Low power WPAN Working Group that later proposed IPv6 over Low Power Wireless Personal Network (6LowPAN) [120]. 6LowPAN is an adaptation layer that enables the transport of IPv6 packets over IEEE 802.15.4 links by performing packet fragmentation and reassembly. Packets larger than the IEEE 802.15.4 frame payload are fragmented at the source, and reassembled at the destination. 6LowPAN also

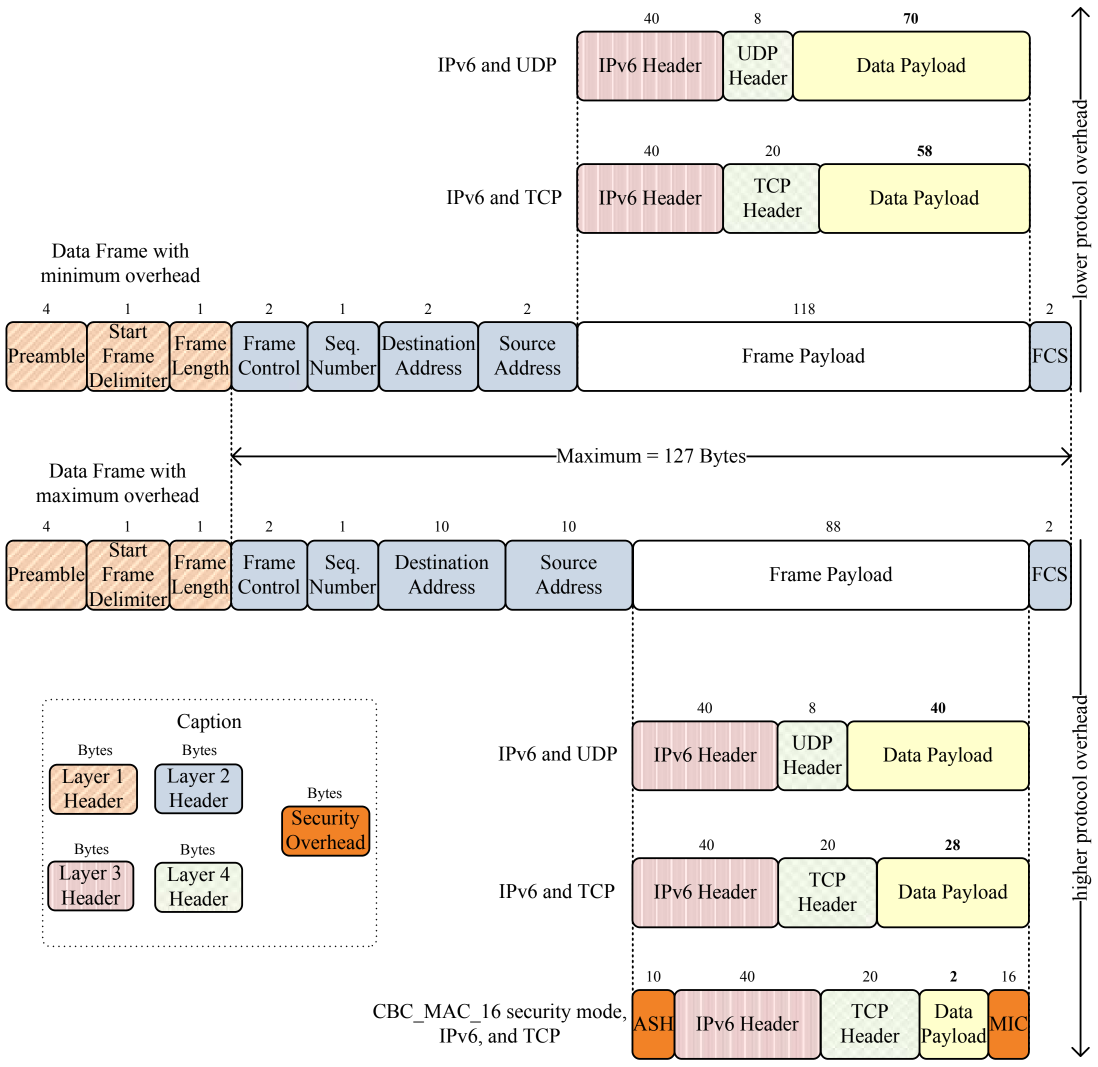


Figure 2.15: Protocol overheads in IEEE 802.15.4 standard

performs header compression. It compresses the IPv6, the ICMP and the UDP headers (see Fig. 2.14). New fields were added, namely the Dispatch Code field, the Header Compression (HC)1 which used to convey compressed IPv6 header, and the HC2 which is used to convey the compressed UDP header, among others. The compression provided by 6LowPAN is stateless, not requiring nodes to maintain compression related state information. RFC 4919 [121] provides an overview and presents the problem statement, while the base specification is in RFC 4944 [120], later updated by the RFC 6282 [122] and by the RFC 6775 [123]. Although 6LowPAN was initially intended to be used only with the IPv6, in [124] a proposal for its use with IPv4 is described. The project SICSlowpan [125] provides an implementation of 6LowPAN for the Contiki OS.

6LowPAN is implemented in Tiny OS and Contiki OS. The implementation of IPv6 and 6LowPAN in the Tiny OS is provided by Blip [126]. In the Contiki OS, the implementation of 6LowPAN is known as SICSlowpan provided by the SICSlowpan Project [125].

2.3.4 RPL Routing Protocol

Routing is one of the functions of the network layer. RPL standard is defined in the RFC 6550 [127] and it consists of a routing protocol that was designed for Low-power and Lossy Networks (LLNs) such as the WSNs. RPL is defined by the IETF Routing Over Low-power and Lossy networks (ROLL) and it is developed for devices with limited processing, memory and energy resources. Multiple instances of RPL can be run in a single network topology, where the nodes are organized in distinct tree topologies named Destination-Oriented Directed Acyclic Graph (DODAG), or simply Directed Acyclic Graph (DAG). Each DAG has a DAG root, the node where all paths terminate.

Within a given DAG, the Objective Function (OF) defines how to the metrics/constraints are converted into a rank value, i.e. a value representing the distance/cost to the DAG root. The OF also defines how a node selects its Preferred Parent (PP) from a set of Candidate Parents (CPs). ROLL defined two OFs: Objective Function Zero (OF0) in RFC 6552 [128], and Minimum Rank with Hysteresis Objective Function (MRHOF) in RFC 6719 [129]. When using the OF0, the node will always select the parent with the lowest rank. When using the MRHOF, the node, in the parent selection process, considers the lowest rank combined with a hysteresis value. Fig. 2.16 depicts how the best parent selection is made when using MRHOF. Fig. 2.16 assumes that a best parent $p1$ already exists, and that $p1$ and another CP, $p2$, advertise their ranks to a specific node. This node will select $p2$ only if its rank is lower than the rank of $p1$ by at least a given hysteresis value.

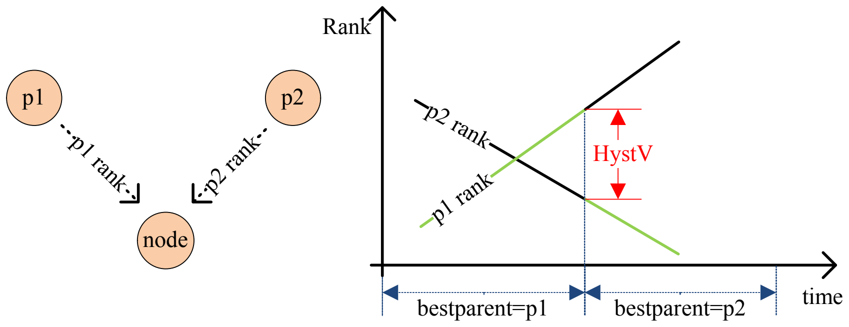


Figure 2.16: Best parent decision using MRHOF in RPL

RFC 6550 [127] defines multiple control messages to create and maintain routing information in each node: DAG Information Solicitation (DIS), DAG Information Object (DIO), Destination Advertisement Object (DAO), DAO-ACK, secure versions of the previous messages and a Consistency Check message.

The format of the DIS message is shown in Fig. 2.17. The Flags and Reserved field of the DIS messages are always initialized to zero by the sender and ignored by the receiver. The Options field is used to transport multiple options inside DIS and is common to DIS, DIO, DAO and DAO-ACK messages.

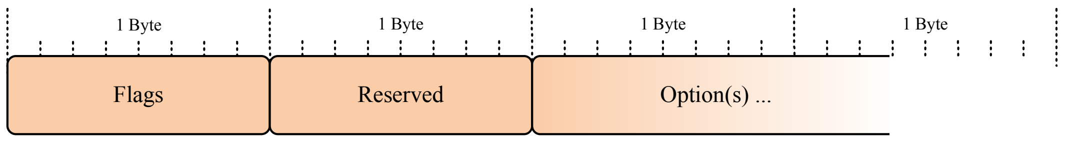


Figure 2.17: DIS message format

The format of the DIO message is shown in Fig. 2.18. The RPL Instance ID field indicates the ID of the RPL Instance. The Version Number field contains a sequential counter that is incremented by the DAG root to form a new Version of a DAG. The Rank field indicates the DAG rank of the node sending the DIO message. The Grounded (G) field contains a flag that indicates whether the advertised DAG can satisfy the goal defined by the application. The Mode of Operation (MOP) field identifies the mode of operation of the RPL Instance. All nodes who join the DAG must be have the same MOP which is encoded using the following values:

- 0: No Downward routes maintained by RPL
- 1: Non-Storing MOP
- 2: Storing MOP with no multicast support
- 3: Storing MOP with multicast support

The DAG Preference (PRF) field defines how preferable the root of this DAG is compared when to other roots within the same instance. The Destination Advertisement Trigger Sequence Number (DTSN) field is used as part of the procedure to maintain the downward routes. The Flags and Reserved fields are always initialized to zero by the sender and ignored by the receiver. The DAG ID field conveys the IPv6 address of the DAG root. The Options field is used to transport multiple options inside DIS and is common to DIS, DIO, DAO and DAO-ACK messages.

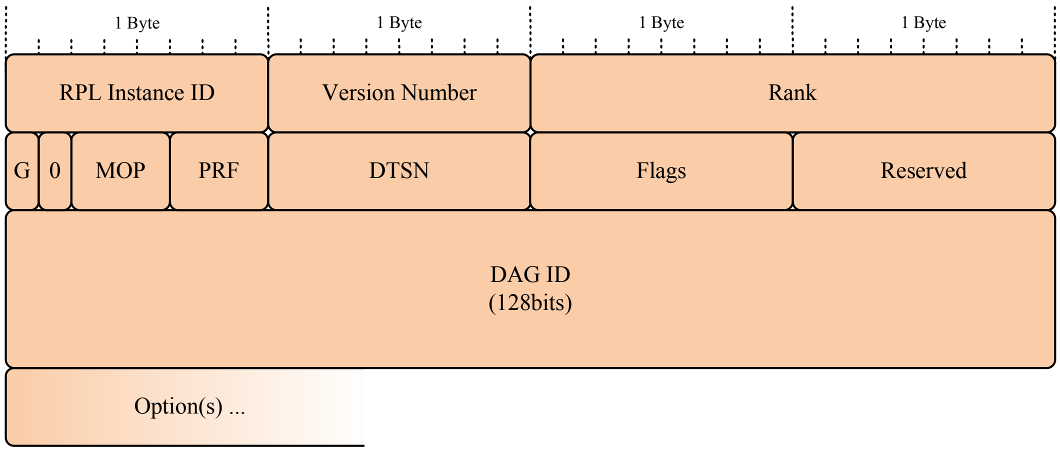


Figure 2.18: DIO message format

The format of the DAO message is shown in Fig. 2.19. The RPL Instance ID field indicates the ID of the RPL Instance. The *K* flag indicates if the recipient is expected to send back a DAO-ACK. The *D* field indicates whether the DAG ID field is present or not. The DAO Sequence field is incremented whenever a DAO message is transmitted by a node and repeated in the DAO-ACK message.

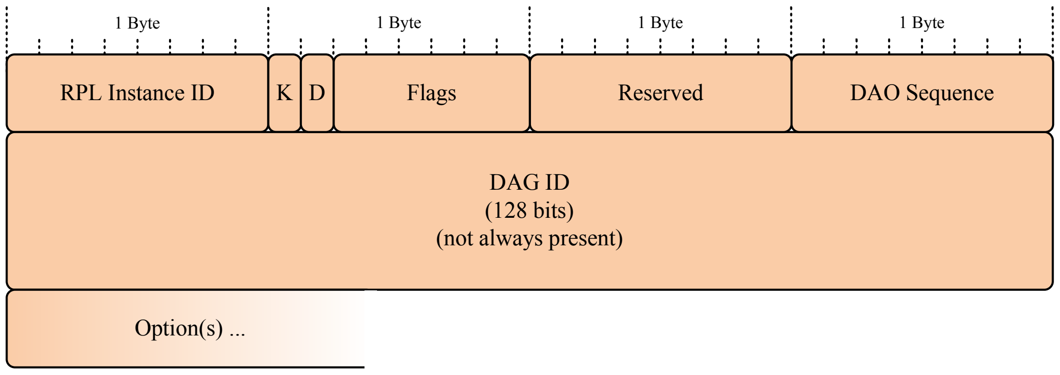


Figure 2.19: DAO message format

The format of the DAO-ACK message is shown in Fig. 2.20. The RPL Instance ID field transports the ID of the RPL Instance. The *D* flag indicates if the DAG ID field is present. The Flags field are always initialized to zero by the sender and ignored by the receiver. The DAO Sequence is incremented at each unique DAO message from a node and repeated in the DAO-ACK message. The Status conveys information about the role that the parent is willing to accept. The DAG ID conveys the IPv6 address of the DAG root (when flag *D* is set).

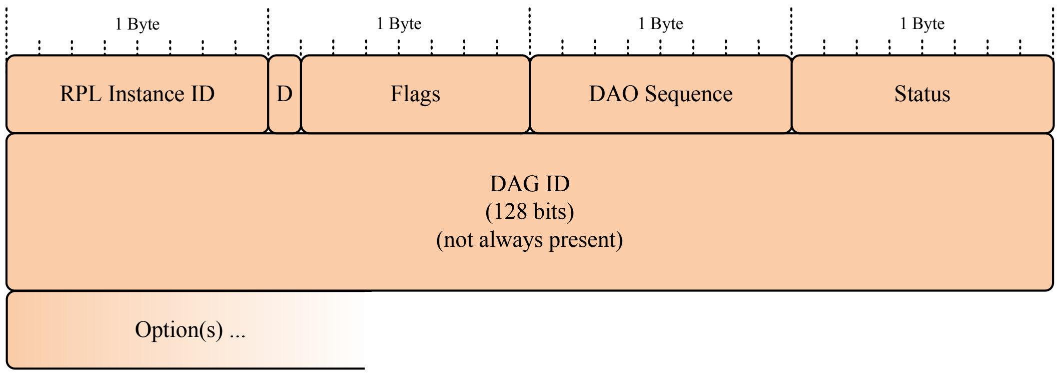


Figure 2.20: DAO-ACK message format

The secure versions of DIS, DIO and DAO messages have the same structure as the ones presented above and the Consistency Check message is used to check secure message counters and to issue challenge-responses.

Fig. 2.21 shows an example of an exchange of RPL control messages between nodes in order to form a DAG (for simplicity, the DAO-ACK is not shown in this figure). The DAG root multicasts a DIO message comprising the RPL instance and the DAG configuration parameters. This allows other nodes to join the DAG, to select a parent and to participate in the DAG. A node, instead of waiting for a DIO message in order to join an already formed DAG, can multicast a DIS message requesting information (DIOs) from other RPL nodes. After receiving a DIO from a candidate parent, the node can calculate the path cost up to this parent and up to the DAG root, taking into account the path cost information conveyed in the DIO. When multiple CPs are available, a PP is elected based on the lowest cost for the path up to the DAG root. After joining a DAG, a node can send or forward data in the upwards direction, towards the DAG root. Although RPL was initially intended to only support MultiPoint-to-Point (MP2P) upwards communications between the multiple devices and the DAG root, Point-to-Multipoint (P2MP) and Point-to-Point (P2P) communications are also supported in reverse direction. In order to support downward routes, unicast DAO messages can be sent from a child node, to its PP, in order to propagate destination information (addresses and prefixes) in the upward direction of the DAG. In this case, a DAO-ACK unicast message is sent in response to the DAO message.

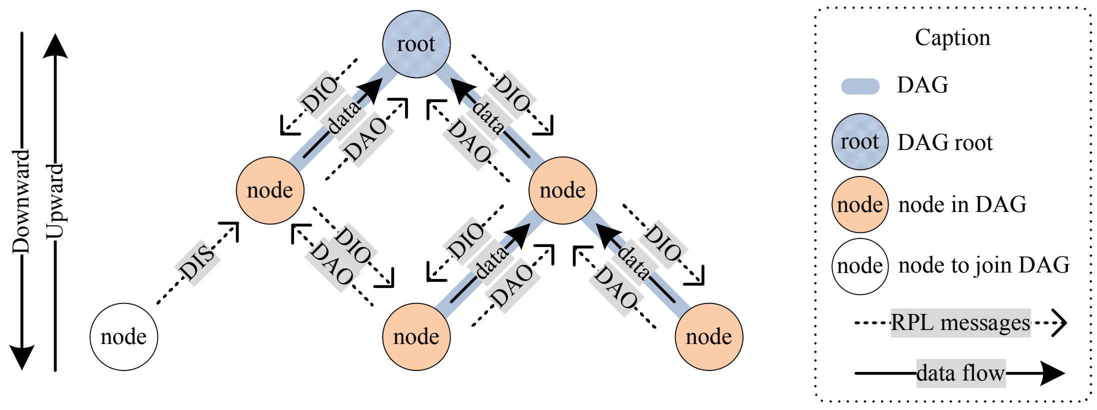


Figure 2.21: RPL control messages and data flow dynamics

DIO, DIS, DAO and DAO-ACK messages support one or multiple fields for options. Table 2.2 shows options which are allowed for each type of RPL control messages. The Pad1 and PadN options are used to insert one or more padding octets in order to align the option fields. The DAG Metric container is the only option used to convey metrics/constraints within the DAG. The DAG Metric Container can only be used in DIO and DAO messages and it can be used multiple times within these messages. The Route information option is used to indicate the prefixes available through the DAG root. The DAG configuration is used to distribute DAG configurations. The RPL Target is used to specify a reachable target address or prefix in the DAG. The Transit Information is used to indicate attributes for a path to one or more destinations. Solicited information option is used to request DIO messages. The Prefix Information option carry the prefix in use in the DAG for nodeś address auto configuration. The RPL Target Descriptor is used to qualify a determined target.

Table 2.2: Options in RPL Control Messages DIO, DIS and DAO

Type	**Options**	**DIO**	**DIS**	**DAO**
0x00	Pad1	X	X	X
0x01	PadN	X	X	X
0x02	DAG Metric Container	X		X
0x03	Route Information	X		
0x04	DAG Configuration	X		
0x05	RPL Target			X
0x06	Transit Information			X
0x07	Solicited Information		X	
0x08	Prefix Information	X		
0x09	RPL Target Descriptor			X

Two of these options perform a center role: the DAG Configuration and the Metric Container options. The DAG Configuration option format is shown in Fig. 2.22; this option is used to distribute the configuration information of the DAG. The information here conveyed is generally static and, therefore, it is not necessary to include it in all DIO messages. The Option Type field is set to 0x04 (DAG Configuration). The Option Length field conveys the length of the DAG Configuration in bytes. The Flags field is always initialized to zero by the sender and ignored by the receiver. The Authentication Enabled (A) field indicates whether the control messages are secured (bit set to “1”) or not (bit set to “0”). The Path Control Size (PCS) is used to configure the number of bits that may be allocated to the Path Control field, a field that allows nodes to request or allow for multiple Downward routes. The DIO Interval Doublings, DIO Interval Min, and DIO Redundancy Constant fields are used to configure the DIO Trickle timer, which controls the rate of the DIO control messages, and are defined in RFC 6206 [130]. The Max Rank Increase field is used to configure the allowable increase in Rank in support of local repair. The Min Hop Rank Increase is used to configure the minimum increase in Rank between a node and any of its DAG parents. The Objective Code Point (OCP) identifies the OF (managed by the IANA). The Default Lifetime field specifies the default lifetime of all RPL routes. The Lifetime Unit field provides the unit, in seconds, that is used to express route lifetimes in RPL.

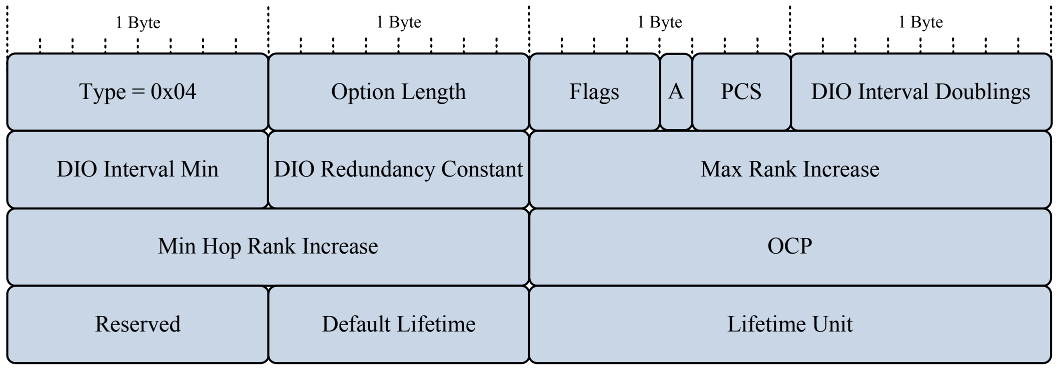


Figure 2.22: DAG Configuration option

The DAG Metric Container option format is shown in Fig. 2.23. It transports the values of the metrics along the DAG. This option may appear multiple times in the same RPL control message. The Option Type field is set to 0x02 (DAG Metric Container). The Option Length field indicates the length of the Metric Data, in bytes. The Metric Data field includes the order, the content, and the coding of the DAG Metric Container data as specified by the RFC 6551 [131].

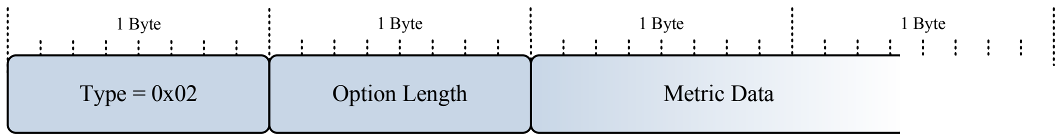


Figure 2.23: DAG Metric Container option

In RFC 6551 [131], ROLL defines a set of routing metrics/constraints types, managed by IANA. Table 2.3 shows these types and classifies them as being either node or link related (defined as the scope). ROLL defines that these metrics/constraints can be addictive, maximum, minimum or multiplicative.

Table 2.3: Routing Metric/Constraint Types and Scopes

Type	**Metric/Constraint**	**Scope**
1	Node State and Attribute	Node
2	Node Energy	Node
3	Hop Count	Link
4	Link Throughput	Link
5	Link Latency	Link
6	Link Quality Level	Link
7	Link ETX	Link
8	Link Color	Link

The RPL implementation in Contiki OS is named ContikiRPL [132]. The performance of this implementation is evaluated in [133] and a detailed analysis of its interoperability can be found in [134].

2.3.5 Simulation

A WSN deployment is preceded by design and test steps. In this context, the simulation tools are useful for developers in order to evaluate design, configurations and behaviors of a WSN prior to its deployment.

Multiple simulators are available for WSNs. Simulators such as OMNeT++ [135], ns-2 [136] or ns-3 [137] assume simplified versions of the real software and hardware of the motes, while other simulators as J-Sim [138] and Sensor Network Package, SENS [139], TOSSIM [140], ATEMU [141] and Cooja [142] allow the simulation of the WSN devices and networks at different levels of abstraction, from physical to application; in some extent, they enable the emulation of particular WSN nodes. In particular, TOSSIM [140] was designed to emulate Tiny OS [95] motes, ATEMU [141] was designed for MICA [84] platform and Cooja was designed for Contiki OS platform motes. The Cooja allows the emulation of Z1 [83], the MICAz [84], or the Tmote Sky [87]. In scenarios using Contiki OS the developers may rely on a development environment named InstantContiki which consists of an Ubuntu Linux running in a VMWare [143] virtual machine with a set of developer tools. Up to date, the latest version of Contiki is 2.7 and it includes the Cooja simulator [142].

Simulators such as ns-2 [136] or ns-3 [137] assume that motes are simplified versions of the real hardware, while Cooja uses full Contiki's source code and real hardware emulation to obtain close-to-real results and enables the fast deployment of the simulated experiments directly over the real motes. Cooja simulator is considered as a well suited validation tool for WSN experiments where contiki OS is used.

In Cooja, the emulation of hardware nodes, the use of the full source code, and the demand for detailed output logs, lead to large simulation times. The simulation times increase when developers need to run multiple simulations in order to obtain statistically sound results. Cooja runs as single-threaded and this means that it uses a single process and a single core at each instant of time, thus if Cooja runs within a machine where a multi-core processor is available, these cores will be underused. This Cooja limitation motivated the development of a simulation framework proposed in [144] and [145] in order to reduce simulation runtimes by using multiple simultaneous Cooja instances.

2.3.6 Discussion

This section provided an overview on the WSN operation and constraints. The hardware and the OS used in a WSN are intended to operate over a long period of time, using batteries and, in some cases, harvesting energy from the surrounding environment. Thus, these devices are designed with processing and communications constraints in order to cope to their limited energy resources.

Considering the specific hardware limitations of WSNs, the IEEE 802.15.4 standard was proposed. This section has detailed the main characteristics of this standard its mode of operation.

This section also provided an overview of the IP-based stacks proposed to be deployed in WSNnodes. These stacks require minimal specifications in terms of memory and processing capabilities, while providing the major IP and IPv6 functionalities. lwIP, uIP, and uIP6 were detailed and their requirements regarding ROM and RAM resources are compared.

The operation of a WSN as a LLN requires a routing protocol, and RPL was designed specifically for the LLNs. This section has provided an overview regarding the operation of this routing protocol as well as its implementations in different nodes pl.

Finally, this section has provided an overview of the available simulators for WSNs with a special emphasis on the Cooja simulator. Cooja simulator provides a close-to-real simulation by using the full Contiki's OS source code, however, this often implies long simulation runtimes which can take up to several hours or even days when performing multiple rounds of simulations.

2.4 Summary

The initial part of this chapter was dedicated to delay measurement and estimation. Section 2.1 provided a detailed characterization of the delays that can be accounted in IP networks and revised the methods available to measure these delays in different network points. Finally, this section described methods that can be used to evaluate the accuracy of a delay estimation process.

In Section 2.2 is provided an overview on actual techniques to perform admission control in IP networks with emphasis on distributed AC proposals. The challenges that emerge when these AC mechanisms are deployed in processing and energy constrained WSN nodes are also depicted.

In Section 2.3 is described the operation and the constraints associated to the sensor nodes in WSNs, regarding hardware, operative systems, and the implemented stacks from the PHY layer to the application layer. A description of RPL operation is also presented. Finally, is provided a general overview of available WSN simulators.

Chapter 3

EED Estimation

The real-time application to be deployed in a WSN generates packets which are assumed to have a maximum EED to reach destination. The main purpose of providing an EED estimation is to anticipate if a packet will be delivered within the EED limit defined by its application.

The state of the art solutions on EED measurement and estimation described in Chapter 2 do not provide a real-time and per-packet delay estimation that is oriented for the WSNs operation and constraints. An EED estimation suitable for a WSN must provide a minimal overhead and avoid the introduction of extra traffic, while still enabling the delay estimation to be available at the source node.

This Chapter proposes a novel EED estimation mechanism intended to provide a per-packet delay estimate from source to destination, avoiding negative impacts on the network performance. This estimation mechanism was defined for the particular scenario of a grid topology WSN, shown in Fig. 1.3; this topology includes three types of nodes: the source, the forwarder and the destination nodes.

The source node generates packets to be sent to destination node; the forwarder node forwards packets from other nodes and can also generate its own packets; the destination node, the network sink, is the destination of all generated packets. These nodes are represented in Fig. 3.1 where the source node s uses the forwarder node f to reach the destination node d. Node f also generates its own packets towards the destination node. Therefore, for the adopted scenario, the EED is the delay comprehended between the application at the source node and the application at the destination node.

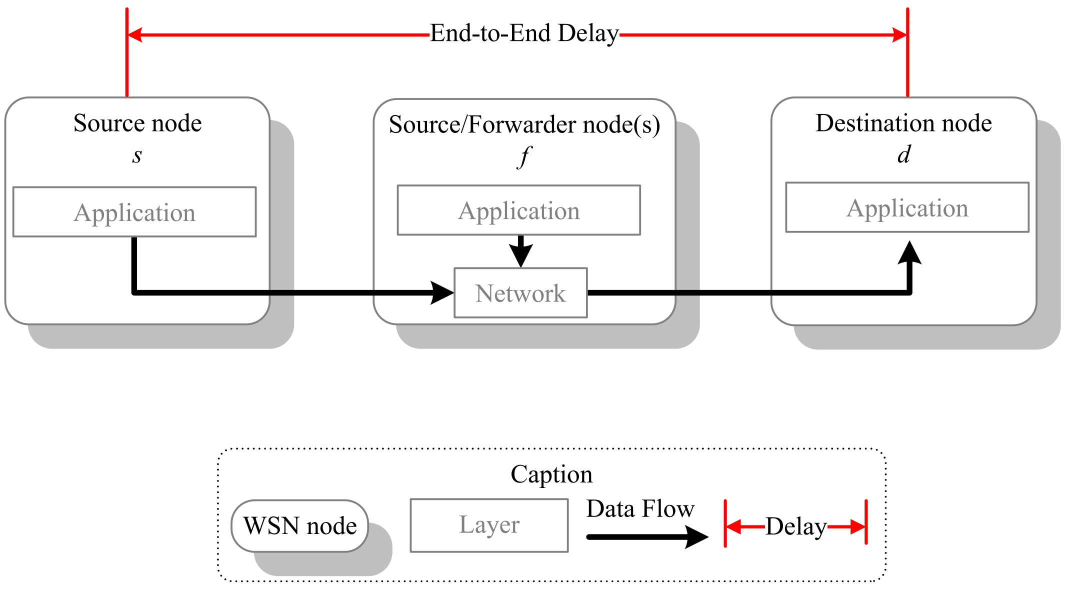


Figure 3.1: WSN nodes and End-to-End Delay estimation scope

3.1 EED Estimation Mechanism

Our proposed EED Estimation Mechanism (EEDEM) estimates the EED for each packet based on the delay experienced by previous data packets sent along the path from the application layer at the source node to the destination's node application layer. An overview of the EED estimation mechanism is presented in Fig. 3.2; for simplicity the figure shows only functions above network layer. The EED estimation is performed by using two functional components: the Internal Delays and the External Delays. The Internal Delays accounts for delays inside the node, while the External Delays captures other nodes' Internal Delays values transported via the RPL routing protocol.

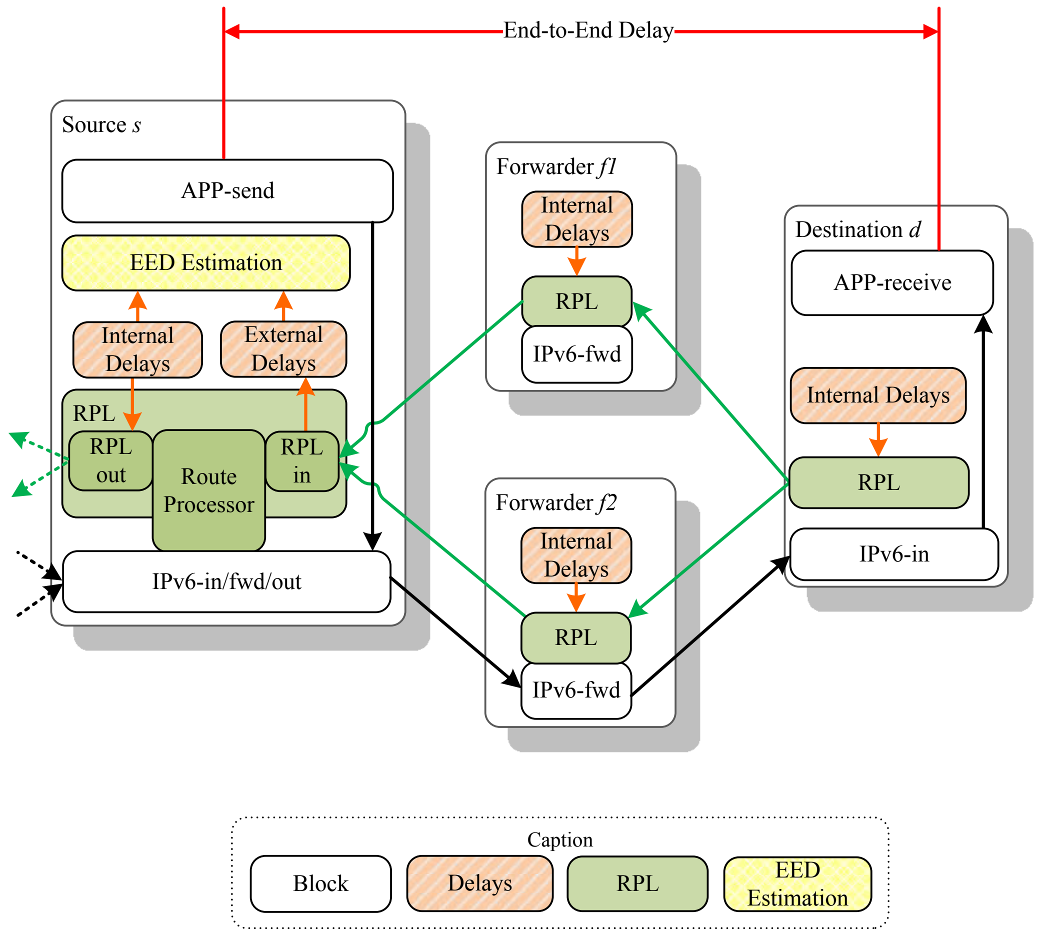


Figure 3.2: EEDEM overview

3.1.1 Internal Delays

EEDEM estimates the EED by measuring all the delays between the labels where the data passes through, from the application in the source node to the application in the destination node. Fig. 3.3 presents the layered communications architecture of WSN nodes, a data flow and a set of rounded-corner boxes inside each layer which represent labels characterizing relevant states in the data communications process. Delay accounting is accomplished by using timers that measure delays between labels inserted into parts of the code where the data flow passes through, ranging from the source application node to the application in the destination node. EEDEM assumes that the WSN nodes run the ContikiOS 2.5 [96] and thus, the labels above were inserted in the ContikiOS code files, according to Table 3.1. In order for all defined timers to have a millisecond precision, the time stamps are saved using 2 Bytes, i.e. with values ranging from 0 to 65535 *ms*.

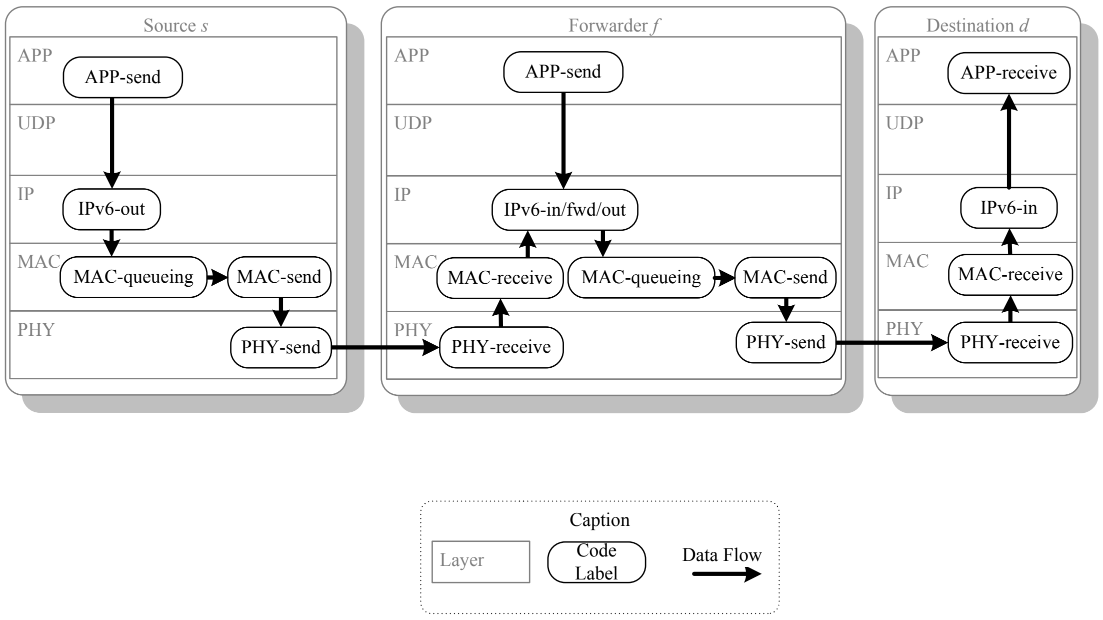


Figure 3.3: Labels (EEDEM)

Table 3.1: Relation between Labels, Functions and Contiki OS Files

Label	**Function in code**	**OS file**
APP-send	send_packet()	udp-client.c
APP-receive	receive_packet()	udp-server.c
uIP6-fwd/out	uip_process()	uip6.c
uIP6-in		
MAC-receive	input_packet()	contikimac.c
MAC-queuing	send_packet()	csma.c
MAC-send	transmit_queued_packet()	
PHY-send		
PHY-receive	mac_call_sent_callback()	

The Internal Delays account the time elapsed while the packet is processed within the stack of the source node, the time elapsed while in the MAC layer queuing and the time elapsed in packet transmissions. These internal paths and associated timers are shown in Fig. 3.4. The LxLyD format represents the delays between layer x and layer y, the MAC QueueD is the interval between the time the packet is inserted into the MAC queue until its removal, and the TransD is the time interval required for the packet successful transmission, including the ACK reception (frames used are configured to acknowledge reception, that is, the ACK Request subfield depicted in Fig: 2.12 is set to "1"). The Internal Delays includes the link related delays, (QueueD and TransD), and the processing related delays (LxLyD). It takes into account the time elapsed while packets are being processed inside the nodes.

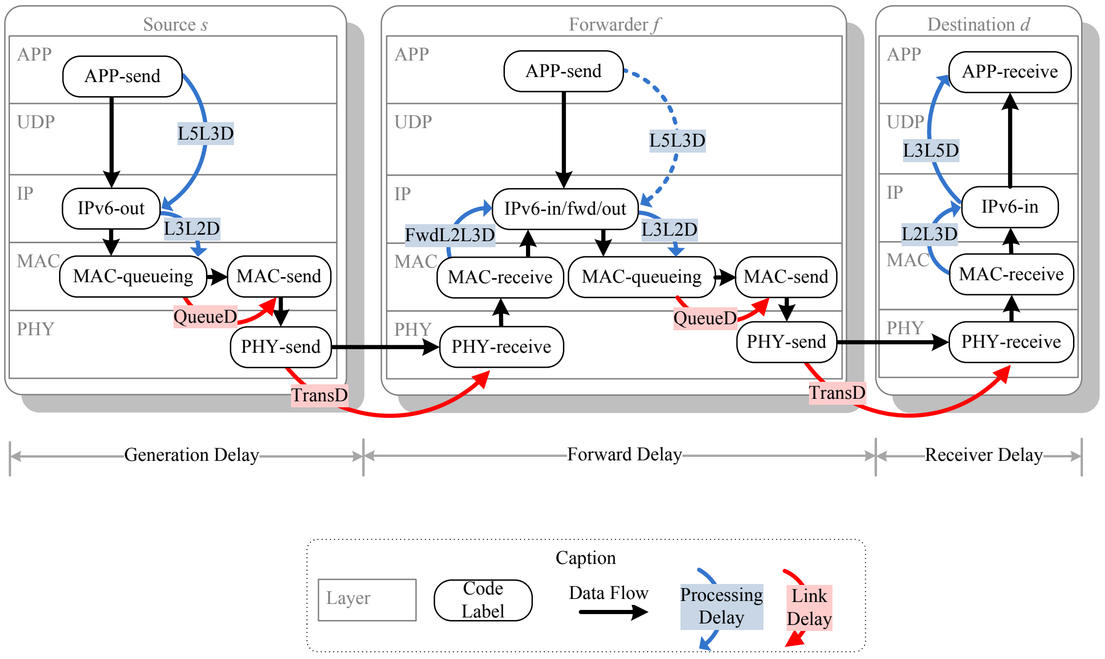


Figure 3.4: Labels and timers (EEDEM)

All the delays accounted in the Internal Delays (i.e. L5L3D, L3L2D, FwdL2L3D, L2L3D, L3L5D, QueueD, TransD) are obtained by using EWMA. Whenever a new delay item (Delay_i) is accounted, a delay estimation for a future packet p (Delay_p) is obtained using all delay history items as follows:

$$\text{Delay}_p = \beta.\text{Delay}_i + (1-\beta).\text{Delay}_{p-1} \tag{3.1}$$

According to the nodes' role, from packet generation until packet reaches destination, three types of Internal Delays are considered: the Generation Delay (GenD), registered when packets are generated, from the APP-send label down to the MAC-queuing label; the Forward Delay (FwdD), registered when packets are being forwarded, from MAC-receive label until the MAC-queueing label; the Receiver Delay (RcvD), registered when packets are received in the destination node, since MAC-receive label until they are delivered to the APP-receive label. Each of these Internal Delays, are obtained by the processing delay component (namely

Generation Processing Delay (GenProcD), Forward Processing Delay (FwdProcD), and Receiver Processing Delay (RcvProcD)) and the link delay components Generation Link Delay (GenLinkD), Forward Link Delay (FwdLinkD), and Receiver Link Delay (RcvLinkD). The processing delay component includes delays related to the processing time within the stack, i.e. all LxLyD, and the link delay component includes delays related to and dependent on the link (i.e. QueueD and TransD). For a source node s with a forwarder node f and a destination node d, the GenD estimated for a future packet p is obtained as follows:

$$\text{GenD}_p^{sf} = \text{GenProcD}_p^{s} + \text{GenLinkD}_p^{sf} \tag{3.2}$$

where:

$$\text{GenProcD}_p^{s} = \text{L5L3D}_p^{s} + \text{L3L2D}_p^{s} \tag{3.3}$$

$$\text{GenLinkD}_p^{sf} = \text{QueueD}_p^{s} + \text{TransD}_p^{sf} \tag{3.4}$$

The FwdD is obtained as follows:

$$\text{FwdD}_p^{fd} = \text{FwdProcD}_p^{f} + \text{FwdLinkD}_p^{fd} \tag{3.5}$$

where:

$$\text{FwdProcD}_p^{f} = \text{FwdL2L3D}_p^{f} + \text{L3L2D}_p^{f} \tag{3.6}$$

$$\text{FwdLinkD}_p^{fd} = \text{QueueD}_p^{f} + \text{TransD}_p^{fd} \tag{3.7}$$

The RcvD is obtained as follows:

$$\text{RcvD}_p^{d} = \text{RcvProcD}_p^{d} + \text{RcvLinkD}_p^{dd} \tag{3.8}$$

where:

$$\text{RcvProcD}_p^{d} = \text{L2L3D}_p^{d} + \text{L3L5D}_p^{d} \tag{3.9}$$

$$\text{RcvLinkD}_p^{dd} = 0 \tag{3.10}$$

3.1.2 External Delays and RPL Operation

In order to obtain the delay of the path up to the destination node in the source nodes, without imposing extra traffic, which would consume more processing and energy resources, the EEDEM conveys the External Delays using the RPL messages. EEDEM uses the following RPL metrics: the Path Delay Metric (PathDMetric) which represents the cumulative link delays up to the DAG root, and the Processing Delay Metric (ProcDMetric) which represents the cumulative processing delays up to the DAG root. Table 3.2 presents the mapping between the metrics used by EEDEM and the metric types defined by ROLL in [131].

Table 3.2: EEDEM routing metrics mapped to ROLL metric types

Type	**Scope**	**Mapped to**	
		Metric/Constraint	**Type**
PathDMetric	Link	Link Latency	5
ProcDMetric	Node	Node State and Attribute	1

Both metrics are addictive and used to obtain a node rank. According to Fig. 3.4 a node s with an RPL preferred parent f and destination d, advertises the following PathDMetric:

$$\text{PathDMetric}^{sd} = \text{FwdLinkD}_p^{sf} + \text{PathDMetric}^{fd} \tag{3.11}$$

and advertises the following ProcDMetric:

$$\text{ProcDMetric}^{sd} = \text{FwdProcD}_p^{s} + \text{ProcDMetric}^{fd} \tag{3.12}$$

The destination node d advertises to its neighbors the following metrics:

$$\text{PathDMetric}^{dd} = \text{RcvLinkD}_p^{dd} = 0 \tag{3.13}$$

$$\text{ProcDMetric}^{dd} = \text{RcvProcD}_p^{d} \tag{3.14}$$

3.1.3 End-to-end Delay Estimation Mechanism Output

Based on the previous operations with Internal and External Delays, the EED estimate ($\widehat{\text{EED}}$) is obtained using two major components: Path Delay (PathD) composed of the end-to-end path delay which corresponds to the sum of all link delays; ProcD which corresponds to the sum of all processing delays. Therefore, for a source s with parent f, the PathD and ProcD for future packet p sent to the destination d are obtained, respectively, using Eq. 3.15 and Eq. 3.16.

$$\text{PathD}_p^{sd} = \text{GenLinkD}_p^{sf} + \text{PathDMetric}^{fd} \tag{3.15}$$

$$\text{ProcD}_p^{sd} = \text{GenProcD}_p^{s} + \text{ProcDMetric}^{fd} \tag{3.16}$$

The procedure to obtain the $\widehat{\text{EED}}$ is implemented in all nodes of the WSN, except the in DAG root. When a node s needs to send a data packet p it estimates the EED towards destination d using PathD and the ProcD as follows:

$$\widehat{\text{EED}}_p^{sd} = \text{PathD}_p^{sd} + \text{ProcD}_p^{sd} \tag{3.17}$$

3.1.4 Validation Environment

A test scenario was deployed and the Cooja Simulator [142] was used to validate EEDEM. The network topology used is shown in Fig. 1.3 and the simulation parameters are presented in Table 3.3.

Table 3.3: Simulation Parameters (EEDEM)

Parameter	**Value**
Number of nodes	16 + sink node
Deployment area	100 m x 100 m
Transmission range	30m
Channel	Unit Disk Graph Medium
Packet size	100 Bytes
Transport/Application	UDP/CBR

Each node was simulated as a Tmote Sky [88] composed of a MSP430F1611 micro-controller and a CC2420 radio with a data rate of 250 *k*bit/s using IEEE 802.15.4 MAC and PHY layer specifications, with transmission and interference ranges of 30 m, and using the Unit Disk Graph Medium (UDGM) as physical channel model. The nodes run the Contiki OS 2.5 [96] and were programmed to enable the debug of application and RPL messages. Extra code was programmed to implement the timers in each node and the respective processing delay was measured, having an impact of 16 *m*s per processed packet. The application layer uses UDP as transport layer and it generates packets of 100 Bytes in a CBR by using constant

Inter-packet Generation Intervals (IGIs). Each simulation was configured to stop when the sink has received 500 packets from each node. Multiple simulations were conducted; in each round, the simulations were repeated 10 times using random seeds.

EEDEM was compared against an EED estimate provided by an ETT-based solution. The latter is based on Eq. 2.15 where RPL is configured to use the ETX metric, S is the packet size of 100 Bytes, and D is the data rate of the link of 250 $k$bit/s, as follows:

$$\mathrm{ETT} = \mathrm{ETX} \times \frac{S}{D} = \mathrm{ETX} \times \frac{(100 * 8)\ (\mathrm{bit})}{250\ (k\mathrm{bit/s})} = \mathrm{ETX} \times 3.2\ (ms) \tag{3.18}$$

In order to characterize the EED estimation accuracy, the EED Estimation Error (EEDError) for both solutions was compared. When a packet is generated each WSN node obtains an $\widehat{\mathrm{EED}}$. Also, simulator was configured to output the time instant when a packet is generated and when a packet reaches the destination application, obtaining the real packet's EED. Both $\widehat{\mathrm{EED}}$ and EED per each packet are saved, and compared with the EED. EEDError was obtained per each packet p according to Eq. 3.19.

$$\mathrm{EEDError}\ (ms) = \left|\widehat{\mathrm{EED}}_p - \mathrm{EED}_p\right| \tag{3.19}$$

At the end of each simulation, the Number of Received Packets (#RcvdPkts) and the Number of Sent Packets (#SentPkts) were collected and using #RcvdPkts the EEDError using MAE, i.e. $\mathrm{EEDError}_{(\mathrm{MAE})}$, was obtained per simulation according to Eq. 2.16, as follows:

$$\mathrm{EEDError}_{(\mathrm{MAE})}\ (ms) = \frac{1}{\#\mathrm{RcvdPkts}} \sum_{p=1}^{\#\mathrm{RcvdPkts}} \left|\widehat{\mathrm{EED}}_p - \mathrm{EED}_p\right| \tag{3.20}$$

Also, the $\mathrm{EEDError}_{(\mathrm{MAPE})}$ was obtained according to Eq. 2.19, as follows:

$$\mathrm{EEDError}_{(\mathrm{MAPE})}\ (\%) = \frac{1}{\#\mathrm{RcvdPkts}} \sum_{p=1}^{\#\mathrm{RcvdPkts}} \frac{\left|\widehat{\mathrm{EED}}_p - \mathrm{EED}_p\right|}{\mathrm{EED}_p} (\times 100) \tag{3.21}$$

In order to evaluate the distribution of both delay components (ProcD and PathD) in the total delay estimation, their weight was accounted and compared with $\widehat{\mathrm{EED}}$. To measure the RPL overhead introduced by both solutions, the average number of RPL packets per node was also accounted. In order to measure impact on network performance, the average EED and

Packet Reception Ratio (PRR) were also obtained. The average EED was obtained as follows:

$$\text{Average EED }(ms) = \frac{1}{\text{\#RcvdPkts}} \sum_{p=1}^{\text{\#RcvdPkts}} \text{EED}_p \tag{3.22}$$

The PRR was obtained as follows:

$$\text{PRR }(\%) = \frac{\text{\#RcvdPkts}}{\text{\#SentPkts}}(\times 100) \tag{3.23}$$

The average values of $\text{EEDError}_{(\text{MAE})}$, $\text{EEDError}_{(\text{MAPE})}$, EED, and PRR were obtained for each round of simulations as well as their respective confidence intervals of 90%.

3.1.5 Results

Fig. 3.5 shows the average $\text{EEDError}_{(\text{MAE})}$ and respective confidence intervals for ETT-based solution and EEDEM, for IGIs ranging from 1 to 10 s. The results show that, for IGIs below 2 s the $\text{EEDError}_{(\text{MAE})}$ for EEDEM is higher than the obtained for the ETT-based solution. For IGIs equal or larger than 2 s (smaller traffic loads), EEDEM presents an $\text{EEDError}_{(\text{MAE})}$ below the ETT-based solution. For an IGI higher than 3 s, the difference obtained from the both solutions is approximately constant, having a value around 250 *m*s. Both solutions present high values for the confidence intervals and they always overlap each other in the tested IGIs. This happens because the nodes closer to the sink (e.g. node 8) have an estimation error smaller than the nodes more distant from the sink (e.g. node 5). Also, for the nodes far from the destination node, the difference between the estimations using the ETT-based solution and EEDEM is higher.

Fig. 3.6 shows the average $\text{EEDError}_{(\text{MAPE})}$ for both solutions, for IGIs ranging from 1 to 10 s. The results show that, for IGIs shorter than 2 s, EEDEM presents an higher estimation error than the ETT-based solution. For IGIs equal or above to 2 s, the average $\text{EEDError}_{(\text{MAPE})}$ for both solutions are approximately constant, with EEDEM presenting lower percentage error of around 57%, against around 87% obtained for the ETT-based solution.

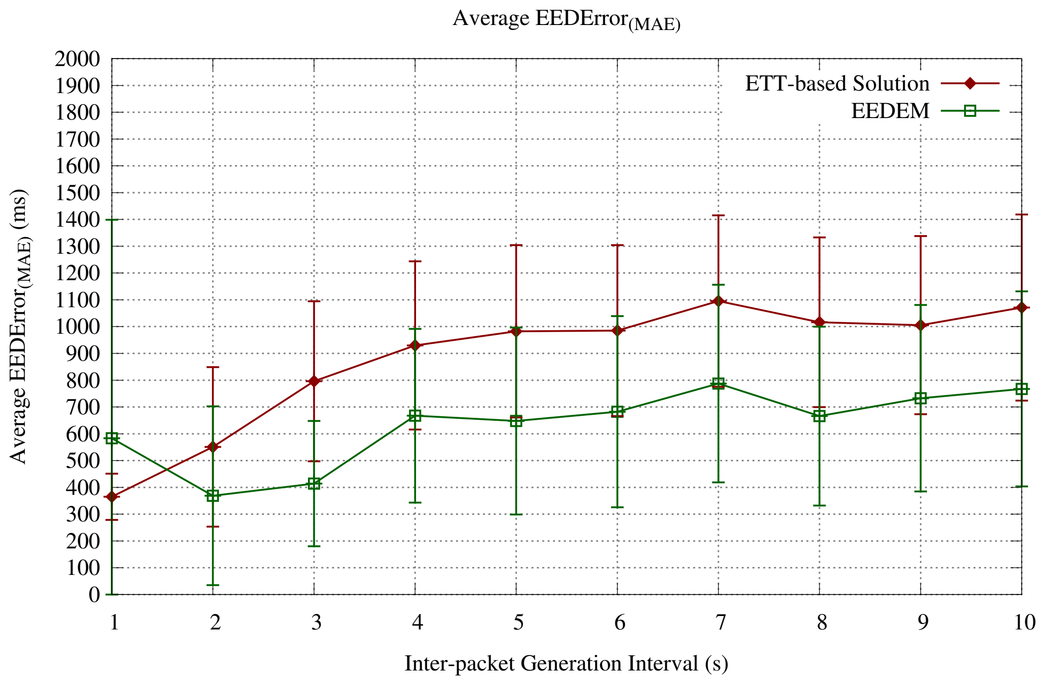


Figure 3.5: Average $\text{EEDError}_{(\text{MAE})}$ (EEDEM)

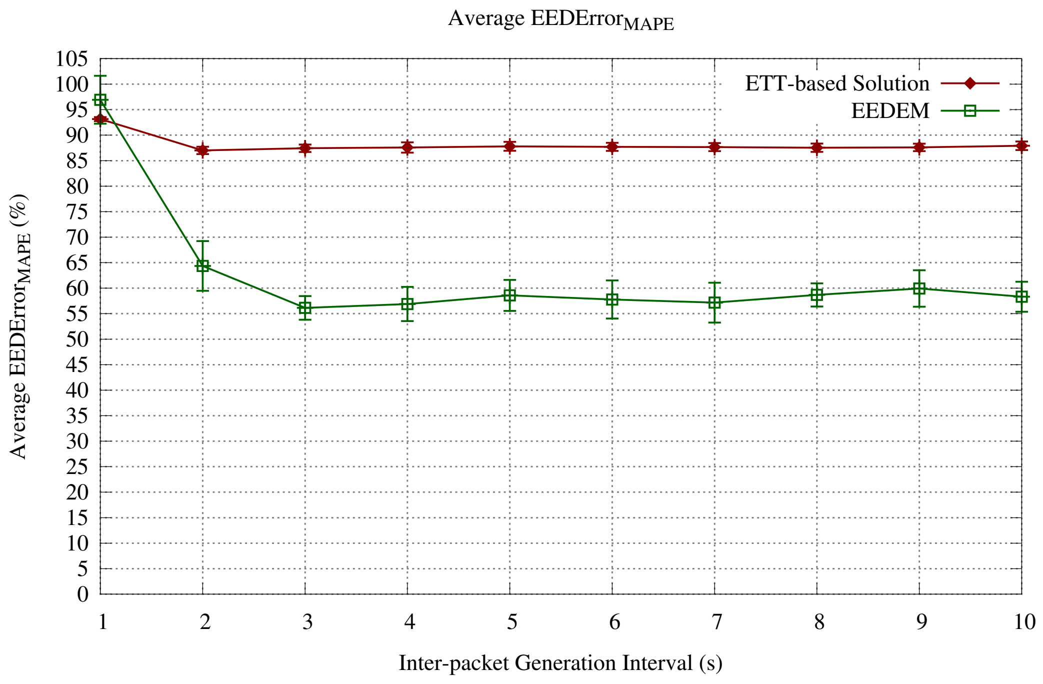


Figure 3.6: Average $\text{EEDError}_{(\text{MAPE})}$ (EEDEM)

Fig. 3.7 shows the average PathD and ProcD components distribution for EEDEM when IGIs range from 1 to 10 s. For IGI of 1 s, the ProcD represents approximately 15% of the total EED estimated while ProcD is the major component with around 75%. For IGIs larger than 1 s, the ProcD component becomes approximately constant and it accounts for about 40% of the EED estimation.

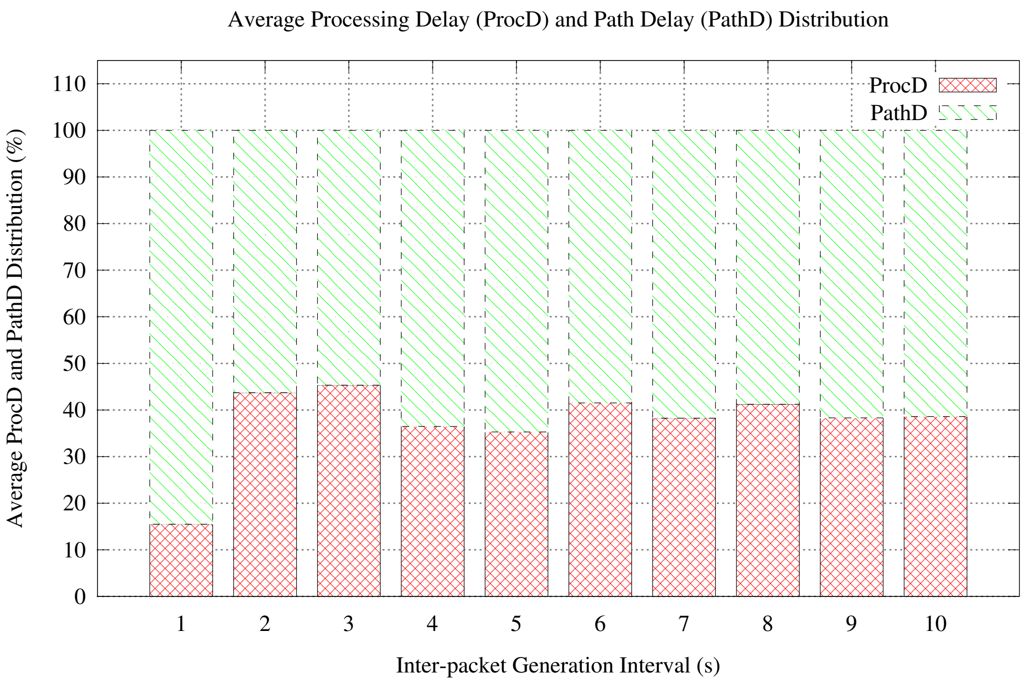


Figure 3.7: Average ProcD and PathD distribution (EEDEM) (IGIs from 1 to 10 s)

The Fig. 3.8 zooms Fig. 3.7 and shows the average PathD and ProcD distributions of the EED estimation, for EEDEM for IGIs ranging from 0.5 to 5 s. For IGIs ranging from 0.5 to 2.5 s, the ProcD component accounts 5% of the total EED estimation and it increases gradually up to 35%. For IGIs larger than 2.5 s, the ProcD component becomes approximately constant and it accounts for about 35% of the EED estimation. From the results shown in Figs. 3.8 and 3.7 it can be concluded that, for IGIs below 1.5 s, the PathD has an impact higher than the ProcD. For these high network loads, the PathD suffers from the links instability and it turns highly unpredictable making the EED estimation less accurate. For IGIs above 1.5 s, the ProcD represents around 30% of the EED estimation and EEDEM presents higher accuracy than the obtained by the ETT-based solution, benefiting from the consideration of processing delays.

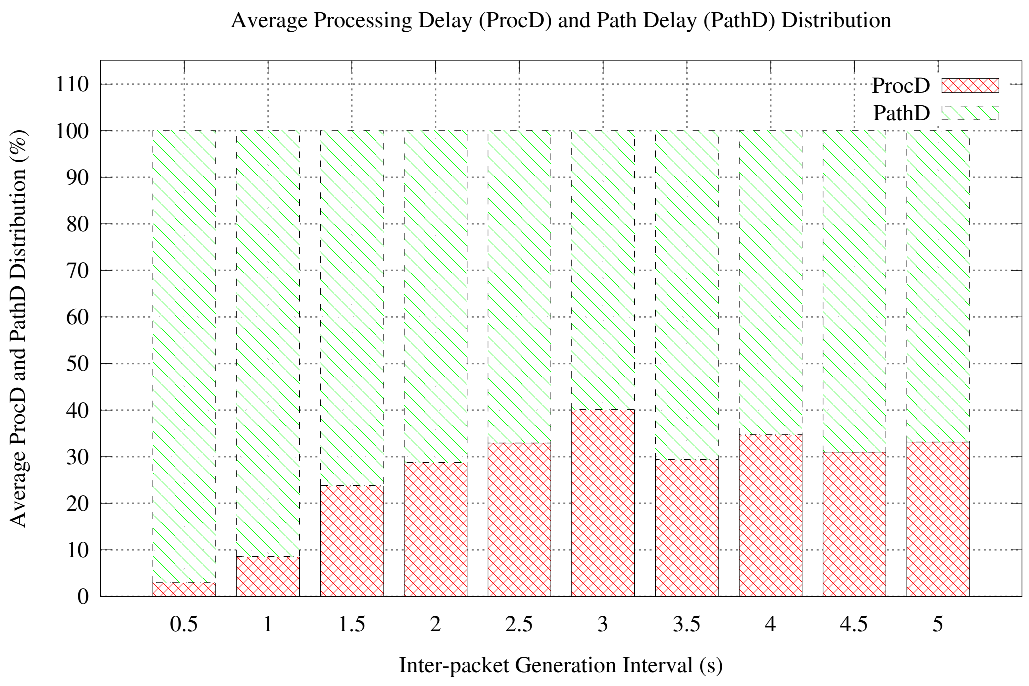


Figure 3.8: Average ProcD and PathD distribution (EEDEM) (IGIs from 0.5 to 5 s)

Fig. 3.9 presents the average number of RPL packets sent by both solutions, per node and per simulation, for IGIs ranging from 1 to 10 s. The results show that, for all IGIs, the number of RPL packets generated by EEDEM is always higher than those generated by the ETT-based solution. From the results shown it can be concluded that the RPL metrics used by EEDEM lead to an higher advertisement rate due to the ProcDMetric and PathDMetric changes that occur more often than the ETX metric. Thus, for shorter IGIs (high network loads), EEDEM has higher estimation errors than the ETT-based solution. This high advertisement rate becomes a benefit for EEDEM for IGIs larger than 1 s, since it enables a more accurate estimation.

Fig. 3.10 presents the average EED for both solutions, for IGIs ranging from 1 to 10 s. The results show that, for all IGIs, the average EED obtained for EEDEM is higher than the obtained for ETT-based solution. The difference between both solutions is more accentuated for IGIs smaller than 3 s. For IGIs equal or larger than 3 s, the difference is smaller and approximately constant.

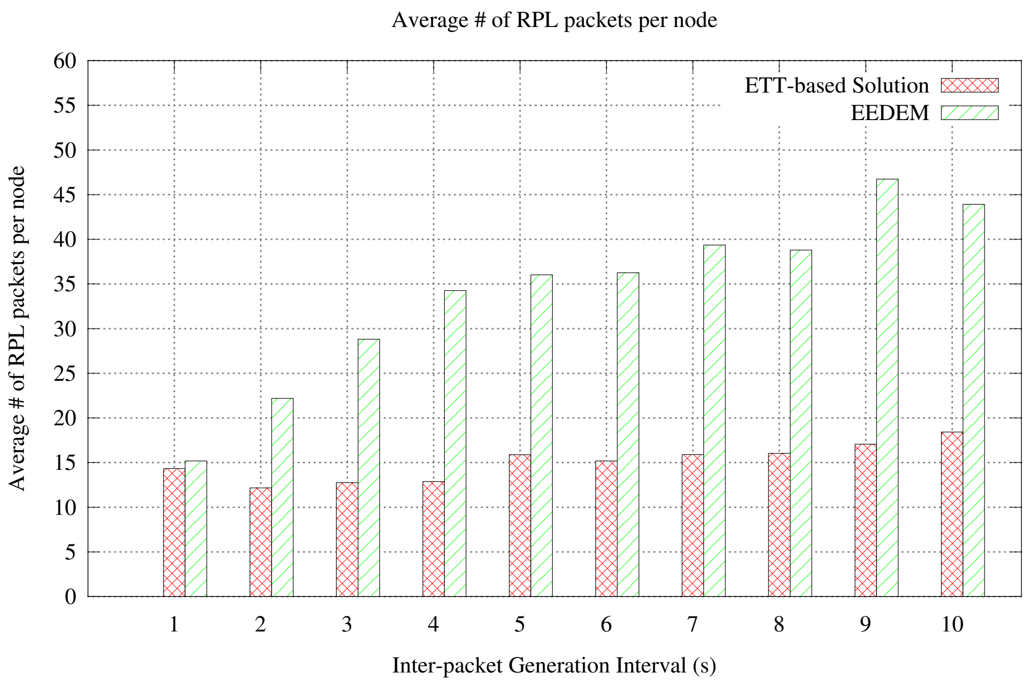


Figure 3.9: Average number of RPL packets per node (EEDEM)

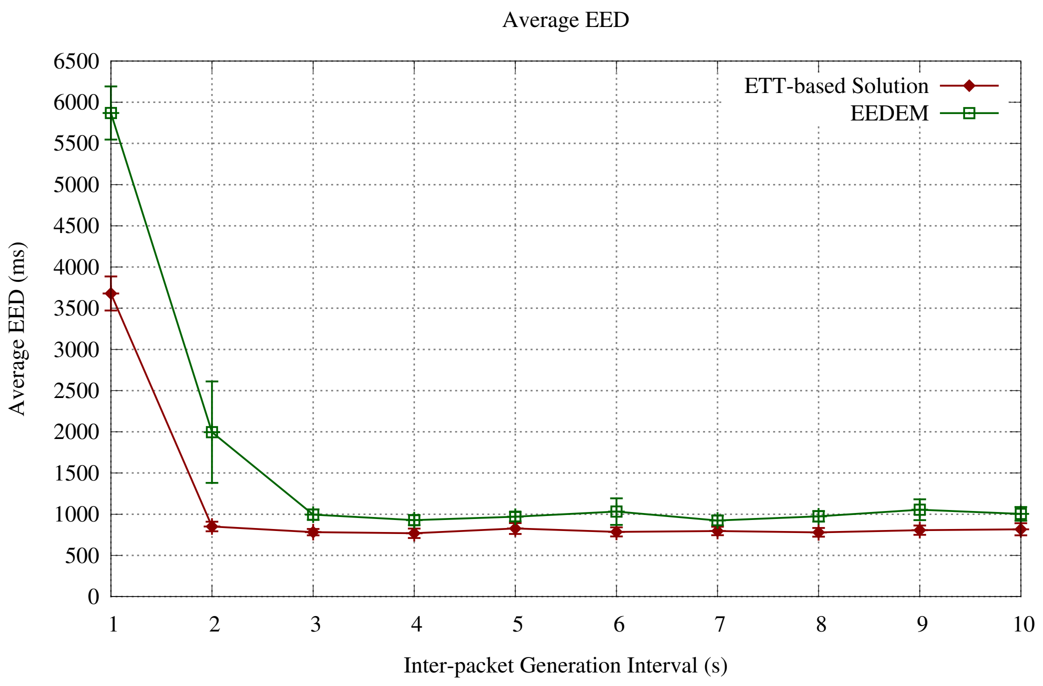


Figure 3.10: Average End-to-End Delay (EEDEM)

Fig. 3.11 presents the average PRR for both solutions, for IGIs ranging from 1 to 10 s. The results show that for IGIs shorter than 7 s, the average PRR of EEDEM is lower than the average PRR obtained for the ETT-based solution, with a constant difference of approximately 10 percentage points (pp). The results presented in Figs. 3.10 and 3.11 indicate that EEDEM has no significant impact on these performance items for an IGI higher than 7 s. For IGI shorter than 3 s, the average EED increases significantly when compared to ETT-based solution; for an IGI shorter than 7 s, the average PRR is affected in approximately 10 pp. This impact is due to the higher refresh rates of the RPL metrics used in EEDEM, as explained above.

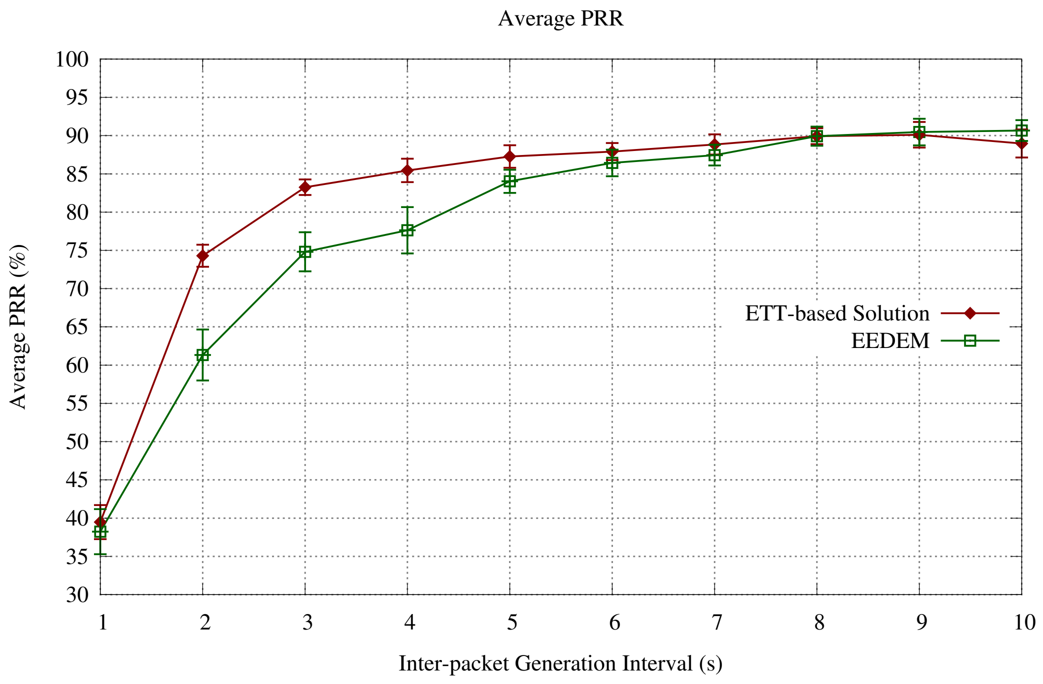


Figure 3.11: Average Packet Reception Ratio (EEDEM)

3.2 RPL Modifications

EEDEM estimates the EED based on the internal delay experienced by previously sent packets and delay information from other nodes through the use of RPL. Therefore, EEDEM depends on the RPL operation; a high refresh rate of routing messages increases the estimation accuracy, but it also increases the RPL overhead causing EED to become less predictable.

This section presents RPL Adaptation for EEDEM (RA-EEDEM), a set of modifications made to RPL in order to improve the EED estimation accuracy. The RA-EEDEM modifications were applied and tested within ContikiRPL [132], an open-source RPL implementation integrated in the ContikiOS [96]. ContikiRPL was used with the uIPv6 stack [105] and, at layer 2, with Carrier Sense Multiple Access (CSMA) and ContikiMAC [146].

Since routing protocol overhead has impact in EEDEM results, RA-EEDEM aims to balance the rate of RPL control messages. High rates, namely those providing feedback of the delays in downwards direction, will allow for higher estimation accuracy. Also, since these control messages compete with data messages for the available network resources, high rates of control messages cause the undesirable effect of increasing the average EED of data packets and degradation of average PRR.

The real-time application to be deployed is assumed only to generate data packets in upwards direction towards the DAG root. Thus, the RPL support for downward routes was disabled (using the MOP with value 0; see Section 2.3.4) and DAO messages were suppressed. Also, node mobility was not considered and therefore, DIS messages were neglected since they are only sent during an initial phase, before nodes join the DAG. The efforts were focused on balancing the rate of the DIO messages, taking into account the following conditions presented by order of importance:

- Maintain the regular routing process for the data packets - The RPL function and stability should not be compromised.
- Assume no specific application data rate - Prior to the deployment, the application data rate is taken as unknown.
- Maximize the accuracy of EED estimation - Improve EED estimation reducing the overhead of the routing protocol.
- Minimize the impact on performance - Minimize the impact on average EED and on average PRR performance.

Since application data rate is taken as unknown, the default configuration regarding DIO messages was not changed and the values defined in ContikiRPL were used. DIO messages are sent downwards and are mainly used to transport routing metrics. Such metrics will then be used by OF in order to select the PP, from a set of CPs. The metrics already defined in EEDEM (namely PathDMetric and ProcDMetric) are dynamic metrics and assume values with a millisecond precision which may cause fast oscillations on parent selection.

Based on these conditions, RA-EEDEM includes changes in two OF procedures: Selection of Best Parent and Update Metrics Procedures, detailed in next subsections.

3.2.1 Selection of Best Parent Procedure Modifications

Since parent selection instability imposes higher generation rate of DIO messages, the selection of best parent procedure was changed. This procedure is recursive and tests all the CPs within sets of two (*p1* and *p2*), returning the best one in each round. After testing multiple pairs of CP this procedure outputs a PP. The PP is recurrently compared with new pairs of CPs. The selection of best parent procedure was changed operate according to the Fig. 3.12. The first condition imposes this procedure to select parents with existing ProcDMetric values in favor of parents without ProcDMetric. This allows all nodes to quickly obtain processing delays, and thus improve estimation and reduce convergence time. After that, a Total Delay Metric (TotalDMetric) for each CP is calculated. If neither of both CPs is the PP, this procedure returns the parent with lowest TotalDMetric. If one of the parents is the PP, a latter comparison is performed using a variable Hysteresis Value (HystV) returned by the hysteresis function.

The HystV depends on the node's EED Rough Estimation (EED_RE) towards the DAG root. In order to obtain the EED_RE, a new metric, named Hop Count Metric (HopMetric) was added to those already defined in the initial EEDEM proposal (ProcDMetric and PathDMetric). The metrics defined were mapped according to the metric types defined by ROLL according to Table 3.4.

Table 3.4: RA-EEDEM routing metrics mapped to ROLL metric types

Type	**Scope**	**Mapped to**	
		Metric/Constraint	**Type**
PathDMetric	Link	Link Latency	5
ProcDMetric	Node	Node State and Attribute	1
HopMetric	**Link**	**Hop Count**	**3**

The HopMetric counts the hops up to the DAG root and thus a forwarding node f with an RPL Preferred Parent f and destination d, will advertise the following HopMetric:

$$\text{HopMetric}^{sd} = \text{HopMetric}^{fd} + 1 \tag{3.24}$$

The destination node d advertises, to its neighbors, the following HopMetric:

$$\text{HopMetric}^{dd} = 0 \tag{3.25}$$

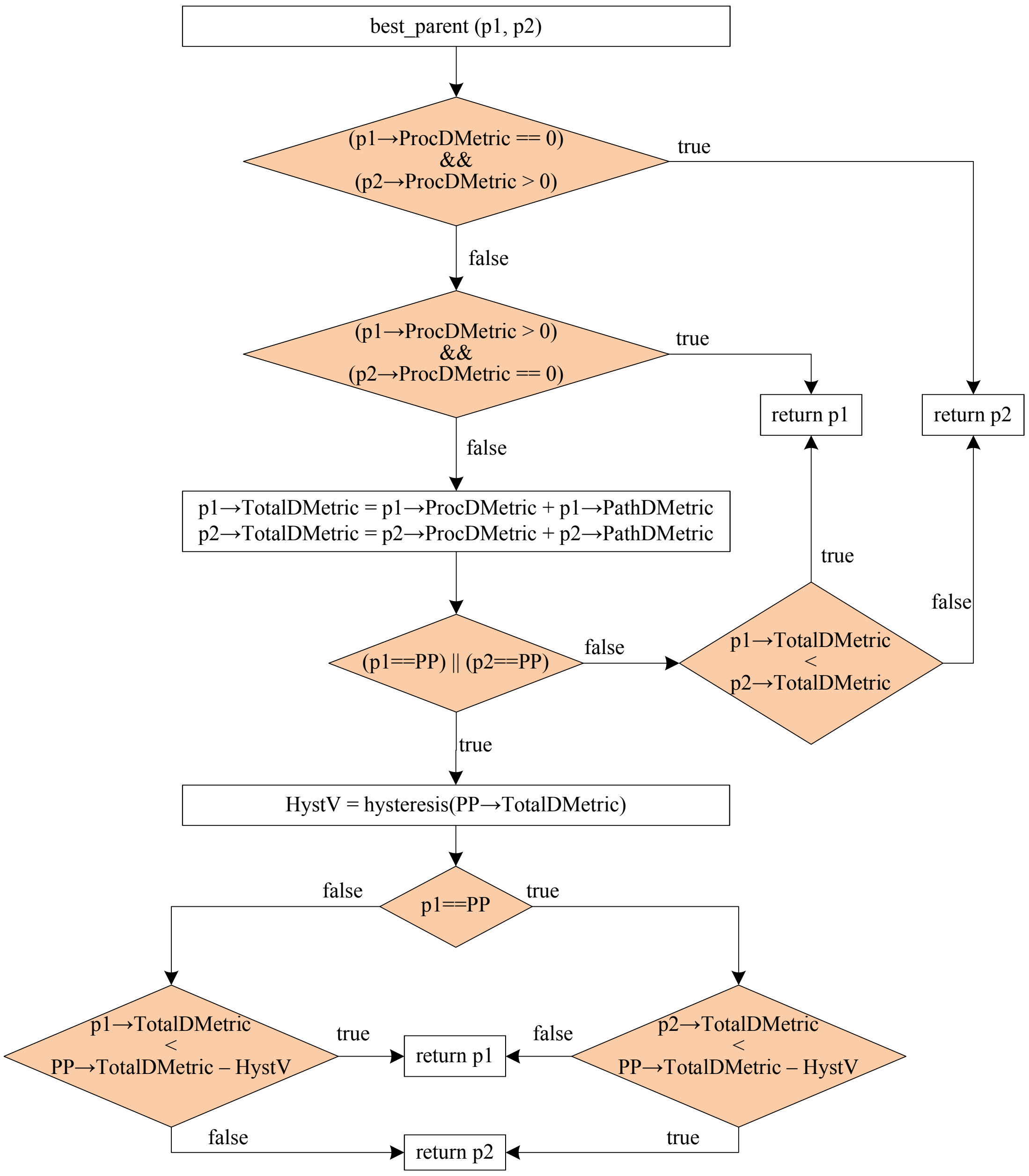

best_parent (p1, p2)
(p1→ProcDMetric == 0) && (p2→ProcDMetric > 0)
true
false
(p1→ProcDMetric > 0) && (p2→ProcDMetric == 0)
true
return p1
return p2
false
p1→TotalDMetric = p1→ProcDMetric + p1→PathDMetric
p2→TotalDMetric = p2→ProcDMetric + p2→PathDMetric
true
false
(p1==PP) || (p2==PP)
false
p1→TotalDMetric < p2→TotalDMetric
true
HystV = hysteresis(PP→TotalDMetric)
false
p1==PP
true
p1→TotalDMetric < PP→TotalDMetric – HystV
true
return p1
false
p2→TotalDMetric < PP→TotalDMetric – HystV
false
return p2
true

Figure 3.12: Selection of best parent procedure (RA-EEDEM)

The algorithm behind the hysteresis function is shown in Algorithm 1. The EED_RE is obtained using HopMetric multiplied by a constant value *K* that is assumed to be the worst transmission delay value per hop experienced by a sender node. The *K* value is obtained using a constant value of 125 *ms*, which is the default receiver wake-up interval defined in ContikiMAC [146] doubled to include MAC queue delay. The graph of the hysteresis function is presented in Fig. 3.13. If the PP→TotalDMetric is less than EED_RE, a negative slope line is used. Otherwise, the Minimum Hysteresis Value (MinHystV) is assumed to be 50 *ms*. The lower the PP→TotalDMetric value is, the higher the HystV will be, making the parent change less probable. Since a parent change will reset the DIO message timer, this algorithm controls the rate of DIOs in the network and avoids parent selection instability.

Algorithm 1: Hysteresis function (RA-EEDEM)

```
$hysteresis$(TotalDMetric){
    $K = 250$;
    EED_RE = $K \times$ HopMetric;
    $if$(TotalDMetric < EED_RE){
            HystV= $\frac{(\frac{EED\_RE}{2})-MinHystV}{0-EED\_RE} \times TotalDMetric + \frac{EED\_RE}{2}$;
    }$else${
            HystV= MinHystV;
    }
$return$ HystV;
}
```

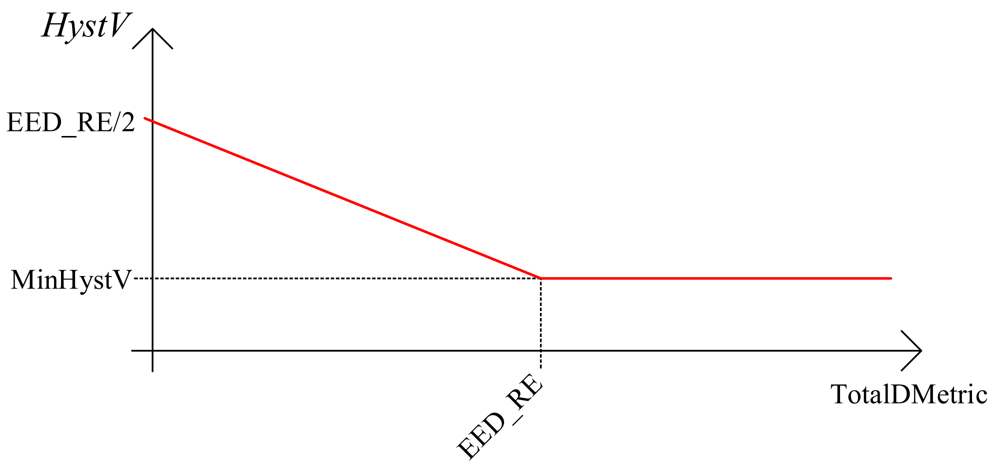


Figure 3.13: Hysteresis function graph (RA-EEDEM)

3.2.2 Update Metrics Procedure Modifications

In order to provide more accurate metrics to the remaining nodes, with delay information that starts from the DAG root, minor changes were also applied to the update metric procedure, which updates the metrics that are used in the DIO. As a result, each node will only advertise its metrics (ProcDMetric, PathDMetric and HopMetric) if these are already available from its PP. Whenever a node receives the metrics from a parent it should assume that all metric values in the metric container account the entire path to the DAG root.

3.2.3 Validation Environment

RA-EEDEM was evaluated in the grid topology shown in Fig. 1.3 with simulation parameters presented in Table 3.5.

Table 3.5: Simulation Parameters (RA-EEDEM)

Parameter	**Value**
Number of nodes	16 + sink node
Deployment area	100 m x 100 m
Transmission range	30m
Channel	Unit Disk Graph Medium
Packet size	100 Bytes
Transport/Application	UDP/CBR

The Cooja simulator [142] was used and each node was simulated as a Tmote Sky [88]. The destination node is defined as the DAG root. IEEE 802.15.4 MAC and PHY layer specifications were applied. The nodes ran the Contiki OS 2.5 and were programmed to enable both the debug of application and RPL messages. The application layer used UDP and it generates packets of 100 Bytes in a constant bit rate implemented with a constant IGI. Simulations were configured to stop whenever the destination received 200 packets from each node. Each simulation was configured to stop when the sink has received 500 packets from each node. Multiple simulations were conducted; in each round, the simulations were repeated 10 times using random seeds. The EED estimations obtained with RA-EEDEM were tested against those obtained with EEDEM.

The simulator was configured to output the instant of time when a packet is generated and when the packet reaches the destination application. In order to characterize the accuracy of the EED estimation, when a packet is generated the $\widehat{\text{EED}}$ was obtained, saved, and later compared with the EED. EEDError was obtained using MAPE according to Eq. 2.19 as follows:

$$\text{EEDError}_{(\text{MAPE})}(\%) = \frac{1}{\#\text{RcvdPkts}} \sum_{p=1}^{\#\text{RcvdPkts}} \frac{\left|\widehat{\text{EED}}_p - \text{EED}_p\right|}{\text{EED}_p} (\times 100) \tag{3.26}$$

In order to measure the RPL overhead introduced by each solution, the average number of RPL packets per node was accounted. To evaluate the impact on network performance, the average EED was accounted using Eq. 3.22, and the average PRR was accounted using Eq. 3.23.

3.2.4 Results

Fig. 3.14 shows the average $\text{EEDError}_{(\text{MAPE})}$ and its standard deviation for the three solutions (ETT-based solution, EEDEM and RA-EEDEM) using different IGIs. The results show that the $\text{EEDError}_{(\text{MAPE})}$ tends to be higher for shorter IGIs. For IGIs larger than 2 s, both RA-EEDEM and EEDEM solutions present an $\text{EEDError}_{(\text{MAPE})}$ lower than that obtained with the ETT-based solution. For an IGI equal or larger than 3 s, RA-EEDEM and EEDEM present estimation errors (values ranging from 50% to 60%) lower than the estimation error from the ETT-based solution (values ranging from 85% to 90%). RA-EEDEM presents errors which are 35 pp lower than the estimations obtained by the ETT-based solution, and 5 pp lower than the obtained with EEDEM.

Fig. 3.15 shows the average number of RPL packets generated per node and per simulation. The results show that for IGIs of 1 s all solutions generate almost the same number of RPL packets. For IGIs larger than 1 s, the ETT-based solution uses a lower number of RPL packets, when compared with the other two solutions. When comparing RA-EEDEM against EEDEM, RA-EEDEM presents a lower number of RPL messages in all circumstances. Combining the results from Figs. 3.14 and 3.15, it can be concluded that RA-EEDEM provides a more accurate EED estimation while reducing the overhead of the routing protocol.

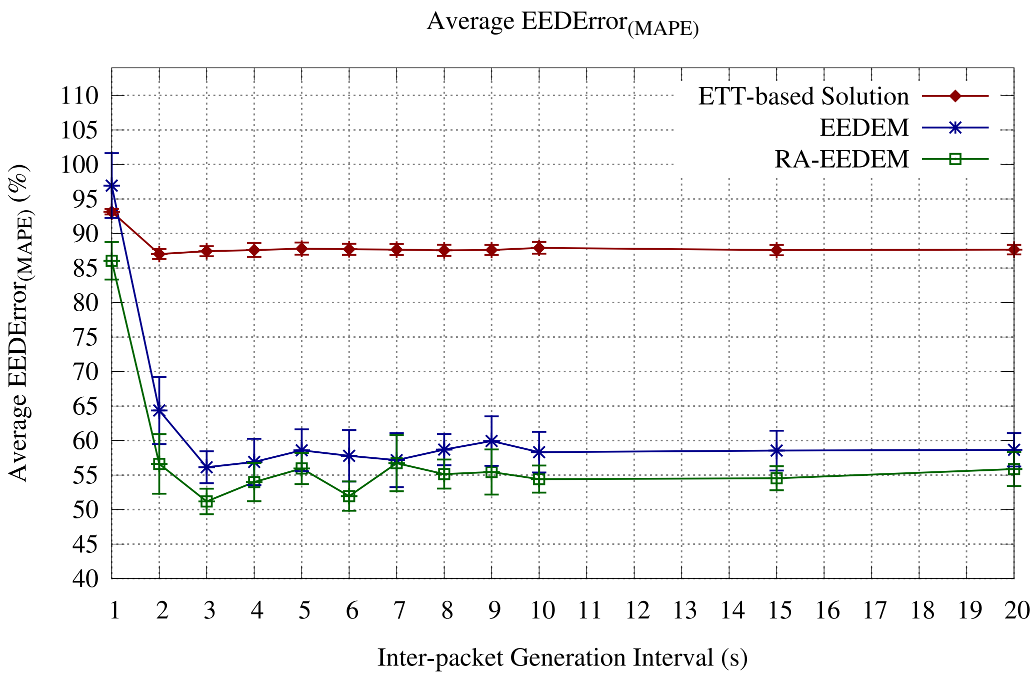


Figure 3.14: Average $\text{EEDError}_{(\text{MAPE})}$ (RA-EEDEM)

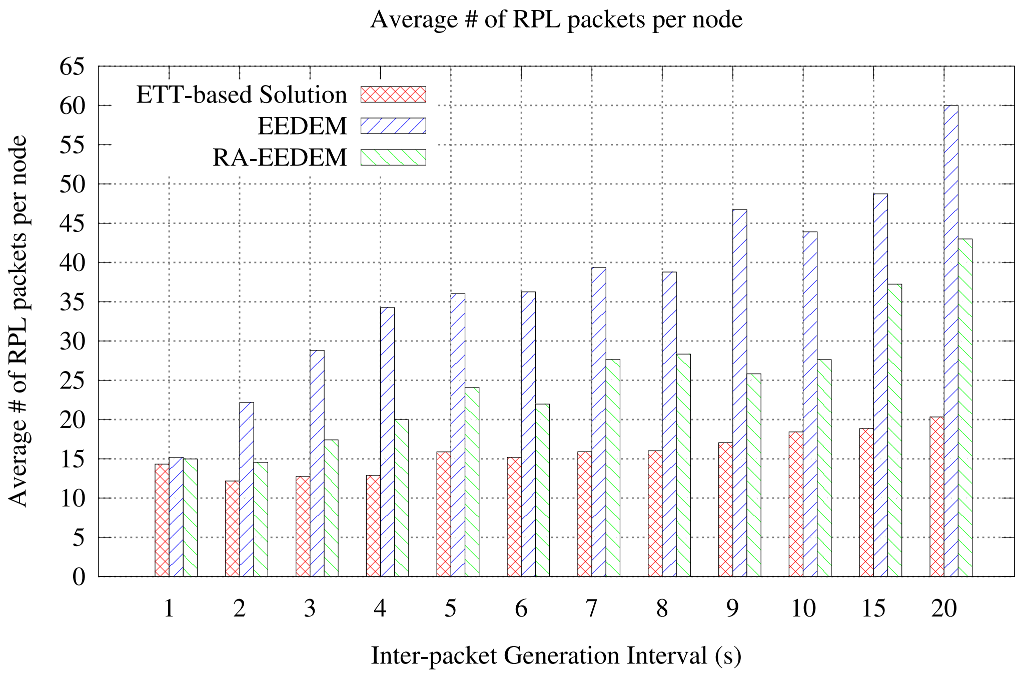


Figure 3.15: Average number of RPL packets per node (RA-EEDEM)

Fig. 3.16 shows the average EED for all solutions. The results show that RA-EEDEM and EEDEM present a higher EED, on average, when compared with the ETT-based solution. This is due to the variation of metrics used in RA-EEDEM and EEDEM which impose higher rate of DIO messages. Considering IGI values between 1 and 3 s, the RA-EEDEM solution presents a lower average EED, when compared to EEDEM. For IGIs larger than 3 s, both RA-EEDEM and EEDEM present approximately the same results, differing from the ETT-based solution by roughly 200 *m*s. With these results it can be concluded that the RA-EEDEM estimation present better results in terms of average EED in high network loads than those obtained using EEDEM.

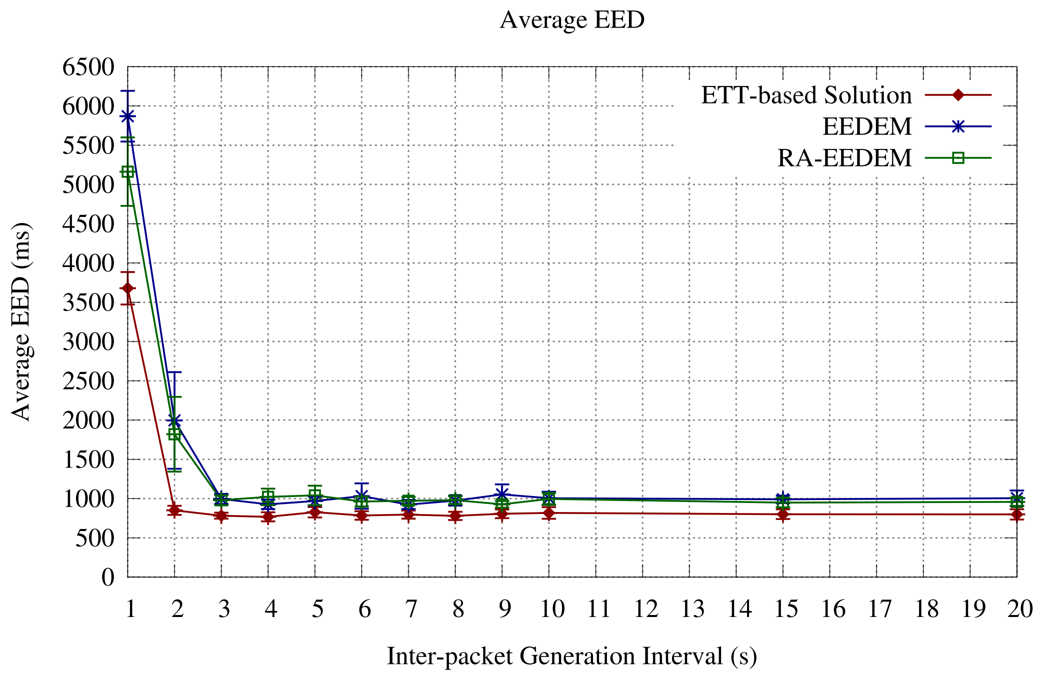


Figure 3.16: Average End-to-End Delay (RA-EEDEM)

Fig. 3.17 shows the average PRR for all solutions. For IGIs shorter than 7 s, the average PRR obtained by RA-EEDEM is about 5 pp higher than the average PRR obtained using EEDEM, and closer to the results obtained using ETT-based solution. For IGIs higher than 9 s, PRR of RA-EEDEM and EEDEM is higher than that obtained using the ETT-based solution. Combining the results shown in Figs. 3.15, 3.16 and 3.17, it can be concluded that the average EED and PRR will benefit from the reduction of the RPL overhead in high network loads.

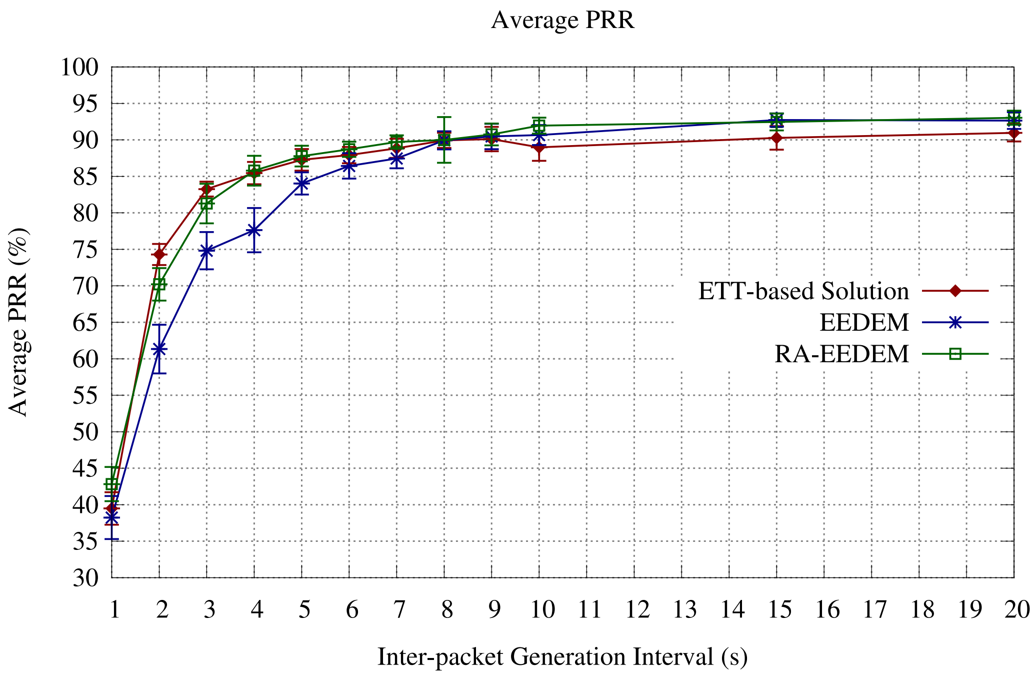


Figure 3.17: Average Packet Reception Ratio (RA-EEDEM)

3.3 Delay Accounting Optimization

EEDEM accounts delays using timers that make use of an EWMA function shown in Eq. 3.1, where the smoothing factor (β) is constant and 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. In this section is detailed an optimization procedure for EEDEM that works by evaluating the network load and by adapting the smoothing factor (β) of the EWMA function accordingly. Results show that this optimization leads to a more accurate EED estimation for different network loads.

3.3.1 Preliminary Experiments

Preliminary experiments were performed in order to better understand how the EEDError$_{(\text{SMAPE})}$ changes in relation to different β values. The Cooja simulator [142] was used and all nodes were simulated as Tmote Sky [88] with the simulation parameters presented in Table 3.6.

Table 3.6: Simulation Parameters (DAOP)

Parameter	**Value**
Number of nodes	16 + sink node
Deployment area	100 m x 100 m
Transmission range	30m
Channel	Unit Disk Graph Medium
Packet size	100 Bytes
Transport/Application	UDP/CBR

The application layer used UDP and it generated packets of 100 Bytes in a constant rate here defined as IGI. Each simulation was configured to stop whenever the destination node received 100 packets from each node and, in each round, the simulations 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 $\widehat{\text{EED}}$ was collected and later compared with the EED. Finally, when the simulation ended, the EEDError for a set of #RcvdPkts was obtained using the difference between $\widehat{\text{EED}}$ and EED calculated using the SMAPE according to Eq. 2.20, expressed in a value between 0% and 200%. SMAPE compares the difference between $\widehat{\text{EED}}$ and EED with the mean of these two values, thus treating over and under estimations equally. Thus, EEDError$_{(\text{SMAPE})}$ is obtained as:

$$\text{EEDError}_{(\text{SMAPE})}(\%) = \frac{1}{\#\text{RcvdPkts}} \sum_{p=1}^{\#\text{RcvdPkts}} \frac{\left|\widehat{\text{EED}}_p - \text{EED}_p\right|}{(\widehat{\text{EED}}_p + \text{EED}_p)/2} (\times 100) \tag{3.27}$$

The results obtained for the average EEDError$_{(\text{SMAPE})}$ and its confidence interval are shown in Figure 3.18. 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 value provided the lowest EEDError$_{(\text{SMAPE})}$, while for low network loads (IGI above 2.5 s) a lower β value 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.

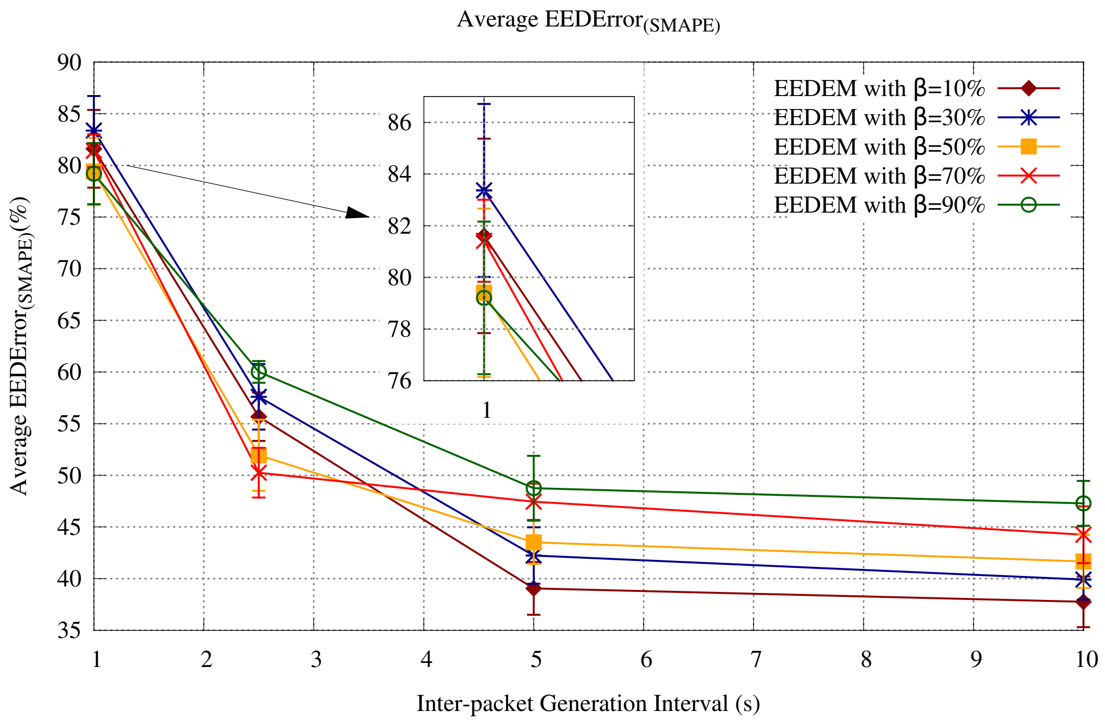


Figure 3.18: Average $\text{EEDError}_{\text{(SMAPE)}}$ using β varying from 10% to 90%

3.3.2 Delay Accounting Optimization Procedure

In order to minimize the $\text{EEDError}_{\text{(SMAPE)}}$, the preliminary experiments demonstrated that each node must be aware of its network load. Thus, the Dynamic 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 3.19 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 $\text{EEDError}_{\text{(SMAPE)}}$ 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)*.

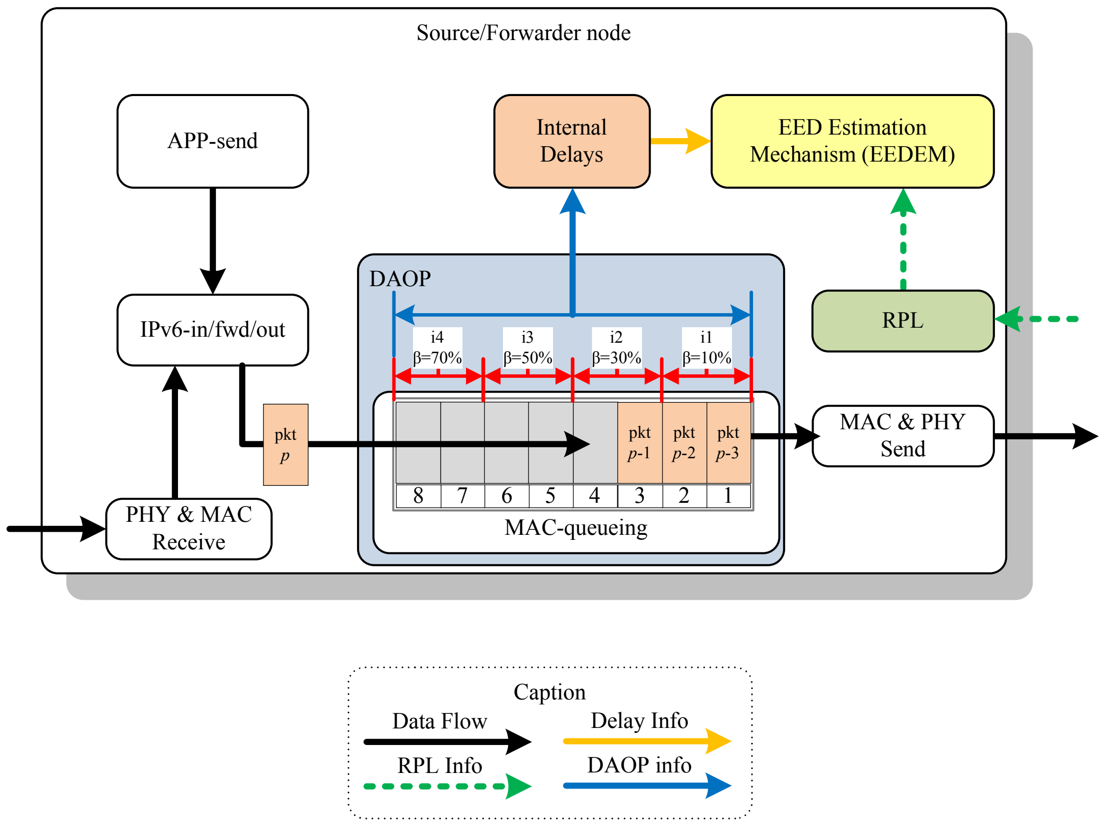


Figure 3.19: DAOP integration in EEDEM

3.3.3 Validation Environment

In order to evaluate DAOP, it were used all parameters described in Section 3.3.1. EED-Error$_{(\text{SMAPE})}$ was obtained and compared with the EEDErrors obtained in the preliminary experiments, when multiple constant β values were used. Also, to relate the EED accuracy to the network load, the MAC queue usage for each node was collected.

3.3.4 Results

The proposed solution monitors the MAC queue usage to infer the network load in real-time. Figure 3.20 shows the usage of the MAC queue for two cases: when the IGI is equal to 1 s, and when the IGI is equal to 5 s. The values were obtained in a node one hop away from to the destination, whenever a packet is to be sent. The results show that, for lower IGIs, the

MAC queue has roughly 6 or more packets, on average, and for an IGI equal to 5 s, the MAC queue has roughly 1 packet during all the simulated time.

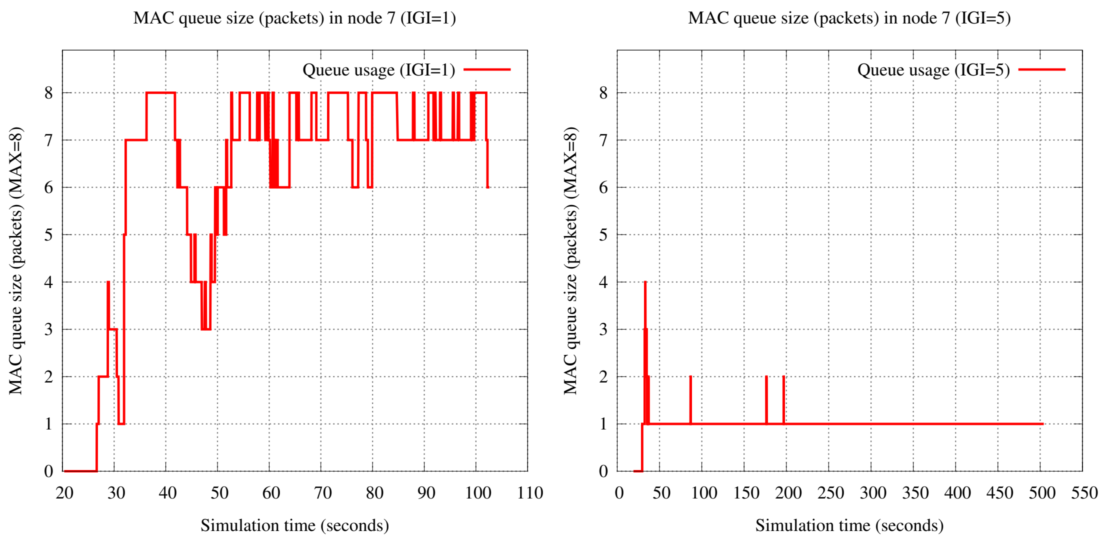


Figure 3.20: MAC queue usage (DAOP) (left: IGI=1 right: IGI=5)

Figure 3.21 compares the $\text{EEDError}_{(\text{SMAPE})}$ obtained using the DAOP 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 $\text{EEDError}_{(\text{SMAPE})}$ for all the different network loads.

3.4 Summary

Section 3.1 presented a novel real-time mechanism named EEDEM to estimate EED in a WSN. EEDEM estimates per-packet EED based on the delays obtained by previous packets and by combining internal timers with two cumulative RPL metrics. Also, EEDEM accounts not only transmission related delays, but also processing delays, which revealed to be important in a WSN where nodes have limited processing resources. EEDEM was compared to an ETT-based solution and the results show that it produces a more accurate EED estimation for the tested network loads, without impacting significantly on the network performance, namely regarding the average EED and the PRR values.

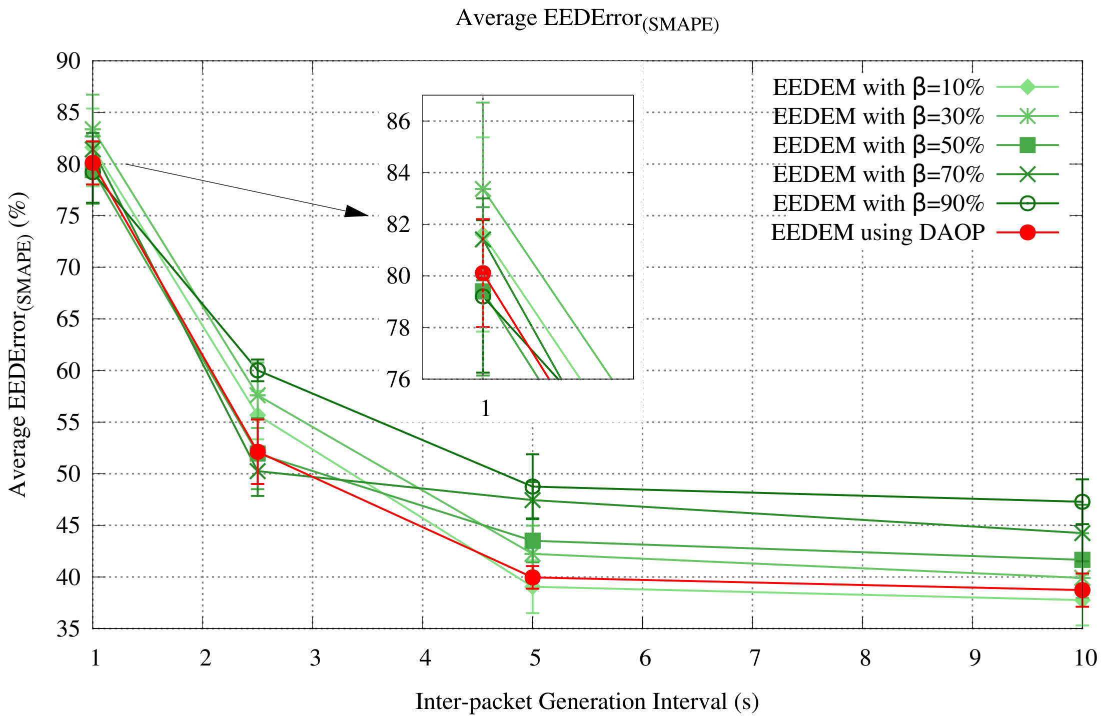


Figure 3.21: Average EEDError$_{(SMAPE)}$ using different β values and using DAOP

EEDEM depends highly on the RPL operation. The refresh rate of routing messages increases the estimation accuracy, but simultaneously increases the RPL overhead causing EED to become less predictable. Section 3.2 presented RA-EEDEM that consists of a set of modifications to RPL aimed to improve the accuracy of the EEDEM. RA-EEDEM comprises changes in OF procedures, namely in the selection of best parent and in the update metrics procedures. RA-EEDEM estimation results were compared to EEDEM and ETT-based solution and the results show that RA-EEDEM improves the accuracy of the EED estimation when compared to the other solutions. When compared to EEDEM, RA-EEDEM improves EED estimation, presents better average PRR, and less average EED for higher network loads.

In order to estimate the EED, EEDEM accounts internal delays obtained using an EWMA function, where the smoothing factor (β) is constant and defined *a priori*. Experiments showed that the best EED estimation error results are obtained by varying the β value as a function of the network load. Section 3.3 presented DAOP that dynamically adapts the β value by inferring the network load through actively monitoring the node's MAC queue size. The results obtained show that DAOP provides a more accurate EED estimation for different network loads than those obtained in EEDEM.

EEDEM and its subsequent improvements (RA-EEDEM and DAOP) intend to provide a WSN node with the most accurate EED estimation when requested. Further, an admission control mechanism interfacing with EEDEM is required in order to drop useless packets, reduce useless network usage and improve the overall performance of the network, while saving energy.

Chapter 4

Distributed Admission Control

Chapter 3 described EEDEM that provides an EED estimate per packet in a WSN. This mechanism is useful to preview, at a source node, if the packet will likely be received at the destination within the EED limit defined by the application. The real-time application envisioned to be deployed in the WSN is assumed to generate delay sensitive packets; each packet will be considered useful if it is delivered within the EED defined by the application, and useless otherwise.

An AC mechanism may be adopted in order to decide if the packets should be transmitted or not according to their potential usefulness. This AC mechanism should accept or drop a packet based on the packet's expectation to comply or miss the EED deadline previously defined by the application. The decision should be taken in real-time at each WSN node and, as consequence, a distributed AC mechanism should be designed.

None of the mechanisms surveyed in Chapter 2 provides a per-packet distributed admission control mechanism using a cross-layer approach in order to control the generation and forwarding functions of a WSN node. In this context, a novel AC distributed mechanism is proposed and evaluated in this chapter.

4.1 Cross-Layer Admission Control Mechanism

The Cross-layer Admission Control (CLAC) is a distributed mechanism running in each WSN node that intercepts packets, requests EED estimations and decides if the node should accept or reject the packets, according to their usefulness estimation regarding the destination application. This decision is taken per packet and it is based on information provided by application regarding the Maximum EED (MaxEED) allowed for its packets, and the real time information conveyed by the network regarding the EED estimation; thus, this mechanism is classified as an hybrid of parameter based and measurement based AC proposal, according to classification presented in Section 2.2.1. CLAC is designed to enhance network performance while fostering energy efficiency in a grid WSN.

Since not all received packets are useful for the destination, in alternative to the common PRR, it is defined Packet Usefulness Ratio (PUR) as the Number of Useful Packets (#UsefulPkts) per #RcvdPkts as follows:

$$\text{Packet Usefulness Ratio (PUR) (\%)} = \frac{\text{\#UsefulPkts}}{\text{\#RcvdPkts}}(\times 100) \tag{4.1}$$

The #UsefulPkts can be accounted as follows:

$$\text{\#UsefulPkts} = \sum_{p=1}^{\text{\#RcvdPkts}} U(p) \tag{4.2}$$

where *U(p)* is 1 if packet *p* is useful, or 0 otherwise, expressed as follows:

$$U(p) = \begin{cases} 1 \rightarrow \text{EED}_p \leq \text{MaxEED} \\ \\ 0 \rightarrow \text{EED}_p > \text{MaxEED} \end{cases} \tag{4.3}$$

In order to preview *U(p)*, CLAC uses the *Usefulness Preview (UP)* function for packet *p* expressed as follows:

$$UP(p) = \begin{cases} \text{progress} \rightarrow \widehat{\text{EED}}_p \leq \text{MaxEED} \\ \\ \text{drop} \rightarrow \widehat{\text{EED}}_p > \text{MaxEED} \end{cases} \tag{4.4}$$

where $\widehat{\text{EED}}_p$ is the estimated EED for each packet *p*, i.e. the result of Eq. 3.17. Fig. 4.1 depicts the *UP* function defined as a binary result for the hard deadline MaxEED. A packet having $\widehat{\text{EED}}$ below MaxEED is an in-profile packet and should progress; above MaxEED it is an out-of-profile packet and should be dropped.

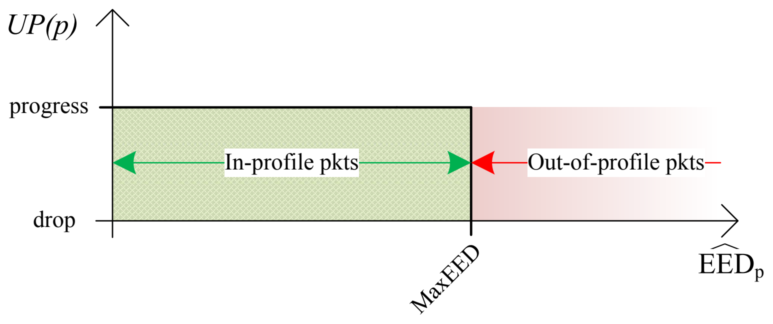


Figure 4.1: Usefulness Preview function (CLAC)

Fig. 4.2 shows the integration of the CLAC mechanism in the WSN. CLAC assumes two types of nodes: the source/forwarder node that generates packets or forwards packets from other nodes, and the destination node which consumes the packets. CLAC is deployed only in the source/forwarder nodes with the support of the EED estimation mechanism that, in turn, depends on delay information conveyed by RPL control messages.

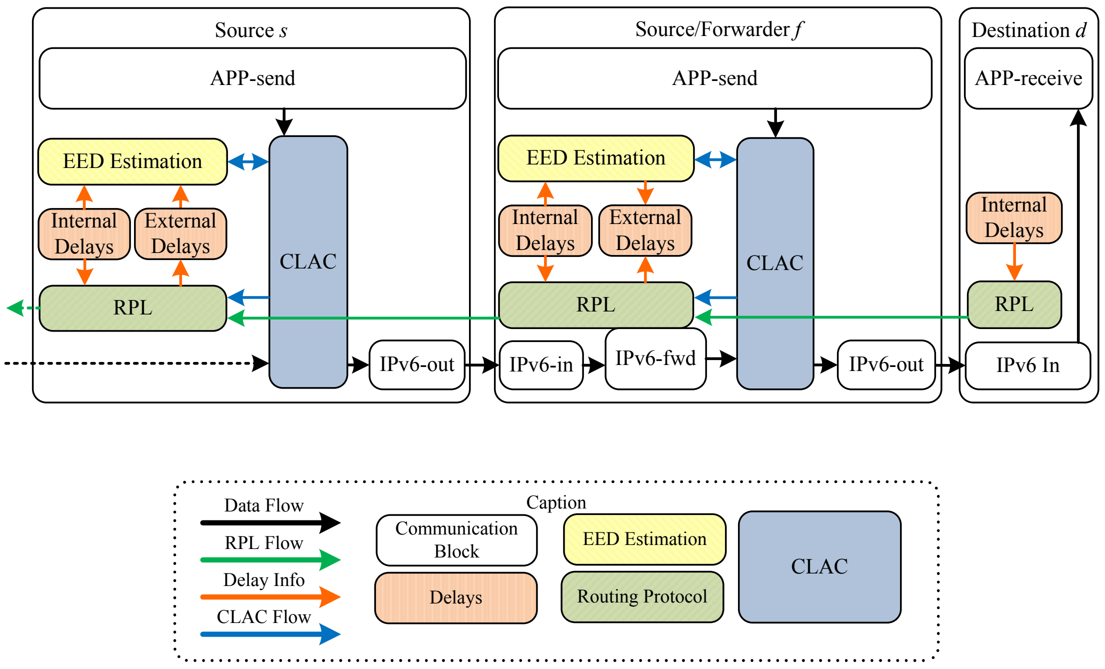


Figure 4.2: CLAC mechanism integration

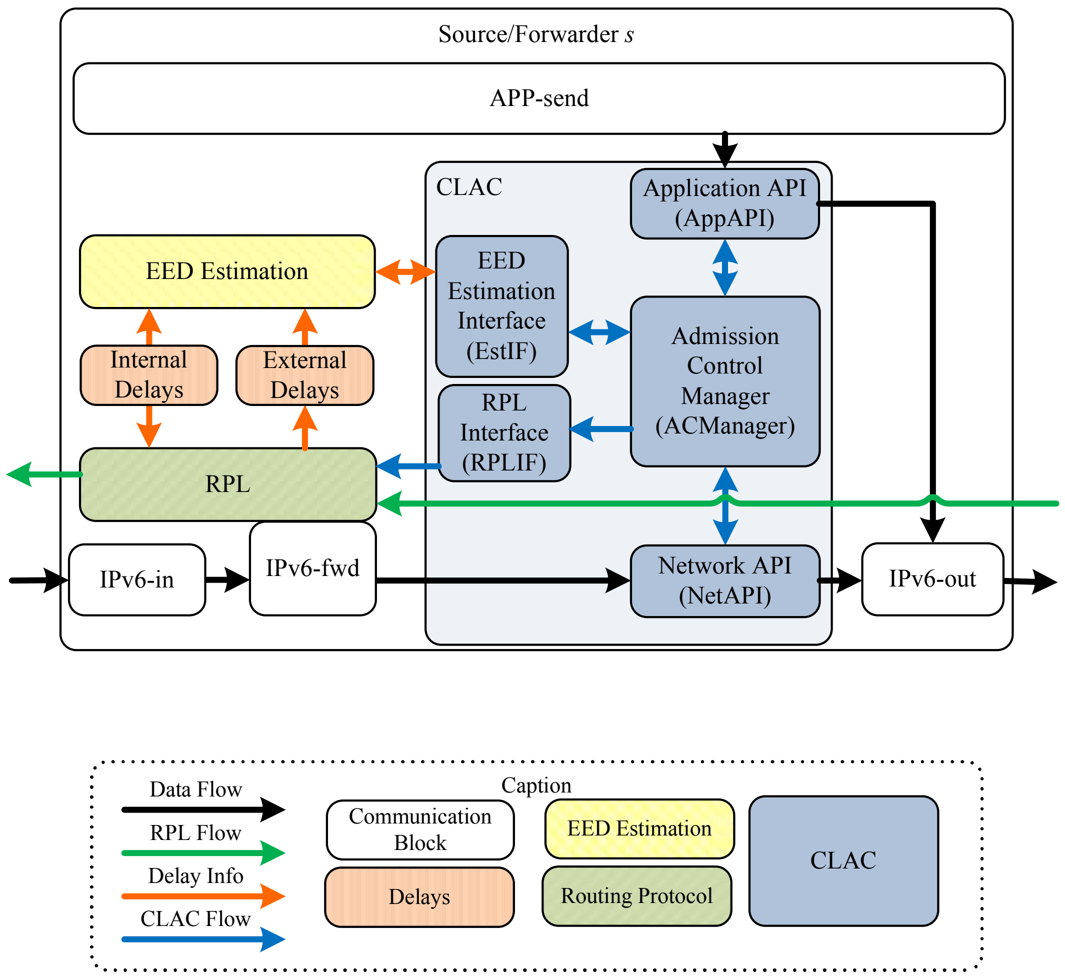


Figure 4.3: CLAC internal overview

Fig. 4.3 presents the internal building blocks of CLAC. In order for the CLAC mechanism to intercept and evaluate the usefulness of each packet, as earlier as possible, it defines two stages for packet interception: after the packet generation at the application layer, and when a packet is being forwarded at the network layer. Two Application Programming Interfaces (APIs) were defined for this purpose: the Application API (AppAPI) and the Network API (NetAPI). Additionally, two interfaces were also defined: the EED Estimation Interface (EstIF), and the RPL Interface (RPLIF). EstIF handles the delay estimation requests issued to the EED estimation mechanism. The RPLIF is used to trigger the sending of RPL control messages when necessary. The core block, named Admission Control Manager (ACManager), receives requests from AppAPI and NetAPI and, according to the caller API, issues requests for delay estimations to the EstIF. Then, the ACManager provides a decision (accept or drop) regarding the progress of a packet and communicates this decision back to the caller API (AppAPI or NetAPI). If the caller API is the NetAPI, the ACManager can also request the sending of RPL control messages to the RPLIF.

In order to support the operation of CLAC, data packets must have a payload format that contains a set of fields that can either be generated by the application or added later by a middle layer. In the adopted scenario, it is assumed that the data packets are generated with the fields shown in Fig. 4.4 a). The data packet payload includes a Source ID (SrcID), a Destination ID (DstID), an Application ID (AppID), a Sequence Number (SeqNr) which enables the per packet EED registering, and a MaxEED which indicates the maximum amount of delay allowed by the application. The MaxEED is defined when the packet is generated and updated while in progress to its destination. In each data packet interception performed at the application layer or at the network layer, the data packet payload (see Fig. 4.4 a)) is mapped to an internal data structure representing the packet according to Fig. 4.4 b), where Estimated EED (EstEED) value represents the $\widehat{\text{EED}}$. For each instance of such data structure, the ACManager requests an EstEED to the EstIF. The returned value will be stored in EstEED field and it will be evaluated by the ACManager which, in turn, decides whether the packet is to be accepted or to be dropped, and stores a value of 1 or 0, respectively, in the Decision field. The EstEED and the Decision fields added to the packet payload are saved in the internal data structure and, according to the Decision field returned by the ACManager, the APIs will map the internal data structure into the packet payload again and allow the data packet to proceed, or not, according to the decision value.

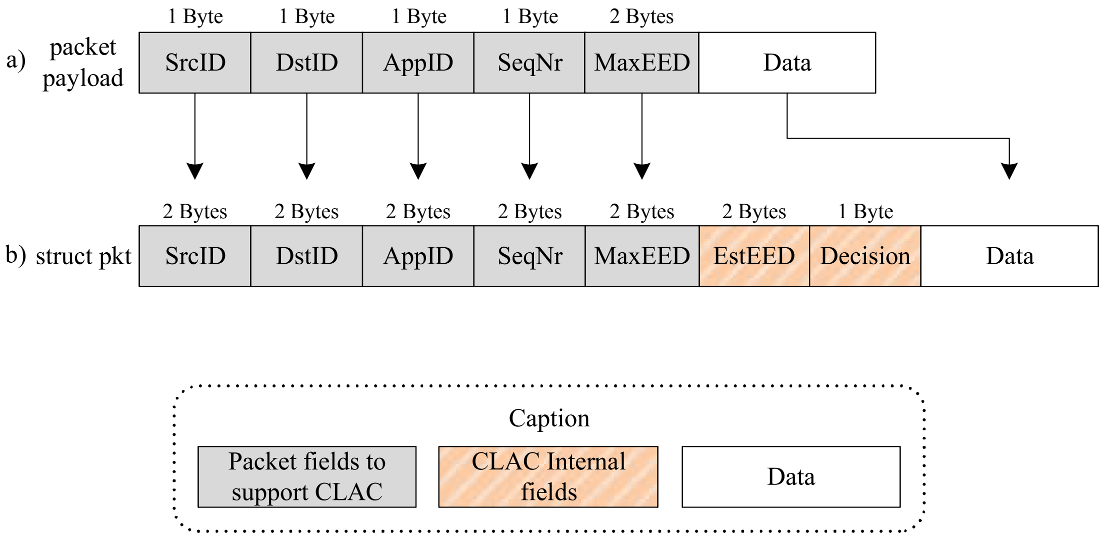


Figure 4.4: Data packet payload mapped into the internal packet struct (CLAC)

A flow diagram of the detailed interaction between the application layer and the CLAC mechanism is shown in Fig. 4.5. The application layer requests a decision to the AppAPI where the packet is mapped into the internal data structure of CLAC. The ACManager requests an EstEED to the EstIF and stores it in the EstEED field. These requests may have a L5 argument, if one wants the EstEED from the Layer 5 up to destination application, or it may have a L3 argument, if the EstEED wanted is the one from Layer 3. The ACManager checks if EstEED is smaller than the value defined for MaxEED. If EstEED does not meet this condition, the ACManager writes a zero in the Decision field and the packet is later dropped by the AppAPI. If the EstEED is below the defined MaxEED, the MaxEED is updated by subtracting the GenD (obtaining using Eq. 3.2 in Section 3.1.1) of the current node, and a one is written in the Decision field. The AppAPI will then return the packet towards its destination.

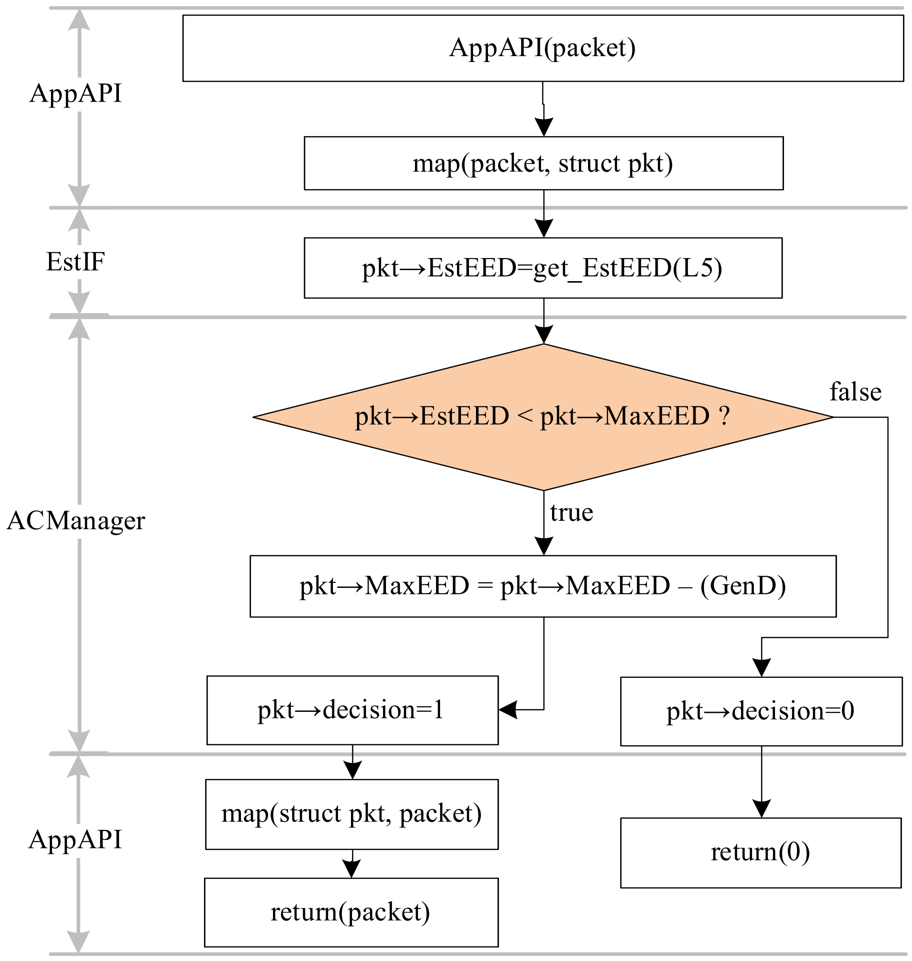


Figure 4.5: CLAC interaction with the application layer (using AppAPI)

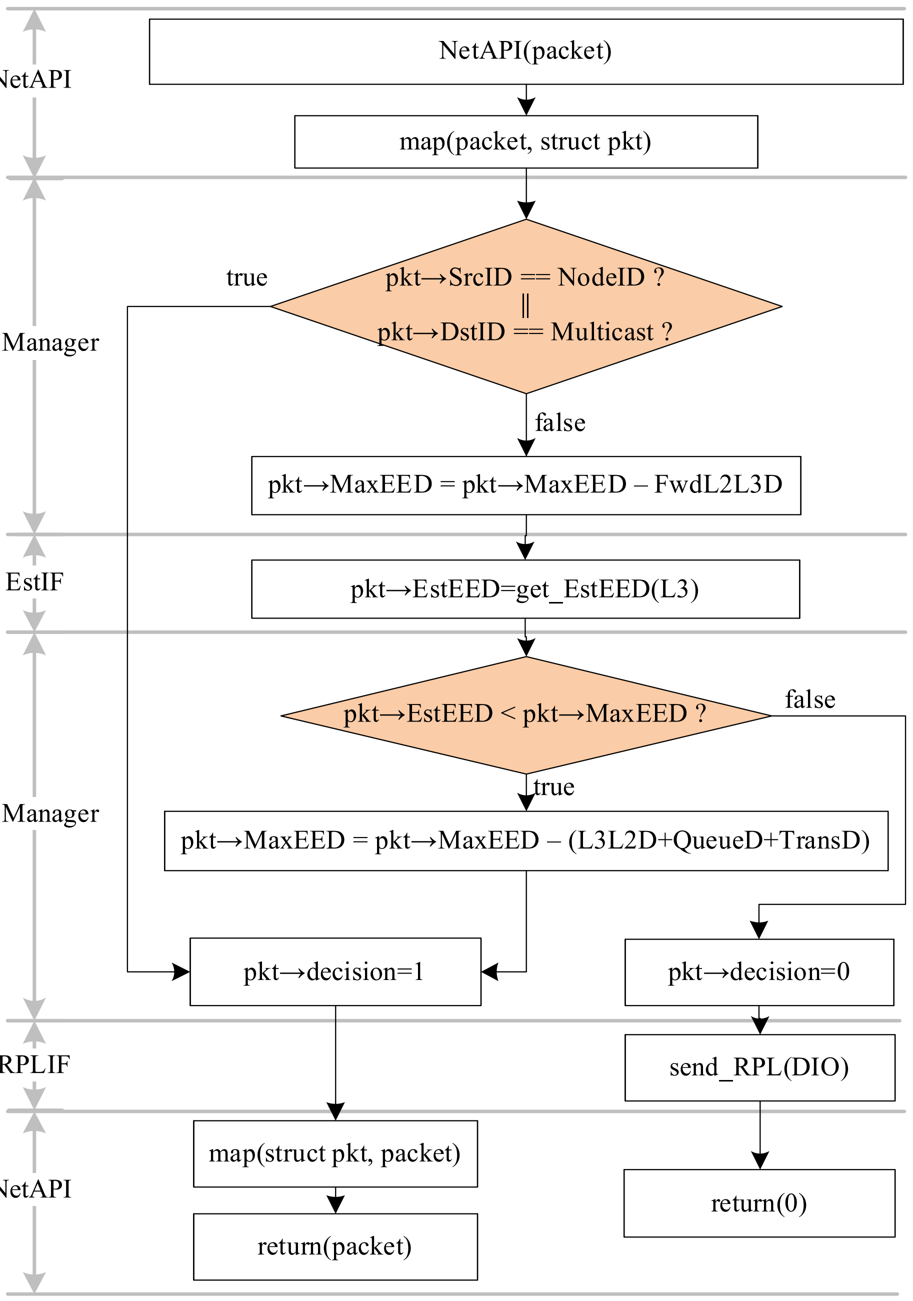


Figure 4.6: CLAC interaction with the network layer (using NetAPI)

Fig. 4.6 presents a flow diagram of the interactions within the CLAC mechanism when a packet is forwarded at the network layer. The NetAPI maps the packet into an internal data structure and then the ACManager checks if the packet has been generated in the current node or if the destination address is a multicast address. If true, the Decision field is set to one and later, NetAPI will map the data structure back into the packet to be forwarded. If not, the ACManager updates the MaxEED by subtracting the FwdL2L3D value. The ACManager then requests an EstEED to EstIF and checks if the returned value is smaller than the current MaxEED. If true, the MaxEED value is updated again by subtracting the values L3L2D, QueueD and TransD. The ACManager sets Decision field to one and NetAPI will map the data structure to the packet in order to be forwarded towards destination. If false, the ACManager sets the Decision field to 0 in order NetAPI to drop the packet. Right after the decision taken by the ACManager, the RPLIF is ordered to send an RPL DIO message to feedback delay information to the previous nodes (forwarders or generators), forcing them to update their delay estimation. Ideally, nodes would not have to discard any packets at the NetAPI, as they should all be discarded at the generation (the AppAPI).

4.2 Validation Environment

CLAC mechanism proposal was tested in the grid topology WSN shown in Fig. 1.3 with the simulation parameters presented in Table 4.1. The simulated scenario consists of 16 source/forwarder nodes placed within a distance of 25 m from each other plus a destination node, deployed in a WSN area of 100 m^2. Each node was simulated as a Tmote Sky [88], with a transmission range of 30 m and an interference range of 60 m, using the UDGM as physical channel model. The nodes ran the Contiki OS 2.5 [96] and were programmed to enable the debug of application and RPL messages. Extra code was inserted to implement the EEDEM estimation and CLAC mechanisms. The application layer uses UDP as transport layer and it generates packets of 100 Bytes in a CBR by using constant IGIs. The simulations were configured to stop when every source has sent 100 packets and were repeated 10 times using different seeds.

The accuracy of the EED estimation was assessed and CLAC was evaluated regarding a set of network performance items and energy savings.

In order to assess the EED estimation accuracy, the $\widehat{\text{EED}}$ for each packet was collected and later compared with the EED of that packet. When the simulation ended, the estimation accuracy was evaluated using the EEDError for a set of #RcvdPkts samples obtained using the MAPE (according to Eq. 2.19) and using SMAPE (according to Eq. 2.20). The average for EEDError$_{\text{(MAPE)}}$ and EEDError$_{\text{(SMAPE)}}$ were obtained, as well as their respective confidence intervals of 90%.

Table 4.1: Simulation Parameters

Parameter	**Value**
Number of nodes	16 + sink node
Deployment area	100 m x 100 m
Transmission range	30 m
Interference range	60 m
Channel	Unit Disk Graph Medium
Packet size	100 Bytes
# of sent packets per node	100 packets
Total # of sent packets (#SentPkts)	1600 packets
Transport/Application	UDP/CBR

CLAC's network performance was evaluated regarding the following items: average EED delay, PRR, in-profile versus out-of-profile packets, and PUR.

The Average EED was obtained using the EED for each packet and, at the end of the simulation, using the following equation:

$$\text{Average EED } (ms) = \frac{1}{\#\text{RcvdPkts}} \sum_{p=1}^{\#\text{RcvdPkts}} \text{EED}_p \tag{4.5}$$

The PRR was obtained using the following equation:

$$\text{PRR } (\%) = \frac{\#\text{RcvdPkts}}{\#\text{SentPkts}} (\times 100) \tag{4.6}$$

It was defined In-profile Packet Ratio (IPR) and Out-of-profile Packet Ratio (OPR), respectively obtained using number of in-profile packets and out-of-profile packets, per #SentPkts (constant in all cases), according to Eq. 4.7 and Eq. 4.8.

$$\text{In-profile Packet Ratio (IPR) } (\%) = \frac{\text{Number of in-profile packets}}{\#\text{SentPkts}} (\times 100) \tag{4.7}$$

$$\text{Out-of-profile Packet Ratio (OPR) } (\%) = \frac{\text{Number of out-of-profile packets}}{\#\text{SentPkts}} (\times 100) \tag{4.8}$$

The PUR was obtained using the following equation.

$$\text{Packet Usefulness Ratio (PUR) } (\%) = \frac{\text{Number of in-profile packets}}{\#\text{RcvdPkts}} (\times 100) \tag{4.9}$$

The average values of EED, PRR, IPR, OPR, and PUR, were obtained for each round of simulations.

CLAC was evaluated also for energy savings. The energy consumed by a device in Joules was obtained using the following equation:

$$\text{Energy (Joules)} = \text{Power (Watts)} \times \text{time (s)} \tag{4.10}$$

Since a Tmote Sky device was used, three different power constants were defined using the values shown in Table 4.2 (obtained from the TMote Sky datasheet [88]).

Table 4.2: Defined Power Constants

Operating Conditions	**Voltage (V)**	**Current Nom. (mA)**	**Power Constant**	
			Name	**Value (mW)**
MCU on Radio RX	3	21.8	Power_{Rx}	65.4
MCU on Radio TX	3	19.5	Power_{Tx}	58.5
MCU on Radio off	3	0.18	Power_{MCUon}	0.54

The Powertracker plugin for Cooja was used to collect the time in milliseconds that each node n was in the monitored state ($\text{Time}^n_{Monitored}$), the time it was in the *on* state (Time^n_{On}), and the time it was either transmitting (Time^n_{Tx}), receiving (Time^n_{Rx}) or interfered (Time^n_{Int}). Using these times and the power constants defined in Table 4.2, three types of energy components were calculated per each node n: the reception component (Energy^n_{Rx}), the transmission component (Energy^n_{Tx}), and the MCU *on* component (Energy^n_{MCUon}), respectively obtained using Eqs. 4.11, 4.12, and 4.13.

$$\text{Energy}^n_{Rx} = \text{Power}_{Rx} \times (\text{Time}^n_{Rx} + \text{Time}^n_{Int}) \tag{4.11}$$

$$\text{Energy}^n_{Tx} = \text{Power}_{Tx} \times \text{Time}^n_{Tx} \tag{4.12}$$

$$\text{Energy}^n_{MCUon} = \text{Power}_{MCUon} \times (\text{Time}^n_{Monitored} - \text{Time}^n_{On}) \tag{4.13}$$

The total energy spent for the Number of Nodes (#Nodes) used per simulation was calculated as follows:

$$\text{Total Energy (Joules)} = \sum_{n=1}^{\#\text{Nodes}} (\text{Energy}_{Rx}^{n} + \text{Energy}_{Tx}^{n} + \text{Energy}_{MCUon}^{n}) \qquad (4.14)$$

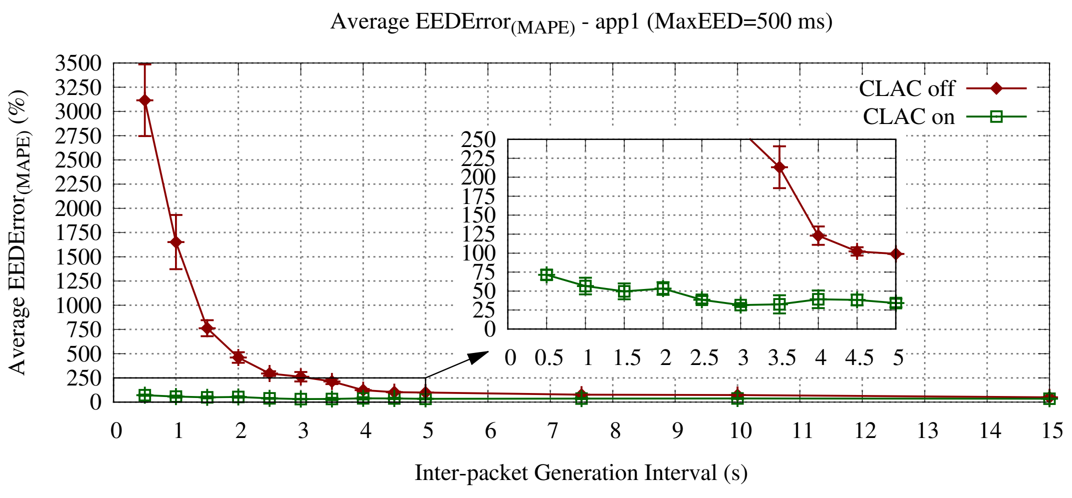
In order to test the CLAC performance for different network loads, three types of applications were defined with different MaxEED. The application 1 (*app1*) was defined with a MaxEED of 500 *m*s, the application 2 (*app2*) with a MaxEED of 1000 *m*s and the application 3 (*app3*) with a MaxEED of 2000 *m*s. Each of these applications was tested separately.

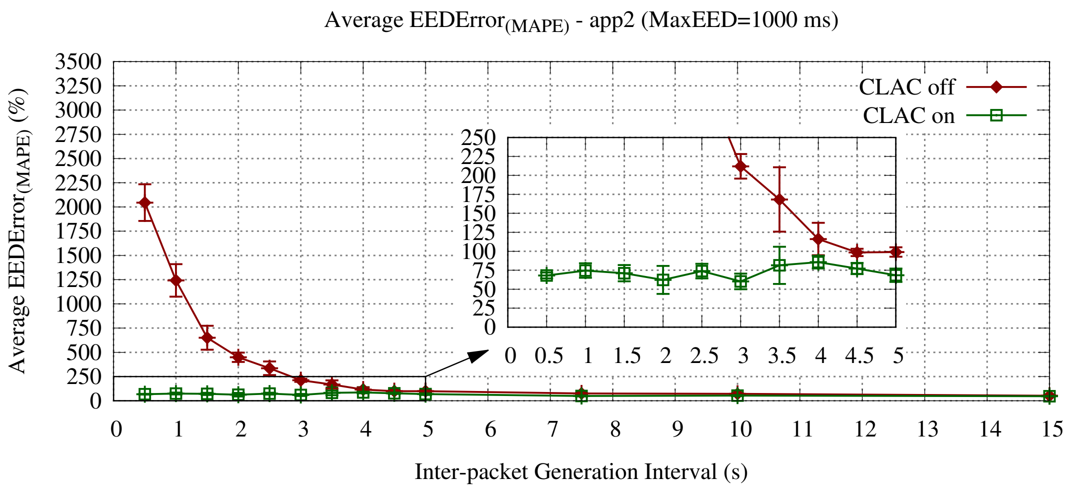
4.3 Results

Fig. 4.7 presents the $\text{EEDError}_{(\text{MAPE})}$ for applications *app1*, *app2*, and *app3*. The results show that, for high network loads (lower IGIs), the $\text{EEDError}_{(\text{MAPE})}$ when using *app1* with CLAC *off* is higher than that obtained with CLAC *on* (between 1000% and 1700%). With CLAC *on*, the $\text{EEDError}_{(\text{MAPE})}$ is reduced to values below 100%.

Fig. 4.8 shows the average EED with CLAC *on* and with CLAC *off* for all applications (*app1*, *app2*, and *app3*), as well as their standard deviations. The left side graphics show the results obtained for IGIs up to 15 s and, the right side ones, show the same results but only up to 5 s. These results show that, for IGIs below 5 s the average EED is lower with CLAC *on*. For IGIs up to 4 s, the average EED reaches approximately 23000 *m*s with CLAC *off*. The results show that, for IGIs higher than 2 s the average EED with CLAC *on* is lower than the MaxEED defined by each application. For IGIs below 2 s with CLAC *on*, the average values are above the MaxEED defined by each application, but still lower than the ones obtained with CLAC *off*.

Fig. 4.9 presents the average PRR for applications *app1*, *app2*, and *app3* and for IGIs up to 15 s. The results show that the average PRR with CLAC *on* is always smaller than the one obtained with CLAC *off*. The difference of results between CLAC *on* and CLAC *off* is larger for *app1*, shorter for *app2* and *app3*, and it is more pronounced when the network load is high (lower IGI values).





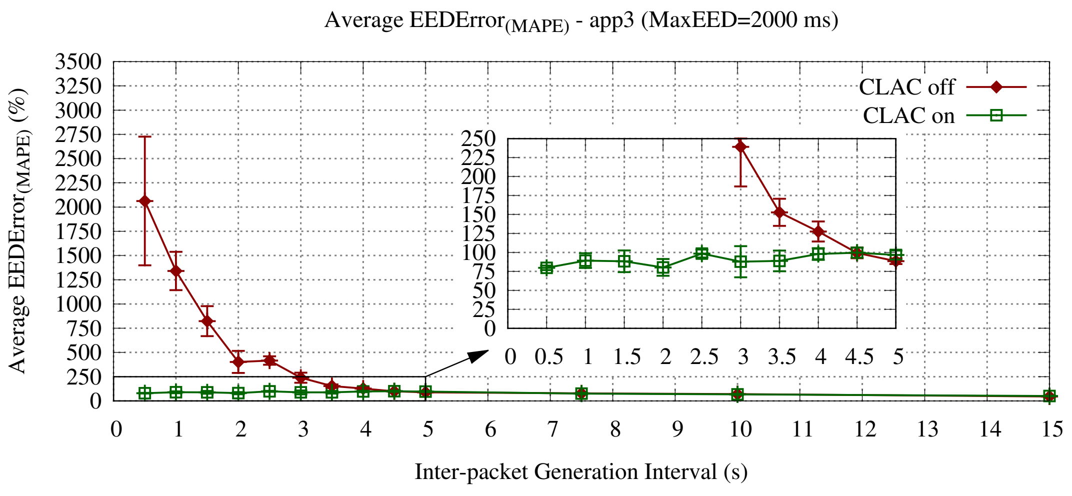


Figure 4.7: Average EEDError$_{(MAPE)}$ (CLAC)

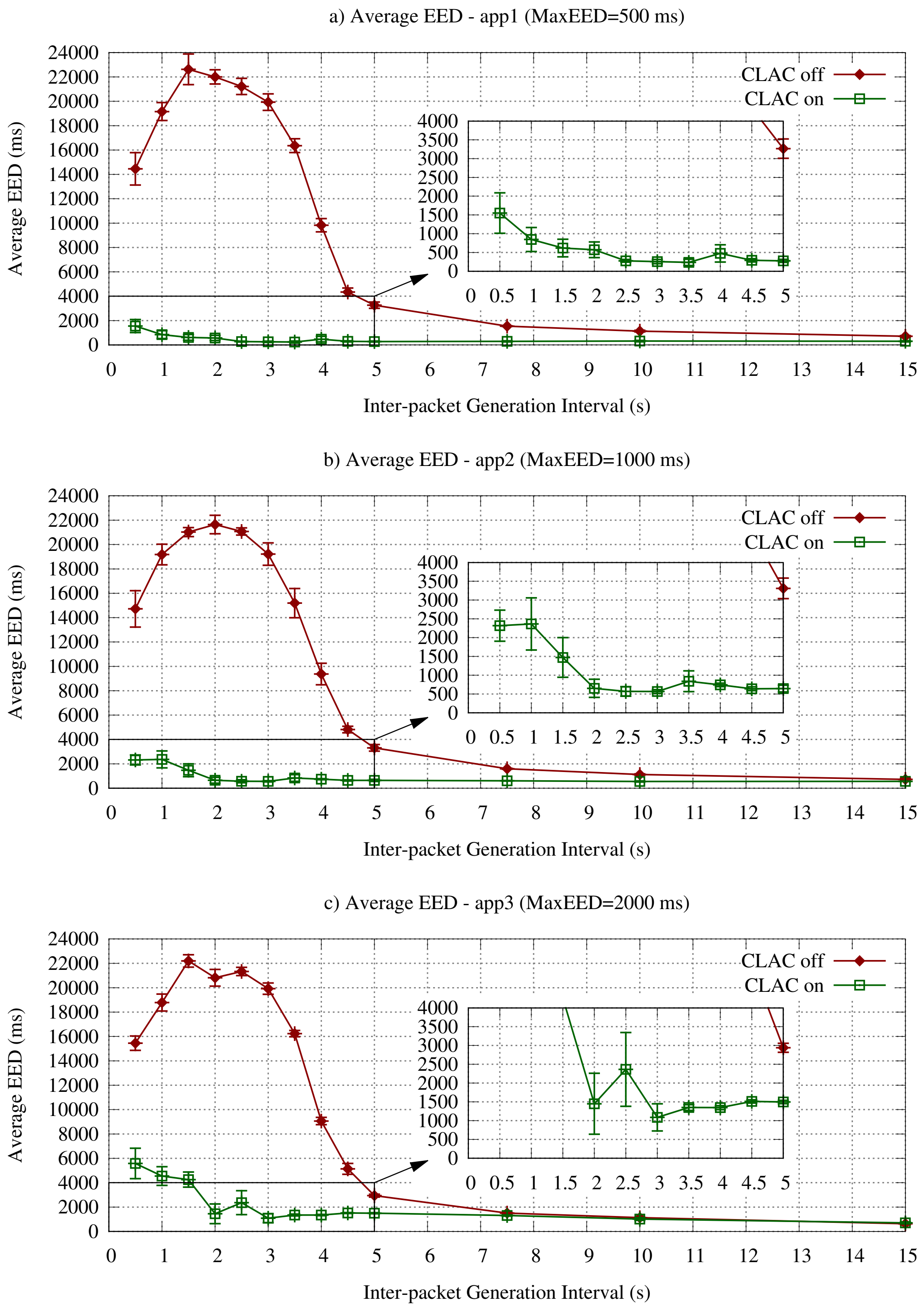
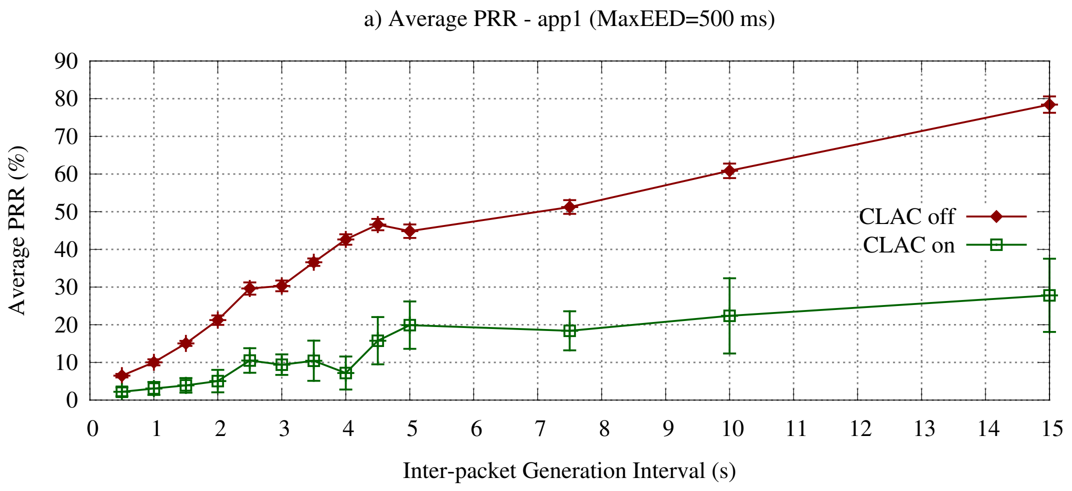
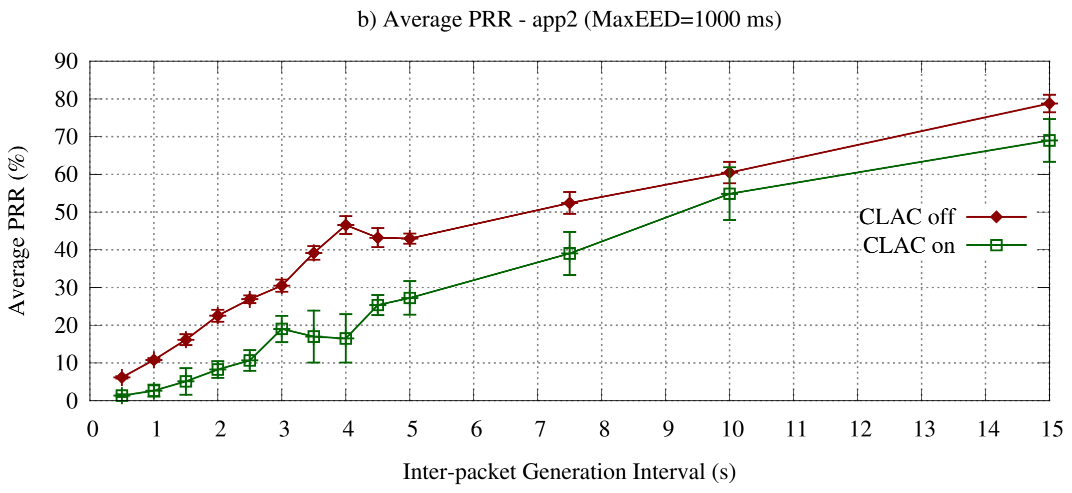


Figure 4.8: Average End-to-End Delay (CLAC)





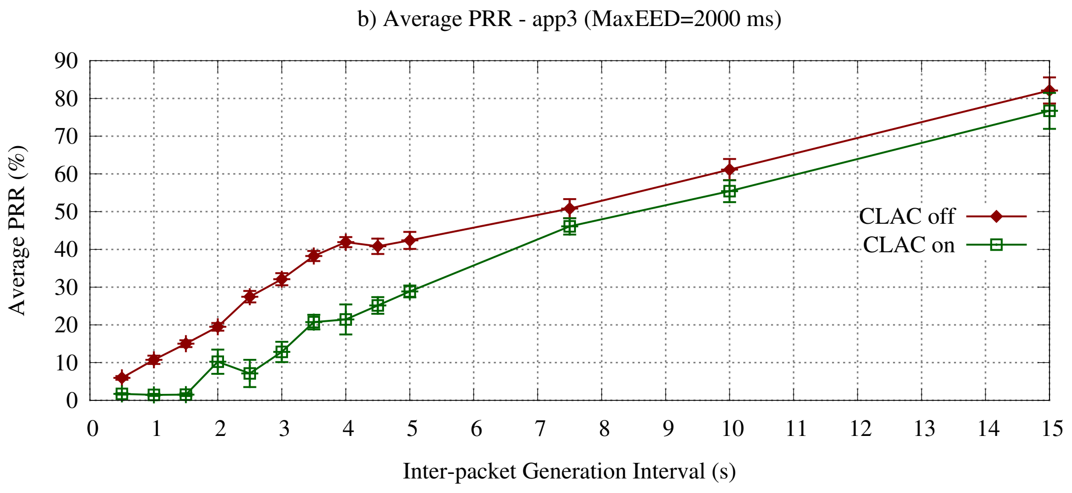


Figure 4.9: Average Packet Reception Ratio (CLAC)

Fig. 4.10 presents the IPR (left) and OPR (right) with CLAC *on* and *off*, for the three applications, obtained for IGIs up to 15 s. The results show that for all applications, the IPR with CLAC *on* is higher than the IPR with CLAC *off*, except in the case IGI equals to 15 s where, nonetheless, using *app2* and *app3* their standard deviation intervals overlap. For IGIs up to 5 s, for all applications the difference between CLAC *on* and CLAC *off* is about 10 pp. Regarding OPR, for all applications, the results show lower values with CLAC *on* than those obtained with CLAC *off*. For IGIs up to 5 s, the OPR with CLAC *off* increases up to 40% while with CLAC *off* the OPR values are below 10%. In the worst case, for IGIs of 4 s and using *app1* and *app2*, the OPR with CLAC *off* is 45% while with CLAC *on* is less than 5%. For all IGIs and applications, with CLAC *on*, the IPR is always higher than the OPR. This is not true with CLAC *off*; using *app1* the OPR never overpasses IPR, using *app2* the IPR overpasses the OPR for the IGI of 10 s, and using *app3* the same is verified for IGI of 7.5 s.

Fig. 4.11 presents the PUR using applications *app1*, *app2*, and *app3*, for IGIs ranging from 1 to 15 s. The results show that the PUR is higher with CLAC *on* for both high and low network loads, with values ranging approximately from 75% to 90%, whenever the IGI is above 1 s. The difference of the PUR between when CLAC is *on* and *off* is higher for high network loads and, in case of low network loads it depends on the application (in *app1* the difference is about 35 pp, for *app2* is about 15 pp, and for *app3* the difference is residual). These results show that network performance is improved when CLAC is turned *on*, mainly in low network loads where the scenario with CLAC *off* presents lower network performance.

Fig. 4.12 presents the total energy consumed per simulation, where the number of sent packet is constant for all simulations, using the three applications with IGIs up to 15 s. The results show that the energy spent with CLAC *on* is lower than the energy spent with CLAC *off*. The difference is higher in *app1* when compared to *app2* and *app3*. A greater impact on the spent energy is obtained for lower MaxEED values. A combined analysis of the results shown in Figs. 4.10, 4.11 and 4.12 leads to the conclusion that, when CLAC is *on*, a higher number of in-profile packets and less number of out-of-profile packets are measured and this is done while using less energy than when CLAC is *off*.

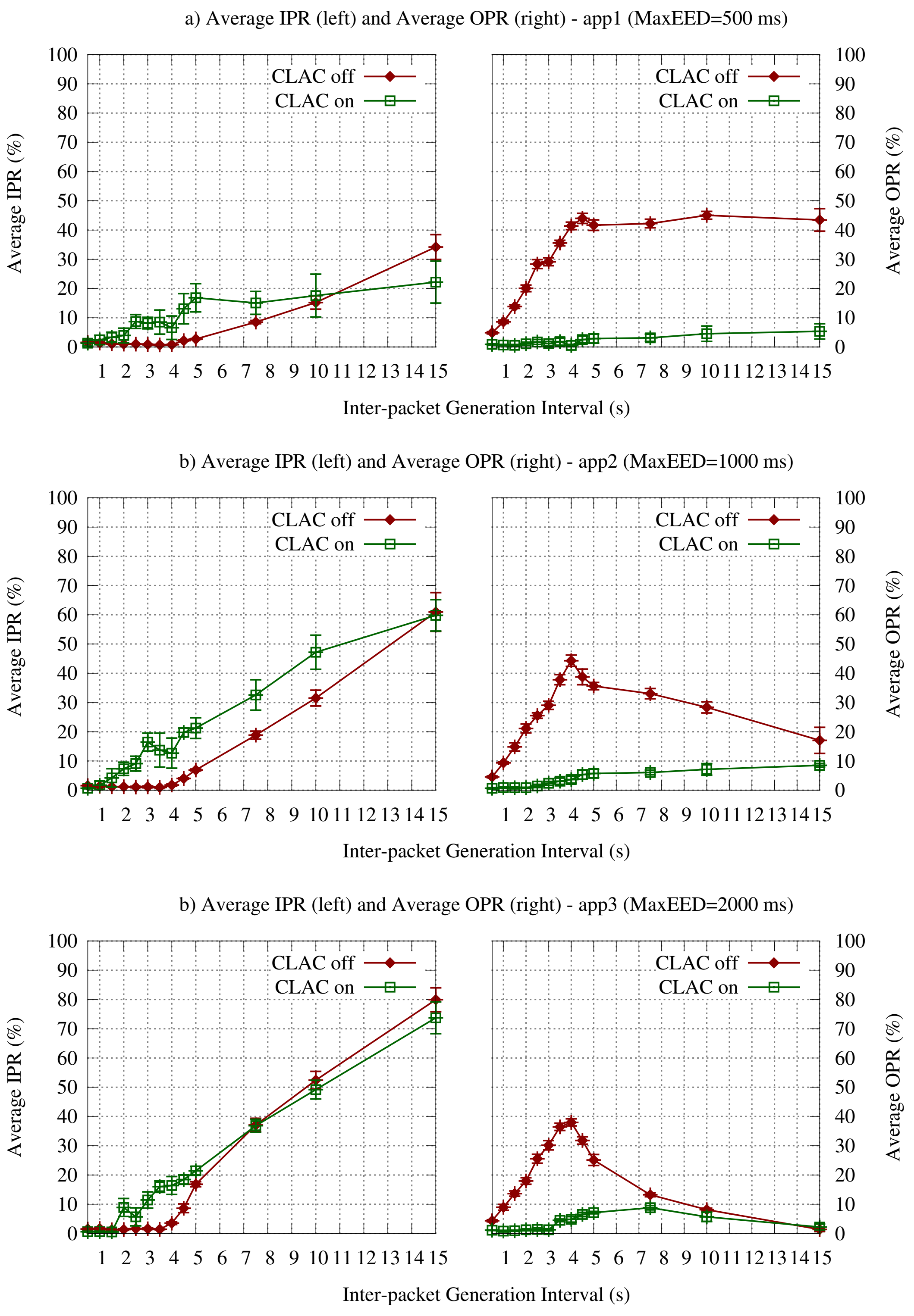
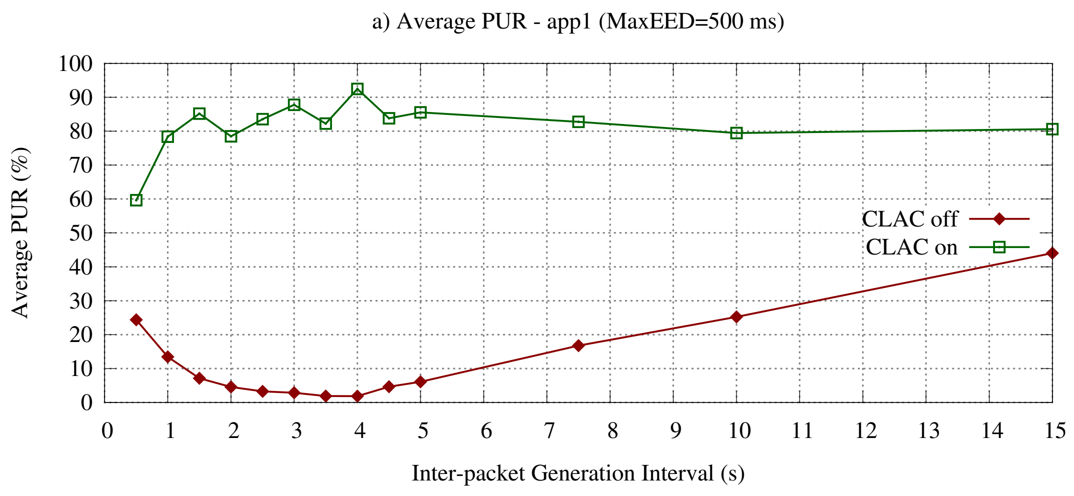
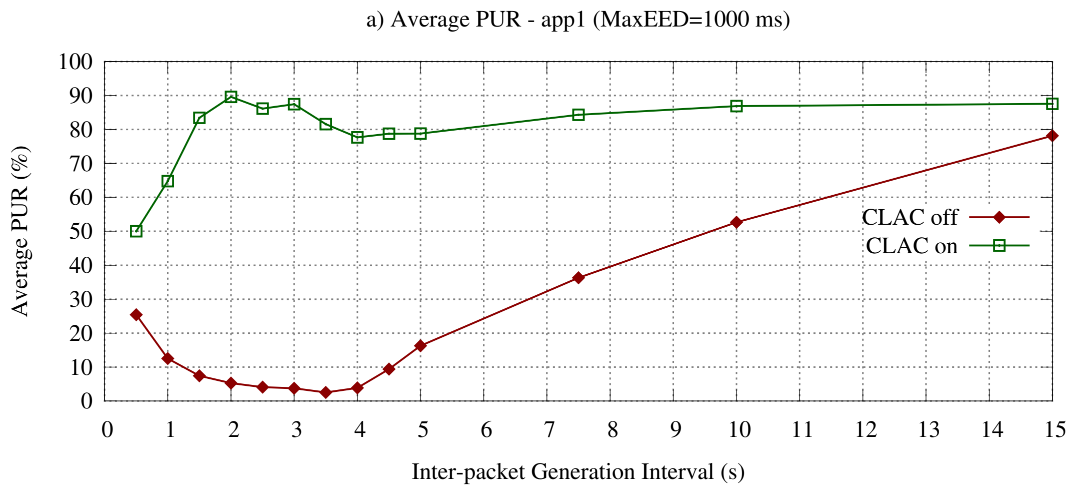


Figure 4.10: Average In-profile Packet Ratio and Average Out-of-profile Packet Ratio (CLAC)





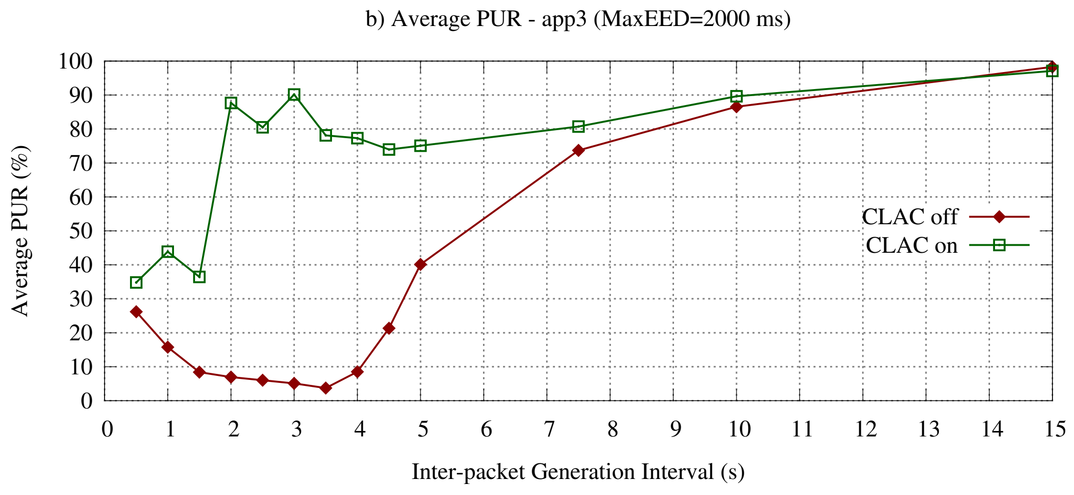


Figure 4.11: Average Packet Usefulness Ratio (CLAC)

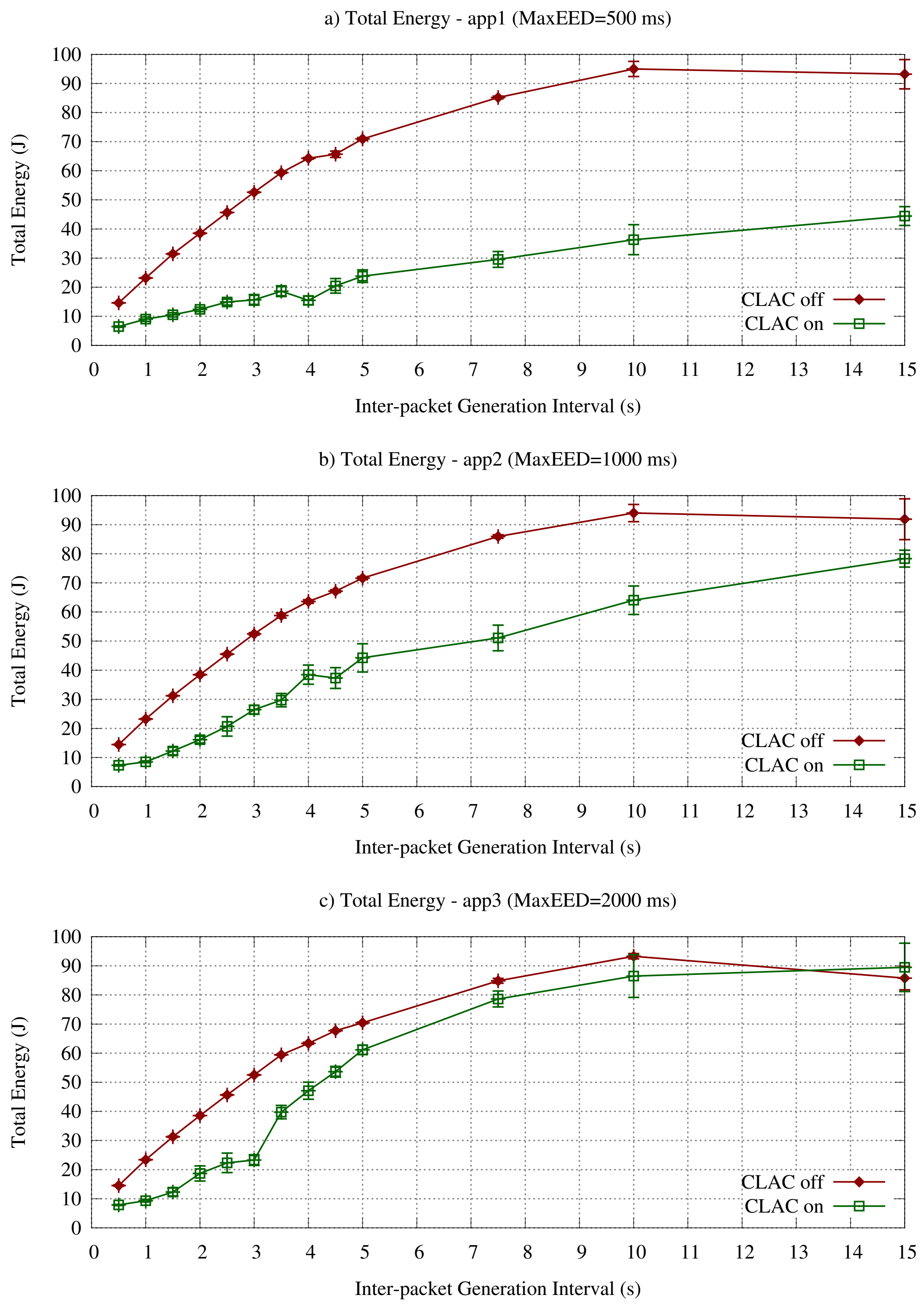


Figure 4.12: Total Energy consumed (CLAC)

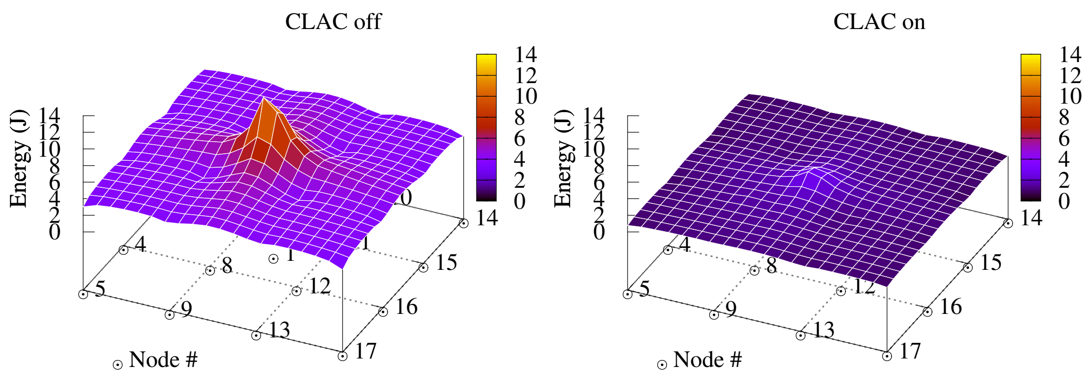
Fig. 4.13 shows the energy consumed by each node, for the three applications and for IGIs up to 15 s. Node 1 (central point of the graphic) consumes the most energy due to being the destination node, constantly receiving data packets from the others nodes and sending routing packets in reverse direction at a high rate. Not considering node 1, the nodes that use more energy are those closer to the destination (node 7, 8, 11, and 12) and can be identified by their position in the grid topology shown in Fig. 1.3. The results also show that the energy savings for all nodes are greater with CLAC *on*, when comparing with the ones obtained with CLAC *off*.

The Fig. 4.14 presents the number of in-profile packets in each node mapped on the grid topology, for the three applications and for an IGI of 5 s. The results show that with CLAC *on* the number of in-profile packets is higher than those obtained with CLAC *off*. This effect is mainly verified on the nodes closer to the destination (central point) as their packets have a lower EED and can meet the imposed deadline. With CLAC *off* and if the application demands a lower MaxEED, almost no in-profile packets are obtained. A combined analysis of Figs. 4.13 and 4.14 leads to the conclusion that CLAC improves performance in low network loads and, at the same time, enables energy savings.

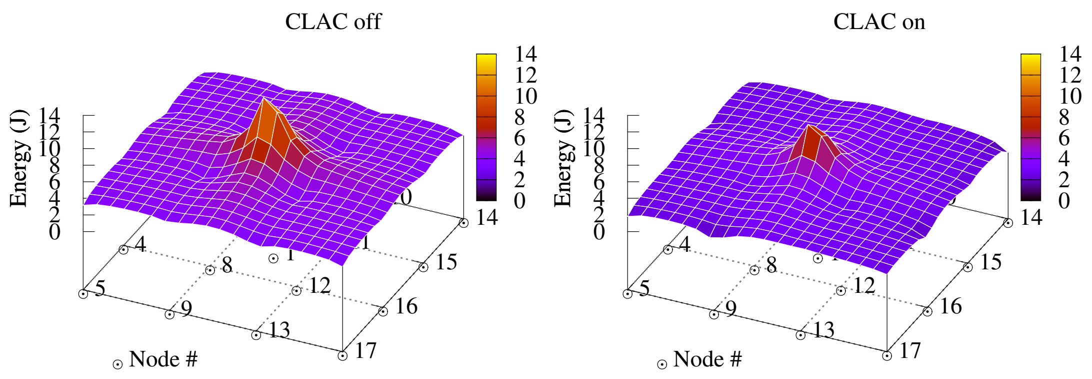
4.4 Summary

This chapter presented the CLAC mechanism that was designed to enhance network performance and to increase the energy efficiency of a WSN, in a grid topology, by avoiding the transmission of potentially useless packets. The CLAC mechanism uses the EEDEM to preview packets EED and performs a decision to accept or drop each packet if it is expected to comply or miss the EED deadline previously defined by the application. The CLAC mechanism was tested using different network loads and the results show that the CLAC enhances the overall network performance by decreasing the number of useless packets and, consequently, increasing the number of useful packets. As a side effect, the CLAC mechanism also improves the WSN energy efficiency, particularly in high network loads.

a) Energy consumed mapped in each node - app1 (MaxEED=500 ms IGI=5 s)



b) Energy consumed mapped in each node - app2 (MaxEED=1000 ms IGI=5 s)



c) Energy consumed mapped in each node - app3 (MaxEED=2000 ms IGI=5 s)

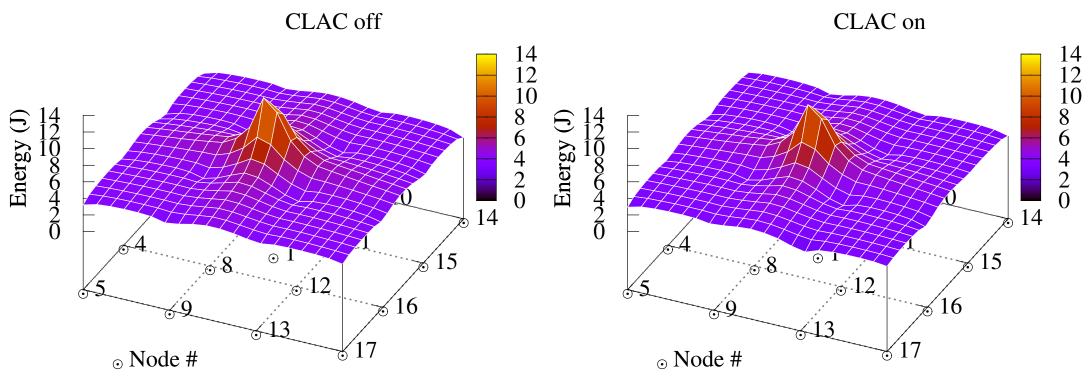
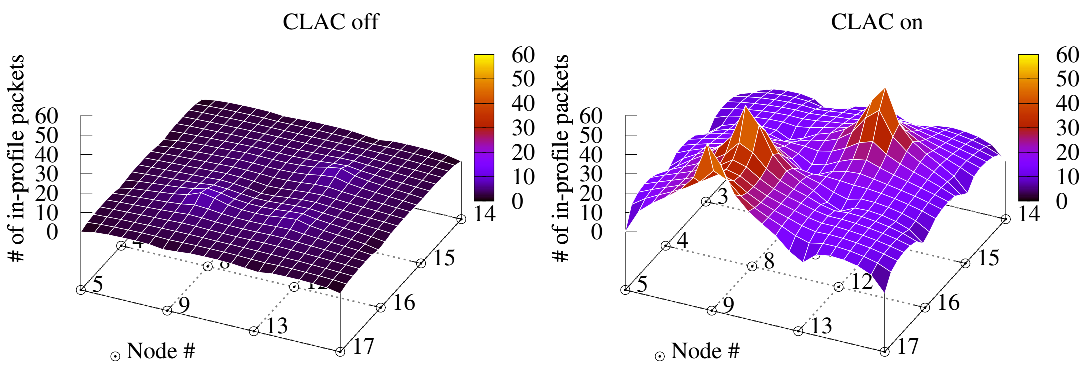
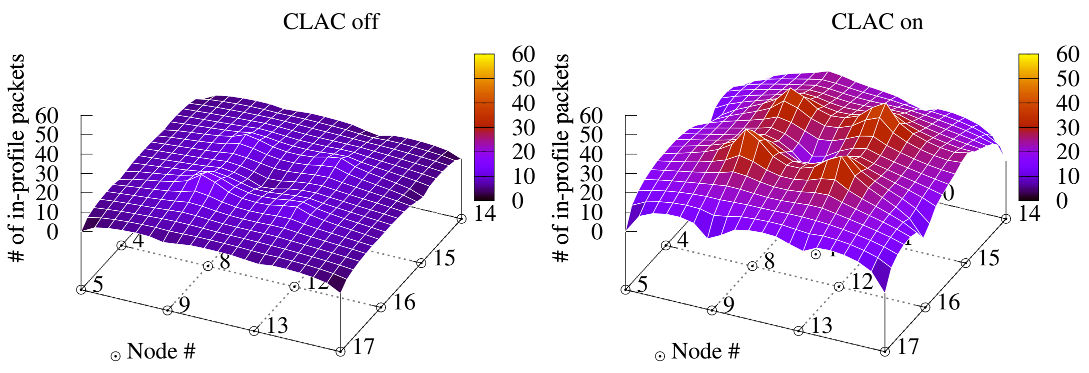


Figure 4.13: Energy consumed mapped in each node (CLAC)

a) Number of in-profile packets mapped in each node - app1 (MaxEED=500 ms IGI=5 s)



b) Number of in-profile packets mapped in each node - app2 (MaxEED=1000 ms IGI=5 s)



c) Number of in-profile packets mapped in each node - app3 (MaxEED=2000 ms IGI=5 s)

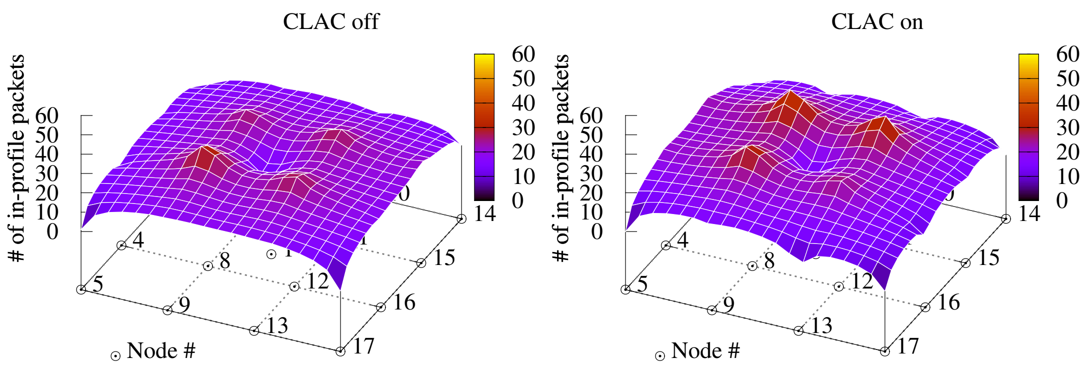


Figure 4.14: Number of in-profile packets mapped in each node (CLAC)

Chapter 5

Conclusion

The main goal of this thesis was to enhance the WSN support for real-time traffic by exploring the hypothesis that potential useless data packets should not be transmitted by the source node. This thesis considers the scenario of a solar smart grid, where each solar panel is equipped with a WSN node that generates real-time streams towards a sink. The real-time traffic generated by the WSN nodes demands a service characterized by parameters such as delay, packet loss, and throughput. In particular, the work focused on guaranteeing a maximum EED at the application layer for packets transported by the WSN. A packet is considered useful if delivered to the application layer of the destination node within the expected maximum EED.

5.1 Work Review

Chapter 2 summarized the state of the art on delay measurement and estimation. It starts by characterizing delays in IP networks and surveys the methods available for measuring delays in different network points. Chapter 2 continues, presenting the state of the art regarding AC mechanisms where special focus was given to distributed AC mechanisms. Then, the operation and constraints of sensor nodes regarding hardware, operative systems, and communications stacks were described.

Existing solutions for EED measurement and estimation do not provide real-time, per-packet delay estimation, with measurement information in the source node and using low overhead. The EEDEM, described in Chapter 3, was proposed to provide an accurate per-packet end-to-end delay estimation and, at the same time, to avoid negative impact on the network performance namely regarding average packet EED and PRR. To provide an EED estimate, EEDEM combines the delays suffered by previous packets with internal timers and RPL. As internals delays EEDEM accounts not only for transmission delays, but also considers processing delays which revealed to be important in WSNs where nodes have limited processing resources. Results showed that it produces an accurate EED estimation without having a significant impact on network performance in terms of EED and PRR values. In addiction, two improvements regarding EEDEM were implemented. EEDEM is highly dependent on the operation of the RPL and, in order to improve its estimation accuracy, RA-EEDEM was proposed. RA-EEDEM consists of a set of modifications to RPL aimed to improve the accuracy of EEDEM. When compared to EEDEM, RA-EEDEM improves the EED estimation, presents a better average PRR, and a smaller average EED for high network loads. Finally, DAOP was also proposed to improve EEDEM accuracy regarding different network loads. DAOP dynamically infers network load by monitoring node's MAC queue size and it applies the best smoothing factor to the estimation function. The results obtained showed that DAOP provides a more accurate EED estimation for different network loads than those obtained by EEDEM.

Chapter 4 proposed the CLAC mechanism that is a distributed per-packet admission control mechanism that inter operates with the EEDEM. CLAC decides if a packet should progress or be dropped, and intercepts packets, in a cross-layer mode, at either application's or network's layer; it uses an RPL interface to automatically adjust the accuracy of each node's decision. Results showed that CLAC enhances the overall network performance by discarding the useless packets and that it increases the number of useful packets. As a side effect, it also improves the overall WSN energy efficiency, particularly in high network loads.

5.2 Contributions Summary

This thesis provides two major original contributions:

- **A novel mechanism to estimate EED based on the RPL routing protocol**: a novel EED estimation mechanism was proposed in order to preview the EED of a packet from source to destination, at the application layer. Other delay estimation mechanisms are proposed in literature but some of them do not provide a real-time and per-packet delay estimation, while others introduce additional traffic in the WSN to provide estimations. The proposed EED estimation mechanism provides a real-time and per packet EED estimation using RPL packets to feedback the EED delay to the source nodes of the previously sent packets, thus avoiding extra traffic in the WSN. To enhance EED estimation accuracy this proposal accounts not only with transmission delays but also with the in-node processing delays which are relevant in limited processing sensor nodes. This contribution has been published in [2]. Furthermore, the accuracy of the EED estimation mechanism was improved by applying a set of modifications to RPL. This improvement proposal was published in [3]. Also, in the context of the EED estimation mechanism and in order to enhance EED estimation when using multiple network loads, a delay accounting optimization procedure was also proposed, and published in [4].

- **A novel cross-layer admission control mechanism based on the EED estimation**: in order to decide the transmission of the packets according to their usefulness to the destination application, a novel cross-layer packet AC mechanism, named CLAC is proposed. CLAC is a distributed mechanism to be deployed in WSN nodes, which is responsible for decision of sending or dropping a packet according to the requirements defined by the application. Other admission control mechanisms are proposed in literature but the novelty of the proposed mechanism is that it operates in a cross-layer operation, namely in application and network layers, and it implements interfaces with the EED estimation mechanism and RPL routing protocol. CLAC proposal was submitted and accepted for publication in [5].

5.3 Future Work

Future work related to this work may include the following topics:

- **Multiple real-time applications:** EEDEM and CLAC consider only one real-time application running on the WSN at the same time. Both mechanisms can be extended to support multiple real-time applications simultaneously. Per packet priorities can be defined according to the remaining time each packet has before its associated deadline. An additional distributed packet scheduling mechanism based on the packet priorities defined can also be implemented and used to enhance the network support for these applications.

- **Data aggregation:** Data aggregation is a common topic in WSNs as sensors tend to read and transmit the same or similar information multiples times. Data aggregation can then be applied to reduce redundant transmissions and save energy. Such transmissions may occur in different time instants hindering the possibility of data aggregation. If a packet can wait more time within a node for other packets, the aggregation ratio may increase. Using the estimation mechanism here proposed, it can be estimated a maximum delay that any packet can suffer in multiple network points and thus, the this proposal can be extended to cooperate with a data aggregation mechanism in order to improve the data aggregation. Further research efforts can be used to identify the best trade off between minimizing processing delay and maximizing the aggregation ratio.

- **Feedback information:** RPL was used to feedback the delays estimated by EEDEM to the source nodes. Other techniques may be used and should be evaluated as alternatives. The use of ACK packets at the L2 using stop and wait ARQ procedure or at data plane may be a viable alternative. Hybrid operation, combining multiple approaches may also be studied.

- **Low power IEEE 802.11:** Wi-Fi networks are widely available and, in most cases, already interconnected to the Internet. Although the Wi-Fi power consumption can be an issue for WSN, the latest IEEE 802.11ah Low power Wi-Fi standard addresses this issue. Therefore, this low-power Wi-Fi standard may be emerging as an alternate to IEEE 802.15.4 regarding the IoT. The deployment of both EEDEM and CLAC in this new standard arises as another interesting research topic.

References

[1] “SELF-PVP Project,” Accessed: 17-Jul-2015. [Online]. Available: http://www.cmuportugal.org/tiercontent.aspx?id=3374

[2] Pedro Pinto, António Pinto and Manuel Ricardo, “End-to-end Delay Estimation using RPL Metrics in WSN,” in *Wireless Days (WD), 2013 IFIP*, Nov. 2013, pp. 1–6.

[3] Pedro Pinto, António Pinto and Manuel Ricardo, “RPL Modifications to Improve the End-to-end Delay Estimation in WSN,” in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, Aug. 2014, pp. 868–872.

[4] Pedro Pinto, António Pinto and Manuel Ricardo, “Delay Accounting Optimization Procedure to Enhance End-to-End Delay Estimation in WSNs,” in *8th International Wireless Internet Conference (WICON 2014) - Symposium on Wireless and Vehicular Communication*, Lisbon, 2014.

[5] Pedro Pinto, António Pinto and Manuel Ricardo, “Cross-Layer Admission Control to Enhance the Support of Real-Time Applications in WSN,” *Sensors Journal, IEEE*, vol. 15, no. 12, pp. 6945–6953, Dec. 2015.

[6] Jean-Chrysostome Bolot, “Characterizing end-to-end packet delay and loss in the internet,” *Journal of High Speed Networks*, vol. 2, pp. 305–323, 1993.

[7] C. J. Bovy, H. T. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, “Analysis of End-to-End Delay Measurements in Internet,” in *ACM Conference on Passive and Active Measurements (PAM)*, fort Collins, Colorado, USA, 2002.

[8] A. Mohammad, X. Hong, M. Islam, and K. Zunnurhain, “Delay analysis of Wireless Ad Hoc networks: Single vs. multiple radio,” in *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, Oct. 2010, pp. 814–820.

[9] J. Li, Z. Li, and P. Mohapatra, “Adaptive Per Hop Differentiation for End-to-end Delay Assurance in Multihop Wireless Networks,” *Ad Hoc Netw.*, vol. 7, no. 6, pp. 1169–1182, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2008.10.005

[10] B. Staehle, D. Staehle, R. Pries, M. Hirth, P. Dely, and A. Kassler, “Measuring One-way Delay in Wireless Mesh Networks: An Experimental Investigation,” in *Proceedings of the 4th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous*

Wireless and Wired Networks, ser. PM2HW2N '09. New York, NY, USA: ACM, 2009, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/1641913.1641918

[11] L. Kleinrock, *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[12] P. Svoboda, M. Laner, J. Fabini, M. Rupp, and F. Ricciato, "Packet delay measurements in reactive IP networks," *IEEE Instrumentation Measurement Magazine*, vol. 15, no. 6, pp. 36–44, Dec. 2012.

[13] V. Paxson, "On Calibrating Measurements of Packet Transit Times," in *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '98/PERFORMANCE '98. New York, NY, USA: ACM, 1998, pp. 11–21. [Online]. Available: http://doi.acm.org/10.1145/277851.277865

[14] S. Keshav, "Packet-Pair Flow Control," *IEEE/ACM Transactions on Networking*, 1994.

[15] M. Coates and R. Nowak, "Network tomography for internal delay estimation," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01)*, vol. 6, 2001, pp. 3409–3412 vol.6.

[16] Y. Tsang, M. Coates, and R. Nowak, "Network delay tomography," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2125–2136, Aug. 2003.

[17] Y. Sun, D. Li, and H. Sun, "Network Tomography and Improved Methods for Delay Distribution Inference," in *The 9th International Conference on Advanced Communication Technology*, vol. 2, Feb. 2007, pp. 1433–1437.

[18] K. Nakanishi, S. Hara, T. Matsuda, K. Takizawa, F. Ono, and R. Miura, "Synchronization-Free Delay Tomography Based on Compressed Sensing," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1343–1346, Aug. 2014.

[19] Y. Gao, W. Dong, C. Chen, J. Bu, T. Chen, M. Xia, X. Liu, and X. Xu, "Domo: Passive Per-Packet Delay Tomography in Wireless Ad-hoc Networks," in *2014 IEEE 34th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2014, pp. 419–428.

[20] O. Gurewitz, I. Cidon, and M. Sidi, "One-way delay estimation using network-wide measurements," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2710–2724, Jun. 2006.

[21] B. Latré, P. D. Mil, I. Moerman, B. Dhoedt, P. Demeester, and N. V. Dierdonck, "Throughput and delay analysis of unslotted ieee 802.15.4." *JNW*, vol. 1, no. 1, pp. 20–28, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/jnw/jnw1.html#LatreMMDDD06

[22] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, "Minimizing End-to-End Delay: A Novel Routing Metric for Multi-Radio Wireless Mesh Networks," in *IEEE INFOCOM 2009*, Apr. 2009, pp. 46 –54.

[23] V. Ozdemir, S. Muthukrishnan, and I. Rhee, “Scalable, low-overhead network delay estimation,” in *IEEE INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 3, Mar. 2000, pp. 1343–1350 vol.3.

[24] Rafael Camilo Lozoya Gámez, Pau Martí, Manel Velasco and Josep M. Fuertes, “Wireless Network Delay Estimation for Time-Sensitive Applications,” Automatic Control Department, Technical University of Catalonia, Research report ESAII RR-06-12, Jul. 2006. [Online]. Available: http://esaii.upc.edu/people/pmarti/nde_06.pdf

[25] J.-H. Choi and C. Yoo, “One-way Delay Estimation and Its Application,” *Comput. Commun.*, vol. 28, no. 7, pp. 819–828, May 2005. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2004.11.010

[26] V. Paxson, “End-to-end Routing Behavior in the Internet,” *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1109/90.649563

[27] M. Allman and V. Paxson, “On Estimating End-to-end Network Path Properties,” in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM ’99. New York, NY, USA: ACM, 1999, pp. 263–274. [Online]. Available: http://doi.acm.org/10.1145/316188.316230

[28] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, “Framework for IP Performance Metrics,” RFC 2330 (Informational), Internet Engineering Task Force, May 1998, updated by RFC 7312. [Online]. Available: http://www.ietf.org/rfc/rfc2330.txt

[29] J. Mahdavi and V. Paxson, “IPPM Metrics for Measuring Connectivity,” RFC 2678 (Proposed Standard), Internet Engineering Task Force, Sep. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2678.txt

[30] G. Almes, S. Kalidindi, and M. Zekauskas, “A One-way Packet Loss Metric for IPPM,” RFC 2680 (Proposed Standard), Internet Engineering Task Force, Sep. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2680.txt

[31] G. Almes, S. Kalidindi, and M. Zekauskas, “A One-way Delay Metric for IPPM,” RFC 2679 (Proposed Standard), Internet Engineering Task Force, Sep. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2679.txt

[32] G. Almes, S. Kalidindi, and M. Zekauskas, “A Round-trip Delay Metric for IPPM,” RFC 2681 (Proposed Standard), Internet Engineering Task Force, Sep. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2681.txt

[33] C. Demichelis and P. Chimento, “IP Packet Delay Variation Metric for IP Performance Metrics (IPPM),” RFC 3393 (Proposed Standard), Internet Engineering Task Force, Nov. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3393.txt

[34] V. Raisanen, G. Grotefeld, and A. Morton, “Network performance measurement with periodic streams,” RFC 3432 (Proposed Standard), Internet Engineering Task Force, Nov. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3432.txt

[35] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, “A One-way Active Measurement Protocol (OWAMP),” RFC 4656 (Proposed Standard), Internet Engineering Task Force, Sep. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4656.txt

[36] “CAIDA - Center for Applied Internet Data Analysis,” Accessed: 17-Jul-2015. [Online]. Available: http://www.caida.org

[37] K. Hedayat, R. Krzanowski, A. Morton, K. Yum, and J. Babiarz, “A Two-Way Active Measurement Protocol (TWAMP),” RFC 5357 (Proposed Standard), Internet Engineering Task Force, Oct. 2008, updated by RFCs 5618, 5938, 6038. [Online]. Available: http://www.ietf.org/rfc/rfc5357.txt

[38] A. Morton and S. V. den Berghe, “Framework for Metric Composition,” RFC 5835 (Informational), Internet Engineering Task Force, Apr. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5835.txt

[39] J. Fabini and A. Morton, “Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM),” RFC 7312 (Informational), Internet Engineering Task Force, Aug. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7312.txt

[40] H. W. Shin and S. Y. Sohn, “Application of an EWMA combining technique to the prediction of currency exchange rates,” *IIE Transactions*, vol. 39, no. 6, pp. 639–644, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1080/07408170600899474

[41] D. Jing-rong, “Combining Stock Market Volatility Forecasts Using an EWMA Technique,” in *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007*, Sep. 2007, pp. 5277–5280.

[42] R. S. Tsay, “Outliers, level shifts, and variance changes in time series,” *Journal of Forecasting*, vol. 7, no. 1, pp. 1–20, 1988. [Online]. Available: http://dx.doi.org/10.1002/for.3980070102

[43] C. Yan-ming, X. Yong-jun, W. Qiu-guang, and X. Lei, “An Adaptive Fault-tolerant Scheme for Wireless Sensor Networks,” in *WRI International Conference on Communications and Mobile Computing, 2009. CMC ’09*, vol. 2, Jan. 2009, pp. 32–36.

[44] J. Shu, L. Liu, and R. Zhang, “An Energy-Effective Link Quality Monitoring Mechanism for Event-driven Wireless Sensor Network,” in *WRI International Conference on Communications and Mobile Computing, 2009. CMC ’09*, vol. 2, Jan. 2009, pp. 111–115.

[45] J. Piorno, C. Bergonzini, D. Atienza, and T. Rosing, “Prediction and management in energy harvested wireless sensor nodes,” in *1st International Conference on Wireless*

Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009, May 2009, pp. 6–10.

[46] Z. Jiang, X. Jin, and Y. Zhang, “A Weather-Condition Prediction Algorithm for Solar-Powered Wireless Sensor Nodes,” in *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, Sep. 2010, pp. 1–4.

[47] M. Xue, Z. Hai, and Z. Jian, “Research on EWMA based link quality evaluation algorithm for WSN,” in *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011*, vol. 1, Jul. 2011, pp. 757–759.

[48] A. Cammarano, C. Petrioli, and D. Spenza, “Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks,” in *2012 IEEE 9th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Oct. 2012, pp. 75–83.

[49] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, “Routing Metrics for Minimizing End-to-End Delay in Multiradio Multichannel Wireless Networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2293–2303, Nov. 2013.

[50] L. Abhilash, D. Goenka, and C. Kumar, “Dynamic data aggregation for energy optimization in multi-hop Wireless Sensor Networks,” in *Advance Computing Conference (IACC), 2014 IEEE International*, Feb. 2014, pp. 143–148.

[51] R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel, “A survey on routing metrics,” *TIK Report 262, ETH-Zentrum*, 2007.

[52] R. Draves, J. Padhye, and B. Zill, “Routing in multi-radio, multi-hop wireless mesh networks,” in *In ACM MobiCom.* ACM Press, 2004, pp. 114–128.

[53] R. Draves, J. Padhye, and B. Zill, “Comparison of Routing Metrics for Static Multi-hop Wireless Networks,” in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM ’04. New York, NY, USA: ACM, 2004, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1015467.1015483

[54] S. Biaz, B. Qi, and Y. Ji, “Improving Expected Transmission Time Metric in Multi-Rate Multi-Hop Networks,” in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, Jan. 2008, pp. 533 –537.

[55] D. Teng, S. Yang, D. Wang, and Y. Hu, “NQETT: Node Quality Adjusted ETT for Wireless Mesh Networks,” in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM ’08. 4th International Conference on*, Oct. 2008, pp. 1 –4.

[56] H. Zhou, C. Huang, Y. Cheng, and G. Wang, “A New Multi-metric QoS Routing Protocol in Wireless Mesh Network,” in *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC ’09. International Conference on*, vol. 1, Apr. 2009, pp. 459 –467.

[57] V. Raman and M. Caesar, "A Practical Approach for Providing QoS in Multichannel Ad-Hoc Networks Using Spectrum Width Adaptation," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, Dec. 2009, pp. 1 –6.

[58] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207006000239

[59] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. Kamaev, "A survey of forecast error measures," *World Applied Sciences Journal - Information Technologies in Modern Industry, Education & Society*, vol. 24, pp. 171–176, 2013.

[60] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633 (Informational), Internet Engineering Task Force, Jun. 1994. [Online]. Available: http://www.ietf.org/rfc/rfc1633.txt

[61] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475 (Informational), Internet Engineering Task Force, Dec. 1998, updated by RFC 3260. [Online]. Available: http://www.ietf.org/rfc/rfc2475.txt

[62] J. Babiarz, K. Chan, and F. Baker, "Configuration Guidelines for DiffServ Service Classes," RFC 4594 (Informational), Internet Engineering Task Force, Aug. 2006, updated by RFC 5865. [Online]. Available: http://www.ietf.org/rfc/rfc4594.txt

[63] D. Gupta, D. Wu, C. Chen, C.-N. Chuah, P. Mohapatra, and S. Rungta, "Experimental Study of Measurement-based Admission Control for Wireless Mesh Networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007*, Oct. 2007, pp. 1–9.

[64] L. Fàbrega and T. Jové, "A review of the architecture of admission control schemes in the internet," *Network Protocols and Algorithms*, vol. 5, no. 3, pp. 1–32, 2013.

[65] D. Mitzel, D. Estrin, S. Shenker, and L. Zhang, "A study of reservation dynamics in integrated services packet networks," in *Proceedings IEEE INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation*, vol. 2, Mar. 1996, pp. 871–879 vol.2.

[66] M. Fidler and V. Sander, "A parameter based admission control for differentiated services networks," *Computer Networks*, vol. 44, pp. 463–479, 2004.

[67] M. Mushtaq and T. Ahmed, "End-to-End QoS Provisioning for Real-Time Video Streaming over SP-Driven P2p Networks Using Admission Control," in *IEEE International Conference on Communications, 2009. ICC '09*, Jun. 2009, pp. 1–5.

[68] S. Georgoulas, P. Trimintzios, G. Pavlou, and K. Ho, "Heterogeneous real-time traffic admission control in differentiated services domains," in *IEEE Global Telecommunications Conference, 2005. GLOBECOM '05*, vol. 1, Nov. 2005, pp. 6 pp.–.

[69] I. Orhan and T. Lindh, "Measurement-Based Admission Control in Wireless Sensor Networks," in *Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on*, Jul. 2010, pp. 447 –452.

[70] O. Brewer and A. Ayyagari, "Comparison and analysis of measurement and parameter based admission control methods for Quality of Service (QoS) provisioning," in *Military Communications Conference, 2010 - MILCOM 2010*, Oct. 2010, pp. 184–188.

[71] W. Jiao, M. Sheng, K.-S. Lui, and Y. Shi, "End-to-End Delay Distribution Analysis for Stochastic Admission Control in Multi-hop Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 3, pp. 1308–1320, Mar. 2014.

[72] S. Y. Yerima, "Implementation and evaluation of measurement-based admission control schemes within a converged networks qos management framework," *International Journal of Computer Networks and Communications, IJCNC*, vol. 3, No.4, 2011. [Online]. Available: http://http://airccse.org/journal/cnc/0711cnc10.pdf

[73] A. Davy, D. Botvich, and B. Jennings, "Empirical Effective Bandwidth Estimation for IPTV Admission Control," in *Real-Time Mobile Multimedia Services*, ser. Lecture Notes in Computer Science, D. Krishnaswamy, T. Pfeifer, and D. Raz, Eds. Springer Berlin Heidelberg, 2007, no. 4787, pp. 125–137. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-540-75869-3_11

[74] X. Yin, X. Zhou, M. Pan, and S. Li, "Admission control with multi-constrained QoS providing in Wireless Sensor Networks," in *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, Apr. 2010, pp. 524 –529.

[75] E. D. Jensen, C. D. Locke, and H. Tokuda, "A Time-Driven Scheduling Model for Real-Time Operating Systems," *RTSS*, vol. 85, pp. 112–122, 1985.

[76] D. Wu, "Providing quality-of-service guarantees in wireless networks," Ph.D. dissertation, Carnegie Mellon University, 2003.

[77] R. Yu, H. Shu, and W. Jiang, "Low-Complexity Packet Scheduling Algorithms for Streaming Scalable Media Based on Time Utility Function," *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2270–2280, Dec. 2014.

[78] L. Breslau, S. Jamin, and S. Shenker, "Comments on the performance of measurement-based admission control algorithms," in *IEEE INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, vol. 3, Mar. 2000, pp. 1233–1242 vol.3.

[79] L. Macpherson, "Overload protection for commodity network appliances," in *Advances in Computer Systems Architecture*, ser. Lecture Notes in Computer Science, C. Jesshope and C. Egan, Eds. Springer Berlin Heidelberg, 2006, vol. 4186, pp. 203–218. [Online]. Available: http://dx.doi.org/10.1007/11859802_17

[80] I. Stoica and H. Zhang, "Lira: An approach for service differentiation in the internet," in *In Proc. of NOSSDAV'98*, 1998, pp. 115–128.

[81] "SeedEye," Accessed: 17-Jul-2015. [Online]. Available: http://www.evidence.eu.com/products/seed-eye.html

[82] "WiSMote," Accessed: 17-Jul-2015. [Online]. Available: http://wismote.org

[83] "Z1 mote," Accessed: 17-Jul-2015. [Online]. Available: http://zolertia.com/products/z1

[84] "MICAz," Accessed: 17-Jul-2015. [Online]. Available: http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf

[85] "Telos Rev A Datasheet," Accessed: 17-Jul-2015. [Online]. Available: https://sites.google.com/site/pedrofcpinto/Home/links/Telos-RevA-Datasheet.pdf

[86] "Telos Rev B Datasheet," Accessed: 17-Jul-2015. [Online]. Available: https://sites.google.com/site/pedrofcpinto/Home/links/Telos-RevB-Datasheet.pdf

[87] "Tmote Sky Project," Accessed: 17-Jul-2015. [Online]. Available: http://www.snm.ethz.ch/Projects/TmoteSky

[88] "Tmote Sky Datasheet," Accessed: 17-Jul-2015. [Online]. Available: https://sites.google.com/site/pedrofcpinto/Home/links/Tmote-Sky-Datasheet.pdf

[89] "MoteIV Application Note 001 - Telos Rev.A, Telos Rev.B and Tmote Sky differences," Accessed: 17-Jul-2015. [Online]. Available: https://sites.google.com/site/pedrofcpinto/Home/files/moteiv-an-001.pdf

[90] "MSP430F15x, MSP430F16x and MSP430F161x Microcontroller Datasheet," Accessed: 17-Jul-2015. [Online]. Available: http://www.ti.com/lit/gpn/msp430f1611

[91] M. Kuorilehto, M. Kohvakka, J. Suhonen, P. Hämäläinen, M. Hännikäinen, and T. D. Hämäläinen, *Ultra-Low Energy Wireless Sensor Networks in Practice: Theory, Realization and Deployment*. Wiley Publishing, 2008.

[92] "Raspberry Pi Foundation Web site," Accessed: 17-Jul-2015. [Online]. Available: https://www.raspberrypi.org/

[93] "Benchmarking the Raspberry PI 2," Accessed: 17-Jul-2015. [Online]. Available: http://hackaday.com/2015/02/05/benchmarking-the-raspberry-pi-2/

[94] "CC2420 RF Transceiver Datasheet," Accessed: 17-Jul-2015. [Online]. Available: http://www.ti.com/lit/gpn/cc2420

[95] "TinyOS," Accessed: 17-Jul-2015. [Online]. Available: http://www.tinyos.net/

[96] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks, 2004.* IEEE, 2004, pp. 455–462.

[97] "Contiki OS Webpage," Accessed: 17-Jul-2015. [Online]. Available: http://www.contiki-os.org/

[98] H. Will, K. Schleiser, and J. Schiller, "A real-time kernel for wireless sensor networks employed in rescue scenarios," in *IEEE 34th Conference on Local Computer Networks, 2009. LCN 2009*, Oct. 2009, pp. 834–841.

[99] E. Baccelli, O. Hahm, M. Wählisch, M. Günes, and T. Schmidt, "RIOT: One OS to Rule Them All in the IoT," INRIA, Research Report, No. RR–8176, Dec. 2012. [Online]. Available: https://hal.inria.fr/hal-00768685/document

[100] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 79–80.

[101] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks," in *International Conference on Information Processing in Sensor Networks, 2008. IPSN '08*, Apr. 2008, pp. 233–244.

[102] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, 2003, pp. 85–98.

[103] A. Dunkels, J. Alonso, and T. Voigt, "Making TCP/IP viable for wireless sensor networks," *SICS Research Report*, 2003.

[104] T. V. Chien, H. N. Chan, and T. N. Huu, "A comparative study on operating system for Wireless Sensor Networks," in *2011 International Conference on Advanced Computer Science and Information System (ICACSIS)*, Dec. 2011, pp. 73–78.

[105] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels, "Making sensor networks IPv6 ready," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, p. 421–422.

[106] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, "Protothreads: simplifying event-driven programming of memory-constrained embedded systems," in *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, 2006, pp. 29–42.

[107] "IEEE Draft Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment- Sub 1 GHz License-Exempt Operation," *IEEE P802.11ah/D4.0, January 2015 (Amendment to IEEE Std 802.11REVmc/D3.0)*, pp. 1–626, April 2015.

[108] Y. Zhou, H. Wang, S. Zheng, and Z. Lei, "Advances in IEEE 802.11ah standardization for machine-type communications in sub-1ghz WLAN," in *Communications Workshops (ICC), 2013 IEEE International Conference on*, June 2013, pp. 1269–1273.

[109] T. Adame, A. Bel, B. Bellalta, J. Barcelo, and M. Oliver, “IEEE 802.11AH: the WiFi approach for M2M communications,” *Wireless Communications, IEEE*, vol. 21, no. 6, pp. 144–152, December 2014.

[110] “IEEE Standard for Information technology – Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs),” *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 1–320, Sep. 2006.

[111] “IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs),” *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, Sep. 2011.

[112] “IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs),” *IEEE Std 802.15.4-2003*, pp. 1–670, 2003.

[113] “A lightweight TCP/IP stack - lwIP Project,” Jan 2001, Accessed: 17-Jul-2015. [Online]. Available: http://savannah.nongnu.org/projects/lwip/

[114] J. Shang and H. Ding, “Application of lightweight protocol stack LwIP on embedded Ethernet,” in *2011 International Conference on Electrical and Control Engineering (ICECE)*, Sep. 2011, pp. 3373–3376.

[115] “Arduino IPv6 stacks,” Accessed: 17-Jul-2015. [Online]. Available: http://departements.telecom-bretagne.eu/rsm/logiciels/arduino-ipv6-stacks/

[116] I. Glaropoulos, V. Vukadinovic, and S. Mangold, “Contiki80211: An IEEE 802.11 Radio Link Layer for the Contiki OS,” in *2014 IEEE Intl Conf on 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)*, Aug. 2014, pp. 621–624.

[117] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[118] R. Daidone, G. Dini, and M. Tiloca, “On experimentally evaluating the impact of security on ieee 802.15.4 networks,” in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, June 2011, pp. 1–6.

[119] Y. Xiao, H. Chen, B. Sun, R. Wang, and S. Sethi, “MAC security and security overhead analysis in the IEEE 802.15.4 wireless sensor networks,” *EURASIP J. Wireless Comm. and Networking*, vol. 2006, 2006. [Online]. Available: http://dx.doi.org/10.1155/WCN/2006/93830

[120] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 6282, 6775. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[121] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919 (Informational), Internet Engineering Task Force, Aug. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4919.txt

[122] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282 (Proposed Standard), Internet Engineering Task Force, Sep. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6282.txt

[123] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," RFC 6775 (Proposed Standard), Internet Engineering Task Force, Nov. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6775.txt

[124] C.-Y. Yum, Y. S. Beun, S. Kang, Y. R. Lee, and J. Song, "Methods to use 6lowpan in IPv4 network," in *The 9th International Conference on Advanced Communication Technology*, vol. 2, Feb. 2007, pp. 969–972.

[125] "SICSlowpan - Internet for low-power, low-cost Wireless Project," Accessed: 17-Jul-2015. [Online]. Available: https://www.sics.se/projects/sicslowpan-internet-for-low-power-low-cost-wireless

[126] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the Performance of RPL and 6LoWPAN in TinyOS," in *Proceedings of Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, April 2011.

[127] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[128] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6552 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6552.txt

[129] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," RFC 6719 (Proposed Standard), Internet Engineering Task Force, Sep. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6719.txt

[130] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," RFC 6206 (Proposed Standard), Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6206.txt

[131] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, “Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks,” RFC 6551 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6551.txt

[132] N. Tsiftes, J. Eriksson, and A. Dunkels, “Low-power wireless IPv6 routing with ContikiRPL,” in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN ’10. New York, NY, USA: ACM, 2010, p. 406–407.

[133] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-haggerty, A. Terzis, A. Dunkels, and D. Culler, “ContikiRPL and TinyRPL: Happy Together,” in *In Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks IP+SN*, 2011.

[134] L. Guan, K. Kuladinithi, T. Potsch, and C. Goerg, “A deeper understanding of interoperability between tinyrpl and contikirpl,” in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, April 2014, pp. 1–6.

[135] A. Varga *et al.*, “The omnet++ discrete event simulation system,” in *Proceedings of the European simulation multiconference (ESM’2001)*, vol. 9, no. S 185, 2001, p. 65.

[136] “NS-2,” Accessed: 17-Jul-2015. [Online]. Available: http://www.isi.edu/nsnam/ns/

[137] “NS-3,” Accessed: 17-Jul-2015. [Online]. Available: http://www.nsnam.org/

[138] “J-Sim Web site,” 2015, Accessed: 17-Jul-2015. [Online]. Available: https://sites.google.com/site/jsimofficial/

[139] S. Sundresh, W. Kim, and G. Agha, “SENS: a sensor, environment and network simulator,” in *Simulation Symposium, 2004. Proceedings. 37th Annual*, Apr. 2004, pp. 221–228.

[140] 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

[141] D. Blazakis, J. McGee, D. Rusk, and J. Baras, “Atemu: a fine-grained sensor network simulator,” in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, Oct 2004, pp. 145–152.

[142] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, “Cross-Level Sensor Network Simulation with COOJA,” in *Proceedings 2006 31st IEEE Conference on Local Computer Networks*, 2006, pp. 641–648.

[143] “VMware,” Accessed: 17-Jul-2015. [Online]. Available: https://www.vmware.com/

[144] Pedro Pinto, António Pinto and Manuel Ricardo, "Reducing WSN Simulation Runtime by using Multiple Simultaneous Instances," in *Symposium on Modelling and Simulation in Computer Sciences and Engineering (ICNAAM 2014)*, Rhodes, Greece, 2014.

[145] Pedro Pinto, António Pinto and Manuel Ricardo, "Reducing Simulation Runtime in Wireless Sensor Networks: A Simulation Framework to Reduce WSN Simulation Runtime by Using Multiple Simultaneous Instances (Book Chapter) - accepted for publishing in april, 2015," in *Handbook of Research on Computational Simulation and Modeling in Engineering*, 2014.

[146] A. Dunkels, "The ContikiMAC radio duty cycling protocol," Dec. 2011, Accessed: 17-Jul-2015.