Springer Theses
Recognizing Outstanding Ph.D. Research

Hayder Al-Kashoash

Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things

Springer

Springer Theses

Recognizing Outstanding Ph.D. Research

Aims and Scope

The series "Springer Theses" brings together a selection of the very best Ph.D. theses from around the world and across the physical sciences. Nominated and endorsed by two recognized specialists, each published volume has been selected for its scientific excellence and the high impact of its contents for the pertinent field of research. For greater accessibility to non-specialists, the published versions include an extended introduction, as well as a foreword by the student's supervisor explaining the special relevance of the work for the field. As a whole, the series will provide a valuable resource both for newcomers to the research fields described, and for other scientists seeking detailed background information on special questions. Finally, it provides an accredited documentation of the valuable contributions made by today's younger generation of scientists.

Theses are accepted into the series by invited nomination only and must fulfill all of the following criteria

- They must be written in good English.
- The topic should fall within the confines of Chemistry, Physics, Earth Sciences, Engineering and related interdisciplinary fields such as Materials, Nanoscience, Chemical Engineering, Complex Systems and Biophysics.
- The work reported in the thesis must represent a significant scientific advance.
- If the thesis includes previously published material, permission to reproduce this must be gained from the respective copyright holder.
- They must have been examined and passed during the 12 months prior to nomination.
- Each thesis should include a foreword by the supervisor outlining the significance of its content.
- The theses should have a clearly defined structure including an introduction accessible to scientists not expert in that particular field.

More information about this series at http://www.springer.com/series/8790

Hayder Al-Kashoash

Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things

Doctoral Thesis accepted by
the University of Leeds, Leeds, UK

Springer

Author
Dr. Hayder Al-Kashoash
Technical Institute/Qurna
Southern Technical University
Basra, Iraq

Supervisor
Prof. Andrew Kemp
Electronic and Electrical Engineering School
The University of Leeds
Leeds, UK

ISSN 2190-5053 ISSN 2190-5061 (electronic)
Springer Theses
ISBN 978-3-030-17731-7 ISBN 978-3-030-17732-4 (eBook)
https://doi.org/10.1007/978-3-030-17732-4

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Supervisor's Foreword

The internet of things (IoT) is rapidly becoming widely accepted and recognized as the communications paradigm of our time. This was not the case when this student was starting his Ph.D. studies but it has rapidly moved forward, at that stage wireless sensor networks (WSNs) were the dominant solution and this migrated to WSNs and 6LoWPAN networks. Indeed in this thesis, the IoT is largely considered to be a combination of WSNs and 6LoWPAN networks. The work contained in this thesis traces the work of a very gifted student who with small encouragement from his supervisor developed a thorough investigation into the control of congestion for IoT applications. Congestion is the overloading of traffic across networks and clearly needs to be avoided. As a result of congestion, routers in a network will find their input and their output buffers will become full and overflow leading to data loss. Fundamentally, it occurs when the aggregate data coming into a network exceeds the total data leaving the network. Methods to avoid congestion are not straightforward but 'boil down' to reducing the data entering the network or increasing the rate at which data leaves the network. In the TCP/IP suite, TCP itself deals with congestion control through managing the rate at which nodes transmit data into the network. TCP is not a ready solution for the IoT due to its heavy header overhead; something more lightweight is needed. The lightweight solution is fundamentally UDP but this does not have any provision for congestion control. Consequently, there is a real need in the rapidly expanding IoT for the provision of congestion control. In this thesis, Hayder examines the problem of congestion, how it can be detected, what action can be taken to avoid it and offers suggestions of how best to alleviate it. He details the networks he is dealing with and the propagation mechanisms which will lead to congestion. This work uses the simulation of networks to explore the actions which can be taken and the anticipated results of those actions. He also uses real network experiments to validate his simulation work. In conclusion, I feel that by working through this thesis the reader have their understanding of networking, and of congestion very greatly enhanced.

Leeds, UK — Prof. Andrew Kemp

Abstract

The internet of things (IoT) is the next big challenge for the research community. The IPv6 over low-power wireless personal area network (6LoWPAN) protocol stack is considered a key part of the IoT. Due to power, bandwidth, memory, and processing resources limitation, heavy network traffic in 6LoWPAN networks causes congestion which significantly degrades network performance and impacts on the quality of service (QoS) aspects. This thesis addresses the congestion control issue in 6LoWPAN networks. In addition, the related literature is examined to define the set of current issues and to define the set of objectives based upon this.

An analytical model of congestion for 6LoWPAN networks is proposed using Markov chain and queuing theory. The derived model calculates the buffer-loss probability and the number of received packets at the final destination in the presence of congestion. Simulation results show that the analytical modelling of congestion has a good agreement with simulation. Next, the impact of congestion on 6LoWPAN networks is explored through simulations and real experiments where an extensive analysis is carried out with different scenarios and parameters. Analysis results show that when congestion occurs, the majority of packets are lost due to buffer overflow as compared to channel loss. Therefore, it is important to consider buffer occupancy in protocol design to improve network performance.

Based on the analysis concluded, a new IPv6 routing protocol for low-power and lossy network (RPL) routing metric called buffer occupancy is proposed that reduces the number of lost packets due to buffer overflow when congestion occurs. Also, a new RPL objective function called congestion-aware objective function (CA-OF) is presented. The proposed objective function works efficiently and improves the network performance by selecting less congested paths. However, sometimes the non-congested paths are not available and adapting the sending rates of source nodes is important to mitigate the congestion.

Accordingly, the congestion problem is formulated as a non-cooperative game framework where the nodes (players) behave uncooperatively and demand high data rate in a selfish way. Based on this framework, a novel and simple congestion control mechanism called game theory based congestion control framework (GTCCF) is proposed to adopt the sending rates of nodes and therefore, congestion

can be solved. The existence and uniqueness of Nash equilibrium in the designed game are proved and the optimal game solution is computed by using Lagrange multipliers and Karush–Kuhn–Tucker (KKT) conditions. GTCCF is aware of node priorities and application priorities to support the IoT application requirements. On the other hand, combining and utilizing the resource control strategy (i.e. finding non-congested paths) and the traffic control strategy (i.e. adapting sending rate of nodes) into a hybrid scheme is important to efficiently utilize the network resources. Based on this, a novel congestion control algorithm called optimization-based hybrid congestion alleviation (OHCA) is proposed. The proposed algorithm combines traffic control and resource control strategies into a hybrid solution by using the network utility maximization (NUM) framework and a multi-attribute optimization methodology, respectively. Also, the proposed algorithm is aware of node priorities and application priorities to support the IoT application requirements.

Declaration

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis, where reference has been made to the work of others. Most materials contained in the chapters of this thesis have been previously published in research articles written by the author of this work (Hayder Ahmed Abdulmohsin Al-Kashoash), who appears as lead (first) author in all of them. The research has been supervised and guided by Prof. Andrew Kemp, and he appears as a co-author on these articles. All the materials included in this document is of the author's entire intellectual ownership.

A. Details of the publications which have been used (e.g. titles, journals, dates, names of authors):

In Chapter 2:

'Congestion Control for Wireless Sensor and 6LoWPAN Networks: Toward the Internet of Things', *Wireless Networks Journal, Springer*, Published. Co-authors: Harith Kharrufa, Yaarob Al-Nidawi and Andrew Kemp.

In Chapter 3:

'Congestion Analysis for Low Power and Lossy Networks', *IEEE 2016 Wireless Telecommunications Symposium (WTS)*, Published. Co-authors: Yaarob Al-Nidawi and Andrew Kemp. (DOI: https://doi.org/10.1109/WTS.2016.748202710.1109/WTS.2016.7482027).

'Analytical Modelling of Congestion for 6LoWPAN Networks', *Elsevier ICT Express Journal*, Published. Co-authors: Fadoua Hassen, Harith Kharrufa and Andrew Kemp.

In Chapter 4:

'Congestion-Aware RPL for 6LoWPAN Networks", *IEEE 2016 Wireless Telecommunications Symposium (WTS)*, Published. Co-authors: Yaarob Al-Nidawi and Andrew Kemp. (DOI: https://doi.org/10.1109/WTS.2016.748202610.1109/WTS.2016.7482026). "***Best Student Paper Award***".

In Chapter 5:

'Congestion Control for 6LoWPAN Networks: A Game Theoretic Framework', *IEEE Internet of Things Journal*, Published. Co-authors: Maryam Hafeez and Andrew Kemp. (DOI: https://doi.org/10.1109/JIOT.2017.266626910.1109/JIOT.2017.2666269).

In Chapter 6:

'Optimization Based Hybrid Congestion Alleviation for 6LoWPAN Networks', *IEEE Internet of Things Journal*, Published. Co-authors: Hayder Amer, Lyudmila Mihaylova and Andrew Kemp.

B. Details of the work contained within these publications which is directly attributable to Hayder A. A. Al-Kashoash:

With the exceptions detailed in section C, the published work is entirely attributable to Hayder A. A. Al-Kashoash: the literature review necessary to construct and originate the ideas behind the published manuscripts, the novel ideas presented in the papers, the implementation of congestion analysis and the proposed algorithms used in the Contiki OS and all the work necessary in the editing process of the manuscripts.

C. Details of the contributions of other authors to the work:

Prof. Andrew Kemp is the co-author for all the publications listed above. These publications have been written under his supervision, benefiting from excellent technical advice and editorial, patient guidance and valuable feedback.

Yaarob Al-Nidawi and Maryam Hafeez contributed with recommendations about how to efficiently structure papers and how to emphasize the originality of the work and to make it more accessible to the reader.

Harith Kharrufa, Fadoua Hassen, Hayder Amer and Lyudmila Mihaylova performed proofreading to the final drafts of the papers to ensure the solidity of the papers.

The right of Hayder A. A. Al-Kashoash to be identified as Author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Acknowledgements

First, I would like to express my sincere appreciation and thanks to my supervisor Prof. Andrew Kemp for his guidance, patience, motivation, continues support and immense knowledge. Your guidance helped me in all the time of research and writing of this thesis. I would like to thank you for encouraging my research and for allowing me to grow as a researcher.

A special thanks to my caring father and loving mother. Words cannot express how grateful I am to you. Your prayers for me was what sustained me this far. It was really difficult for me to be away from you all these years, but you were always beating inside my heart. Whatever I am now is because of you. Thank you from the heart.

I am thankful to my wife for her love, patience and support that have always been my strength. You were always been here with me. Also I am thankful for my sons, you brought the joy to my life and I have found my smile with you in all difficult times throughout this Ph.D. Also, I would like to express my deepest gratitude for all my family members for their love and prayers. Thank you all for being a part of my life.

I also thank the Higher Committee for Education Development/Iraqi Prime Minister Office and Technical Institute/Qurna, Southern Technical University, Basra, Iraq for their valuable support. Without their precious help, it would not be possible to conduct this research.

For all my friends in the school, thank you all for your valuable support and I was really lucky to work with wonderful friends like you.

Contents

Abbreviations

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Network
ACK	Acknowledgement
ACO	Ant Colony Optimization
AHP	Analytical Hierarchy Process
AIMD	Additive-Increase/Multiplicative-Decrease
AODV	Ad hoc On-Demand Distance Vector
BER	Bit Error Rate
BLE	Bluetooth Low Energy
BLIP	Berkeley Low-power IP stack
CCA	Clear Channel Assessment
CoAP	Constrained Application Protocol
CoRE	Constrained RESTfull Environments
CRC	Cyclic Redundancy Check
CSMA	Carrier-Sense Multiple Access
DAG	Directed Acyclic Graph
DAO	Destination Advertisement Object
DGRM	Directed Graph Radio Medium
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination-Oriented DAG
DSDV	Destination-Sequenced Distance Vector
DYMO-low	Dynamic MANET On-Demand for 6LoWPAN
ESB	Embedded Sensor Board
ETX	Expected Transmission Count
FCS	Frame Check Sequence
FFD	Full-Function Device
GRA	Grey Relational Analysis
GTS	Guaranteed Time Slots
HiLow	Hierarchical Routing over 6LoWPAN
HTTP	Hypertext Transfer Protocol

IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPHC	Improved Header Compression
IS-IS	Intermediate System-to-Intermediate System
KKT	Karush–Kuhn–Tucker
LLN	Low-Power and Lossy Network
LOAD	6LoWPAN Ad hoc On-Demand Distance Vector
LPWAN	Low-Power Wide Area Network
LR-WPAN	Low-Rate Wireless Personal Area Network
LQI	Link Quality Indicator
MAC	Medium Access Control
MADM	Multiple Attribute Decision-Making
MANET	Mobile Ad Hoc NETwork
MTU	Maximum Transmission Unit
Nam	Network animator
NED	NEtwork Description
NesC	Network embedded system C
NFC	Near-Field Communication
NHC	Next Header Compression
NUM	Network Utility Maximization
OF	Objective Function
OF0	Objective Function zero
OLSR	Optimized Link State Routing Protocol
OS	Operating System
OSPF	Open Shortest Path First
PDR	Packet Delivery Ratio
PHY	PHYsical layer
POS	Personal Operating Space
QoD	Quality of Data
QoS	Quality of Service
RDC	Radio Duty Cycle
REST	REpresentational State Transfer
RFC	Request For Comment
RFD	Reduced-Function Device
RFID	Radio-Frequency Identification
RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
SAW	Simple Additive Weighting
SD	Standard Deviation
TCP	Transmission Control Protocol
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
UDGM	Unit Disk Graph Medium
UDP	User Datagram Protocol
UGRM	Objective Function

VANET	Vehicular Ad hoc NETwork
WBAN	Wireless Body Area Network
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

Symbols

μ^I	Average departure rate of intermediate node I_l
μ^L	Average departure rate of leaf node L
B	Buffer size
BE	Backoff exponent of CSMA
CC_b	Channel Capacity in bit/s
CC_p	Channel Capacity in packet/s
I_l	Intermediate node
L_{inter}	Average number of lost packets/s at intermedaite node's buffer
L_k	Leaf node L_k
L_{leaf}	Average number of lost packets/s at leaf node's buffer
M	Number of leaf nodes
m	Maximum number of backoffs in CSMA
PL	Data packet length (bit)
n	Maximum number of retransmissions in CSMA
NB	Number of backoffs in CSMA
P_{arr}	Probability of packet arrival to node's buffer
$P_{buffer-loss}$	Probability of packet loss at node's buffer
P^j_{caf}	Probability of packet loss due to channel access failure at node j
$P^j_{ch-loss}$	Probability of channel loss for node j
$P_{coll,j}$	Probability that transmitted packet encounters collision at node j
$P_{dep.}$	Probability of packet departure from node's buffer
P^j_{mrl}	Probability of packet loss due to maximum number of retransmissions limit at node j
S	Sink node
T	Time step of Markov chain state transition
T_{ACK}	Time required to transmit ACK packet
T_{data}	Time required to transmit data packet
$T_{data,coll}$	Actual time required to transmit one data packet with collision
$T_{data,nocoll}$	Actual time required to transmit one data packet without collision
w_{etx}	Weight of ETX metric in CA-OF

w_{bo}	Weight of BO metric in CA-OF
$P_{cca,j}$	Probability that CCA is busy in node j
λ_k^I	Average arrival rate at intermediate node I_l from leaf node L_k
λ_{total}^I	Total arrival rates at intermediate node I_l from all leaf nodes
λ_k	Average data rate of leaf node L_k
λ_S	Average number of received packets at sink node S every second
p_k	Priority of leaf node L_k
N	Number of applications hosted in a leaf node
p_k^j	Priority of application j hosted in leaf node L_k
λ_k^{max}	Maximum sending rate of leaf node L_k
S_k	Strategy space of leaf node L_k
SS	Strategy space of all leaf nodes
Φ_k	Payoff function of leaf node L_k
λ_{-k}	Vector of sending rates of all leaf nodes except leaf node L_k
$U_k(\lambda_k)$	Utility function of leaf node L_k
$C_k(\lambda_k, \lambda_{-k})$	Congestion cost function of leaf node L_k
$P_k(\lambda_k; p_k)$	Priority cost function of leaf node L_k
λ_{out}	Forwarding rate of the parent node
λ_{in}	Incoming rate to the parent node
ω_k	Preference parameter of function $U_k(\lambda_k)$
α_k	Preference parameter of function $C_k(\lambda_k, \lambda_{-k})$
β_k	Preference parameter of function $P_k(\lambda_k; p_k)$
λ_k^*	Optimal sending rate (Nash Equilibrium) of leaf node L_k
s^*	Vector of optimal sending rates (Nash Equilibrium) of all leaf nodes
$H(s)$	Hessian matrix of payoff function $\Phi_k(s)$
$\mathcal{L}_k$	Lagrangian function of leaf node L_k
λ_k^j	Sending rate of application j hosted in leaf node L_k
θ_j	Weight of application j
I_{check}	Congestion check interval time
ψ	Smoothing factor
A	Set of candidate parents
R	Set of routing metrics
G	Weight vector
D	Decision matrix
a_i	Candidate parent i
r_j	Routing metric j
x_{ij}	Normalized value of routing metric r_j for parent a_i
$\gamma(x_{ij}, x_{0j})$	Grey relational coefficient
$\Gamma(a_i)$	Grey relational grade
L	Set of children nodes
N_l	Child node N_l
p_l	Priority of child node N_l
K	Set of hosted applications in child node N_l
app^k	Application k

p_l^k	Priority of application app^k hosted in node N_l
λ_l	Sending rate of child node N_l
$U_l(\lambda_l)$	Utility function of child node N_l
$\boldsymbol{\lambda}$	Vector of sending rates of all children nodes
ϕ_l	Weight of $U_l(\lambda_l)$
λ_l^k	Sending rate of application app^k hosted in child node N_l
$U^k(\lambda_l^k)$	Utility function of application app^k hosted in child node N_l
ϕ^k	Weight of $U^k(\lambda_l^k)$

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

The internet of things (IoT) is considered to be the next big challenge for the Internet research community. Recently, the IoT has drawn significant research attention [1]. The IoT will comprise of billions of communicating devices, which extend the borders of the cyber world with physical entities and virtual components [2, 3]. These things, such as wireless sensor nodes, radio-frequency identification (RFID) tags and near-field communication (NFC) devices, are connected to the Internet with the ability to sense status and use real-time data. Also, they access historical data and developed algorithms, possibly triggering devices. This is leading to very powerful smart environments, e.g. building, health care, etc. [1, 4].

Wireless sensor networks (WSNs) are considered as one of the most important elements in the IoT [5]. IPv6 over low-power wireless personal area network (6LoWPAN) is used for full integration of WSN with the Internet, where sensor nodes implement the internet protocol (IP) stack though it has been originally designed for wired networks. However, the implementation of TCP/IP model in 6LoWPAN has many issues and problems due to the limitation of energy, bandwidth, processing, and buffer resources. Transmission control protocol (TCP) requires connection setup and termination before and after the data transmission and user datagram protocol (UDP) does not provide a congestion control mechanism. Thus, TCP and UDP are not efficient for 6LoWPAN [1, 3]. Therefore, one of the main issues in 6LoWPAN is congestion that causes packet loss, energy consumption and degrades throughput.

As wireless sensor nodes are connected to the Internet through 6LoWPAN, the applications become wider for 6LoWPAN networks, e.g. industrial, automation, health care, military, environment, logistics, etc. Generally, the applications can be categorized into four types (i.e. event-based, continuous, query-based, and hybrid applications) based on the data delivery method [6, 7]. In event-based applications, network traffic is typically low and suddenly becomes high in response to a detected

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_1

event. These high data rate packets cause congestion and therefore it is very important to consider congestion control.

In continuous applications, sensor nodes periodically send packets to the sink after predetermined time intervals. In query-based applications, the sink node sends a query to sensor nodes and they respond to the sink query by sending packets. Lastly, in the hybrid application type, the first three categories are combined into a hybrid application, i.e. sensor nodes send packets in response to an event (event-based) and at the same time send packets periodically (continuous) as well as send a reply to a sink query (query-based). This type of application will be common in the future as WSNs are connected into the Internet as part of the IoT [3, 6]. In the IoT applications, the sensor nodes host many different application types simultaneously (event-based, continuous, and query-based) with varied requirements. Some of them are real-time applications where the application data is time critical and delay constrained, e.g. healthcare monitoring and natural disasters detection (e.g. flooding), while others are non-real- time applications, e.g. measuring temperature and measuring humidity. Some applications send very important data and losing this data is not permitted, e.g. medical applications and fire detection applications. This brings new challenges to the congestion control algorithms and mechanisms designed to be aware of application priorities as well as node priorities.

Many mechanisms and algorithms have been proposed to solve the congestion problem in traditional WSNs. In contrast to traditional WSN, however, the 6LoWPAN networks might host a variety of applications at the same time as they connect to the Internet. Also, the protocol stack of 6LoWPAN is different from the traditional WSN one where sensor nodes in 6LoWPAN implement IP stack as they are connected to the Internet. A new layer is developed between data link layer and network layer called the adaptation layer to support IPv6 packet transmission over IEEE 802.15.4 wireless sensor networks. Moreover, in [8], Michopoulos et al. have demonstrated that Radio Duty Cycle (RDC) mechanisms impact the performance of congestion control algorithms. RDC is a technique used to conserve energy by switching the radio transceiver between sleeping mode and wake up mode periodically. This effect is neglected when designing and implementing many congestion control schemes in traditional WSN. Furthermore, two methods are used to solve or mitigate congestion problems in WSNs: traffic control and resource control. Many congestion control mechanisms have been proposed based on the resource control strategy, where the congestion control algorithm is responsible to construct network topology by selecting a non-congested path from source to destination. However, in 6LoWPAN networks; IPv6 routing protocol for low-power and lossy networks (RPL), which is expected to be the standard routing protocol for 6LoWPAN, is completely responsible for network topology construction by using an objective function (OF) (e.g. objective function zero (OF0)). Therefore, a conflict occurs between RPL protocol operation and resource control strategy based congestion control mechanisms in traditional WSN. Therefore, new congestion control algorithms that address these considerations are needed.

The majority of previous works on congestion control have not carefully considered the unique characteristics of IPv6 and 6LoWPAN in their design. The aim of this

research is to analyse and assess congestion conditions in 6LoWPAN and develop congestion control schemes as a step towards successful implementation of the IoT. Emerging architectures such as IEEE 802.15.4, IPv6 and 6LoWPAN are becoming dominant in WSNs, therefore congestion analysis and proposed mechanism implementations are based on these architectures.

1.2 6LoWPAN Protocol Stack

6LoWPAN enables transmission of IPv6 packets over low-power, low memory, low bandwidth, low processing capability and low-cost devices, which are compatible with the IEEE 802.15.4 standard. 6LoWPAN provides complete integration of wireless sensor nodes with the Internet. Connecting wireless sensor nodes to the Internet enables a wide range of applications for 6LoWPAN, e.g. industrial, automation, health, military, environment, logistics. The 6LoWPAN protocol stack involves IEEE 802.15.4 physical (PHY) and medium access control (MAC) layers, 6LoWPAN adaptation layer, network layer, transport layer and application layer as shown in Fig. 1.1. A review of the 6LoWPAN model layers is given in the next subsections.

1.2.1 Application Layer

The IoT makes the most Internet application protocols important for 6LoWPAN networks [1]. However, 6LoWPAN is challenging due to its small frame size, low data rate, limited memory, limited processing capabilities and power supply. Recently,

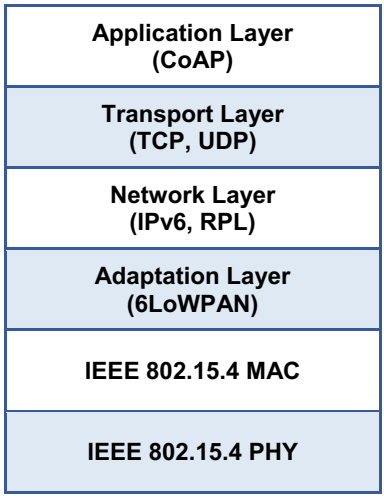

Fig. 1.1 6LoWPAN protocol stack

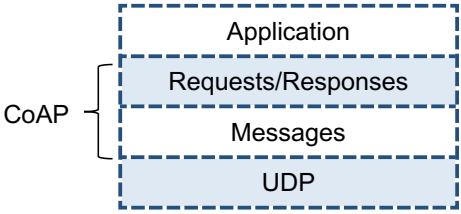

Fig. 1.2 Abstract layering of CoAP [9]

the Constrained RESTfull Environments (CoRE) working group has developed an important application protocol called Constrained Application Protocol (CoAP), which is a REpresentational State Transfer (REST) based web transfer protocol [9]. CoAP includes a subset of HTTP functionalities, which have been re-designed to meet the 6LoWPAN constraints. The CoAP protocol is built on top of UDP instead of TCP as used with HTTP.

The interaction model of CoAP is similar to the client/server model of HTTP. A CoAP request is equivalent to that of HTTP and is sent by a client using GET, POST, PUT and DELETE methods. The server then sends a response with a Response Code. CoAP defines four types of messages: Confirmable, Non-confirmable, Acknowledgement and Reset. Requests can be carried in Confirmable and Non-confirmable messages and responses can be carried in these as well as piggybacked in ACK messages. CoAP is logically considered as a two-layer approach: the messaging layer used to process the messaging features and the request/response interactions layer to deal with the client's requests and the server's responses as shown in Fig. 1.2

1.2.2 Transport Layer

In the IP protocol stack, two main transport protocols are widely used: TCP and UDP. TCP is a reliable connection-oriented byte stream protocol where reliability is achieved by using acknowledgement and retransmission. Also, TCP provides end-to-end flow control and congestion control by using a sliding window algorithm. Figure 1.3 shows the difference between flow control and congestion control [10]. Figure 1.3a illustrates the flow control problem where a small capacity and slower receiver is overwhelmed by a fast-transmitting sender. Flow control is an end-to-end mechanism that controls the traffic between sender and receiver, where the receiver is responsible for detecting and reacting to congestion. While, in the congestion control problem, a limited resources network is congested due to high offered-load packets into the network as shown in Fig. 1.3b. Congestion control is a hop-by-hop mechanism where the nodes along the path between source and destination are responsible for detecting and reacting to congestion. The major benefit of using the hop-by-hop mechanism is that it reacts to congestion occurrence much faster than the end-to-end. Therefore, the majority of congestion control algorithms in WSNs and 6LoWPAN networks use the hop-by-hop strategy. On the other hand, UDP is the simplest protocol in the TCP/IP suite. It does not support reliability and congestion

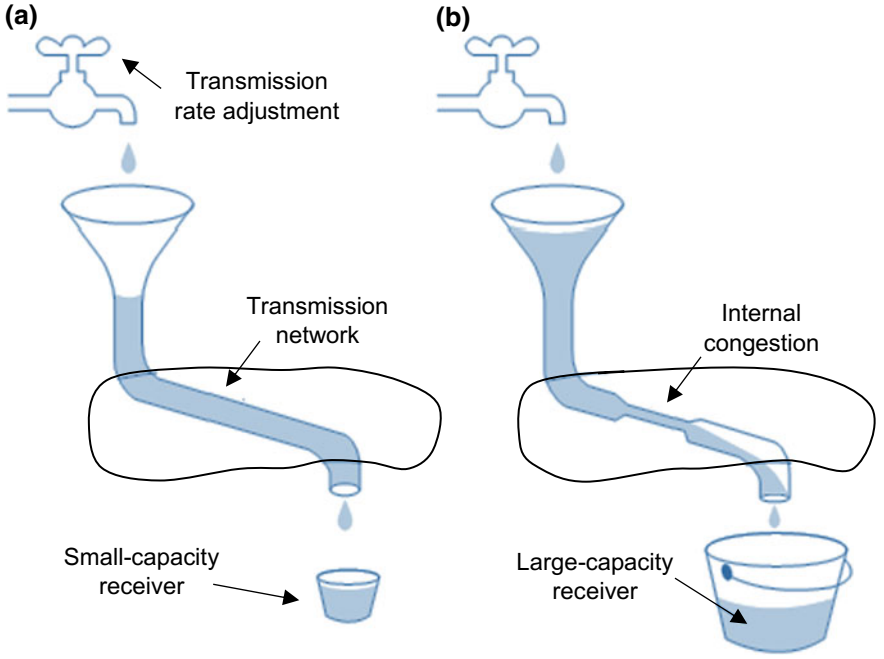

Fig. 1.3 **a** Flow control problem. **b** Congestion control problem [10]

control. Due to 6LoWPAN limitations, UDP is the most common transport protocol used in 6LoWPAN networks.

1.2.3 Network Layer

The main function of the routing protocol is to determine the 'best' path to reach a destination according to various metrics and objective functions. A number of IP routing protocols have been developed in various Internet Engineering Task Force (IETF) working groups, e.g. OSPF, IS-IS, AODV and OLSR. However, these routing protocols do not satisfy the routing requirements for 6LoWPAN networks which are as follows [11]:

- Low overhead on data packets.
- Low routing overhead.
- Minimal memory and computation requirements.
- Support for sleeping nodes considering battery saving.

After the implementation of the adaptation layer in the 6LoWPAN architecture, it is possible to make routing/forwarding decisions either in the network layer or in the adaptation layer. Generally, routing protocols in 6LoWPAN can be divided into two categories: 'mesh-under' and 'route-over' [12]. With the mesh-under scheme,

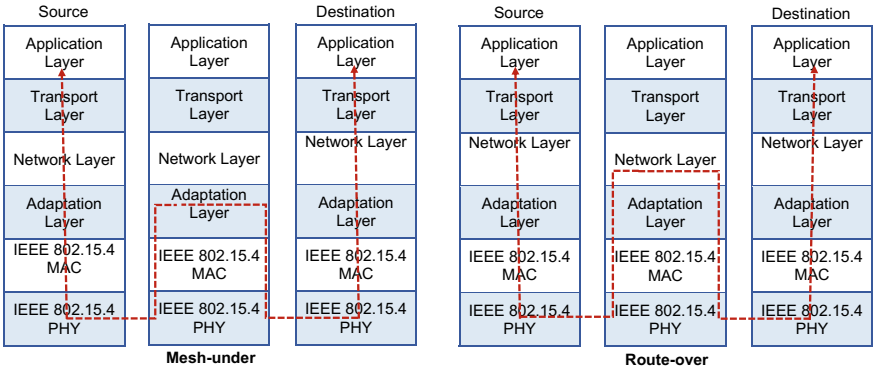

Fig. 1.4 Mesh-under versus route-over

the adaptation layer performs the packet routing and forwarding over multiple hops based on the 6LoWPAN header or the IEEE 802.15.4 link layer address. In the route-over, all routing decisions are taken in the network layer and packets are forwarded to the final destination by using IPv6 addresses. Figure 1.4 shows the difference between mesh-under and route-over.

Recently, a number of routing protocols have been developed for 6LoWPAN such as HiLow, LOAD, DYMO-low and RPL. A brief review of these protocols is given below:

Hierarchical Routing over 6LoWPAN (HiLow) [13]

HiLow uses dynamically assigned 16-bit unique short addresses for a 6LoWPAN device during an association operation with a neighbouring device. In HiLow, each node discovers its parent by sending a broadcast packet. If the node finds a parent node within its transmission range, it associates with that parent node, otherwise it configures itself as a coordinator. HiLow reduces the overhead of maintaining routing tables and supports large scalability. However, HiLow does not support any path recovery mechanism.

6LoWPAN Ad Hoc On-Demand Distance Vector (LOAD) [14]

LOAD is proposed based on the ad hoc on-demand distance vector (AODV) routing protocol. LOAD uses either 64-bit extended or 16-bit short addresses for 6LoWPAN devices. It maintains a routing table and a route request table, which are used in the route discovery phase. LOAD uses the link quality indicator (LQI) and the number of hops as routing metrics to determine the route from source to destination. Also, it uses acknowledged transmission for reliability. Unlike HiLow, LOAD uses a route discovery mechanism to repair the route locally.

Dynamic MANET On-Demand for 6LoWPAN (DYMO-Low) [15]

DYMO-low routing is based on the DYMO routing protocol. DYMO-low operates on the link layer directly to create a mesh network topology of 6LoWPAN devices. It

uses either 16-bit link layer short address or IEEE 64-bit extended address. DYMO-low performs route discovery and maintenance by using route request (RREQ), route reply (RREP) and route error (RERR) messages. Also, it utilizes LQI in addition to the route cost for selecting the best route to the final destination.

RPL [16]

RPL was developed by IETF Routing Over Low power and Lossy networks (ROLL) working group to meet the requirements and challenges of low power and lossy networks (LLNs). RPL is a distance-vector routing protocol which is designed on the basis of IEEE 802.15.4 physical and MAC layers [16]. In RPL networks, there are three types of nodes: root nodes which provide connectivity to other networks, intermediate nodes which forward packets to the root and leaf nodes [17]. RPL is designed to be quickly adaptive to network conditions and to provide an alternative path to the root node when the default path is not available [18].

The construction of RPL network topology is based on the directed acyclic graph (DAG) concept, where every node selects a neighbour as its parent based on an objective function which defines how nodes translate one or more metrics (delay, link quality, hop count, etc.) into rank. RPL organizes nodes as destination-oriented DAGs (DODAG) where a sink node works as the root of the DAG which is responsible to start forming a network topology. The DAG root broadcasts a DODAG information object (DIO) control message, which contains its rank and ID to other nodes in the network. When an intermediate node receives the DIO message, it replies to the root node with destination advertisement object (DAO) for joining the DODAG. Then, the intermediate node computes and updates its own rank and sends a DIO message with its rank to all neighbours. This process continues until the DIO message reaches the leaf nodes. When a node receives a DIO message from more than one neighbour, it selects its parent with best rank. Also, when a node does not receive a DIO message within a specific time, it starts to send a DODAG information solicitation (DIS) message to solicit DIO message from neighbours. Figure 1.5 shows RPL network topology construction process.

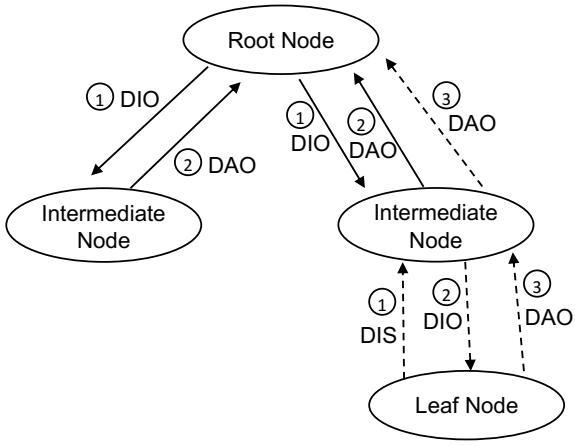

Fig. 1.5 DODAG construction process [19]

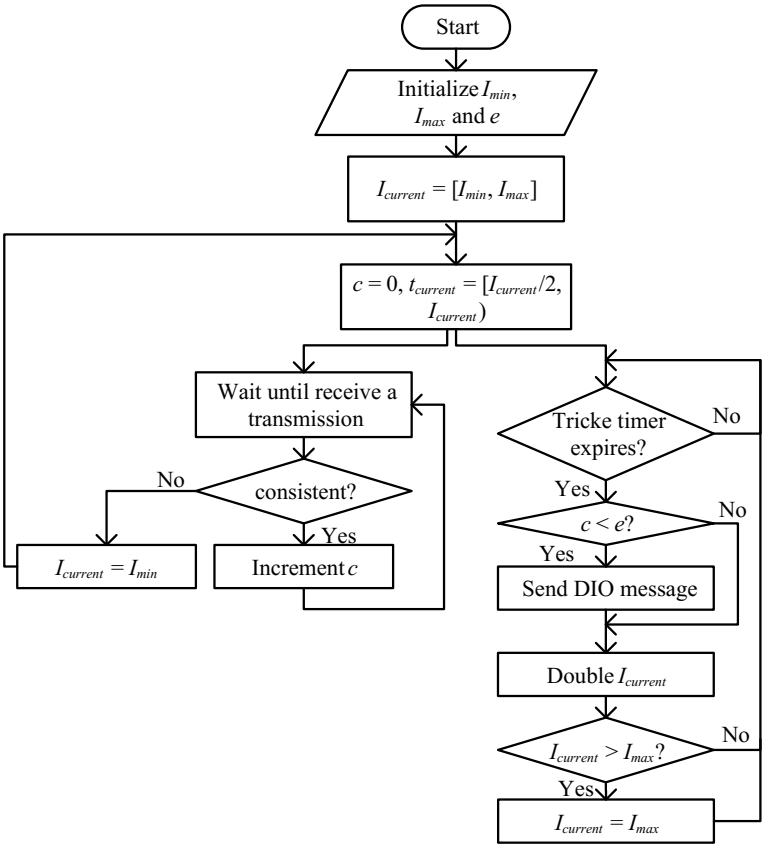

Fig. 1.6 Trickle algorithm

The DIO transmission strategy is controlled by the 'Trickle Algorithm' [20]. The Trickle algorithm maintains a Trickle timer which runs for a defined interval and it has three parameters: minimum interval size I_{min}, maximum interval size I_{max} and a redundancy constant $e > 0$. When the algorithm starts to execute, it sets the current interval size $I_{current}$ to a value in the range of $[I_{min}, I_{max}]$. When the interval begins, the algorithm resets a counter c to 0 and sets time within the current interval $t_{current}$ to a random point in the range $[I_{current}/2, I_{current})$. Whenever the algorithm receives a transmission that is'consistent', it increments c. If the algorithm receives a transmission that is 'inconsistent' and $I_{current}$ is greater than I_{min}, it resets the Trickle timer. At time $t_{current}$, a DIO message is sent if and only if c is less than e. When $I_{current}$ expires, the algorithm doubles the interval length such that it does not exceed I_{max}. Figure 1.6 shows the Trickle algorithm flowchart. The following packets and events are considered inconsistencies in RPL; otherwise it is consistent:

- When routing loops are detected.

- When a node receives a multicast DIS.
- When a node joins a new DODAG.

1.2.4 Adaptation Layer

The IETF 6LoWPAN working group was started in 2007 to address the challenges of enabling wireless IPv6 communication over IEEE 802.15.4 low-power radio with devices of limited power, limited memory, low bandwidth. The 6LoWPAN working group has developed a new layer called the adaptation layer which is located between the network layer and the data link layer to enable transmission of IPv6 packets over an IEEE 802.15.4 link. The adaptation layer has three main functions: (i) IPv6 header compression; (ii) IPv6 fragmentation and reassembly; (iii) routing. As the IEEE 802.15.4 frame overhead is 25 bytes without security support (which needs 21 extra bytes), the remaining frame size at the MAC layer is 102 bytes without security and 81 bytes with security support. For an IPv6 header of 40 bytes and a UDP header of 8 bytes, there is a maximum of only 54 bytes for application payload. Therefore, IPv6 header compression is very important to reduce header overhead and increase application payload space. The Request for Comment (RFC) 6282 [21] defines how to compress the IPv6 and UDP headers efficiently by using IP header compression (IPHC) and next header compression (NHC) methods.

The IEEE 802.15.4 defines the maximum transmission unit (MTU) to be 127 bytes, while IPv6 requires packet transmission with MTU of 1280 bytes. Therefore, the next major function of the adaptation layer is IPv6 fragmentation and reassembly. When an IPv6 packet does not fit into a single IEEE 802.15.4 data frame, the packet is divided into fragments where each fragment is sent over a single IEEE 802.15.4 frame. When all fragments are received at the other end, the IPv6 packets are reassembled and delivered up to the network layer. RFC 4944 [22] specifies how an IPv6 packet is fragmented into a FRAG1-type fragment and a number of FRAGN-type fragments. FRAG1 contains the IPv6 compressed header and part of the payload while FRAGN fragments are sent subsequently and contain the remaining payload. Besides the two functions described, the adaptation layer supports the mesh-under routing scheme to forward packets inside the 6LoWPAN network.

1.2.5 MAC and Physical Layers

IEEE 802.15.4 [23] is a standard, which defines the physical layer and the MAC layer for low-rate wireless personal area networks (LR-WPANs). The standard has been used as a basis for different networks, e.g. ZigBee, ISA100.11a, WirelessHART and 6LoWPAN. The IEEE 802.15.4 defines two types of devices which can participate in the network; a full-function device (FFD), which has full levels of functionality

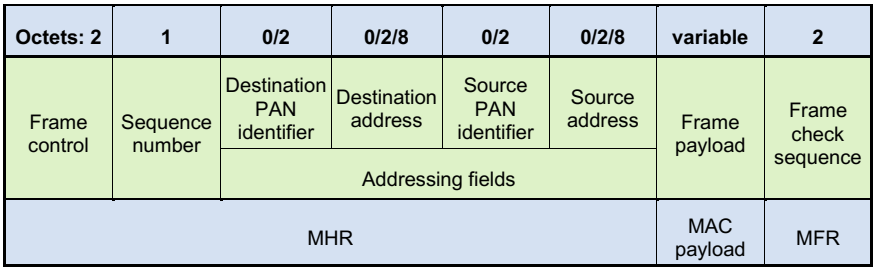

Fig. 1.7 General MAC frame structure [23]

Table 1.1 IEEE 802.15.4 frequency bands and data rates [23]

PHY (MHz)	Frequency band (MHz)	Modulation	Bit rate (kbps)	Symbol rate (ksymbol/s)	Number of channels
868	868–868.6	BPSK	20	20	1
915	902–928	BPSK	40	40	10
2450	2400–2483.5	O-QPSK	250	26.5	16

and can serve as a coordinator, and a reduced-function device (RFD) which has more limited functionality.

The MAC layer has the following features: beacon management, channel access, guaranteed time slots (GTS) management, frame validation, acknowledged frame delivery, association and disassociation. The general MAC frame format is shown in Fig. 1.7. The first field is frame control which is used to specify the frame type: data frame, MAC command frame, ACK frame or beacon frame. Next, the sequence number field is used to match the ACK frame and the addressing fields contain addresses of source and destination of the MAC frame. The frame check sequence (FCS) is a 16-bit cyclic redundancy check (CRC) used for error detection. The IEEE 802.15.4 defines two types of channel access mechanism: non-beacon enabled, which uses un-slotted CSMA/CA, and beacon-enabled mode where slotted CSMA/CA is used. The PHY layer provides the following services: activation and deactivation of the radio transceiver, energy detection of current channel, LQI, channel selection, clear channel assessment (CCA) and transmitting and receiving packets through the wireless channel. The radio can operate at one of three free-licensed bands: 868 MHz (Europe), 915 MHz (North America) or 2450 MHz (worldwide) as summarized in Table 1.1.

1.3 Research Contributions

The contributions of this thesis are as follows:

1. A review of performance metrics, operating systems and simulators used to evaluate and test the proposed congestion control mechanisms has been given.

Numerous papers designing congestion- control algorithms and mechanisms for WSNs and 6LoWPAN networks based on traffic control, resource control and hybrid schemes have been reviewed. The aim of this review is to

- Highlight and discuss the differences between congestion- control mechanisms in WSNs and 6LoWPAN networks and explains whether congestion control approaches for WSNs are suitable and valid for 6LoWPAN networks.
- Give some potential directions in future work for designing a novel congestion control mechanism, which should build upon the 6LoWPAN protocol stack and its characteristics and take into account the IoT application requirements.

2. Analytical modelling of congestion for 6LoWPAN networks through Markov chain and queuing theory has been performed. Also, a comprehensive congestion analysis for 6LoWPAN networks through simulations and real experiments (using 10 CM5000 TelosB sensor nodes) with different scenarios and various parameters has been conducted. The analysis results show that

   - The majority of packets are lost due to buffer overflow as compared to channel packet loss when congestion occurs.
   - It is important to set the value of reassembly timeout parameter to a small value within high data rate networks.
   - There is a similar performance for simulation and outdoor experiments. However, indoor experiment results have the same trend as simulation and outdoor results but slightly different absolute values since parameters such as nodes position, enclosed environment and interfering wireless signals (e.g. Wi-Fi) influenced performance negatively.

3. A new RPL routing metric called Buffer Occupancy (BO) and a new RPL objective function called Congestion-Aware Objective Function (CA-OF) have been proposed. With BO and CA-OF, packets are forwarded through less congested nodes and paths and therefore packet drops reduce highly. It is shown that CA-OF performs better in the presence of congestion in terms of the number of lost packets, throughput, energy consumption and packet delivery ratio as compared to the existing objective functions.
4. A congestion control game for mitigating congestion in 6LoWPAN networks using non-cooperative game theory has been designed. The node's payoff function is formulated to achieve the node demand (preference) for sending high data rate (utility function) and the desirable fairness among leaf nodes according to their priorities (priority cost function), while alleviating and mitigating congestion in the network (congestion cost function). By using the formulated game, we have proposed a novel and simple congestion control algorithm called game theory based congestion control framework (GTCCF). The proposed framework is aware of node priorities and application priorities to support the IoT application requirements. It is shown that GTCCF improves packets loss, throughput, end-to-end delay, energy consumption and weighted fairness index.
5. A new congestion alleviation algorithm called optimization- based hybrid congestion alleviation (OHCA) is proposed. OHCA provides a hybrid solution to

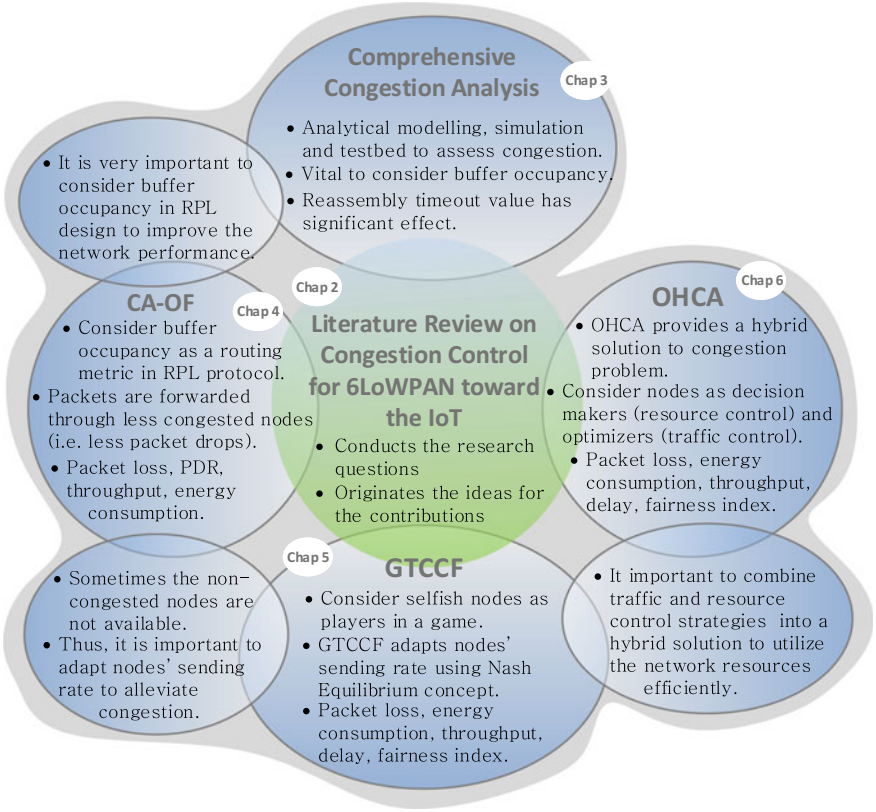

Fig. 1.8 Thesis contributions

the congestion problem in 6LoWPAN networks by combining traffic control and resource control strategies to utilize the network resources effectively. The proposed algorithm first applies the resource control strategy, which searches for the non-congested path. If the resource control method cannot be applied, then the traffic control strategy is executed to reduce the number of injected packets into the network by using optimization theory. Thus, OHCA utilizes the advantages of both strategies by bridging these two methods for congestion control and providing the optimal solution. Also, the proposed algorithm is aware of node priorities and application priorities to support the IoT application requirements.

Figure 1.8 illustrates the major contributions of this thesis and how the chapters are linked.

1.4 Thesis Outline

The remainder of this thesis is organised as follows:

Chapter 2 provides a review of related work on congestion control in WSNs and 6LoWPAN networks. First, an overview of why, how and where congestion occurs is provided and also how to solve congestion is explained. Then, information about performance metrics used to evaluate the proposed congestion control schemes and a short review of operating systems and simulators used to test and evaluate the proposed algorithms in WSNs and 6LoWPAN networks are given. Next, the chapter reviews numerous congestion control algorithms and mechanisms for WSNs and 6LoWPAN networks. Finally, this chapter discusses key issues addressed in the previous work, gives directions for future work and draws conclusions.

Chapter 3 presents simulation environment setup and provides a comprehensive congestion analysis for 6LoWPAN networks through analytical modelling, simulations and testbed. First, the simulation setup used in the experiments (i.e. Contiki OS and Cooja simulator) is presented as well as simulation parameters used in the simulations are described. Second, an analytical modelling of congestion for 6LoWPAN is developed by using Markov Chain and Queuing Theory and it is validated via simulation under various parameters and different scenarios. Third, an extensive congestion analysis for 6LoWPAN networks through simulations with different scenarios and various parameters is presented. Finally, a testbed-based congestion analysis for 6LoWPAN with different scenarios (indoor and outdoor) and various parameters is presented.

Chapter 4 introduces a new RPL objective function called congestion-aware objective function (CA-OF) which works efficiently when congestion occurs by selecting less congested paths. First, a brief overview of RPL is given as well as a literature review of related work about the proposed objective functions in RPL is provided. Second, a new objective function with a new metric, buffer occupancy, is proposed. Finally, The proposed objective function is tested and evaluated on three different network scenarios through simulation and compared with three other objective functions.

Chapter 5 presents a novel and simple congestion control mechanism called game theory based congestion control framework (GTCCF) specially tailored for IEEE 802.15.4, 6LoWPAN networks. First, the congestion problem is formulated as a non-cooperative game framework where the nodes (players) behave uncooperatively and demand high data rate in a selfish way. Then, the existence and uniqueness of Nash equilibrium is proved and the optimal game solution is computed by using Lagrange multipliers and KKT conditions. Next, the implementation of the congestion control game in 6LoWPAN networks is provided. Finally, The proposed congestion control framework is tested and evaluated on different network scenarios through simulation and compared with two other algorithms.

Chapter 6 presents a novel congestion control algorithm called optimization-based hybrid congestion alleviation (OHCA), which combines traffic control and resource

control strategies into a hybrid solution to utilize the positive aspects of each strategy and efficiently use the network resources. First, the network setup and problem formulation are introduced. Second, a multi-criteria optimization approach to combine three routing metrics is used to develop a new objective function called MADM-OF which addresses and solves the parent selection problem in 6LoWPAN networks within congestion. Third, a new traffic control mechanism called NUM-TC is proposed to adapt the source nodes' sending rate by using the NUM framework and optimization theory when the resource control strategy cannot be applied. Next, the implementation of the hybrid congestion control algorithm in 6LoWPAN networks is provided. Finally, the proposed algorithm is tested and evaluated on different network scenarios through simulation and compared with a traffic control based algorithm and a resource control based algorithm.

Chapter 7 concludes this thesis and briefly describes some future research directions in the field of congestion control towards the Internet of Things.

1.5 List of Publications

The following publications have emanated from the work of this research:

- *Journal Papers*

1. Al-Kashoash HAA, Hafeez M, Kemp AH (2017) Congestion control for 6LoWPAN networks: a game theoretic framework. Published in IEEE Internet Things J 4(3):760–771. 10.1109/JIOT.2017.2666269.
2. Al-Kashoash HAA, Kharrufa H, Al-Nidawi Y, Kemp AH, Congestion control for wireless sensor and 6LoWPAN networks: toward the internet of things. Submitted in Elsevier J Netw Comput Appl.
3. Al-Kashoash HAA, Amer HM, Mihaylova L, Kemp AH, Optimization based hybrid congestion alleviation for 6LoWPAN networks. Accepted with Minor Revisions in IEEE Internet J.
4. Al-Kashoash HAA, Hassen F, Kharrufa H, Kemp AH, Analytical modelling of congestion for 6LoWPAN networks. Accepted with Minor Revisions in Elsevier ICT Express J.

- *Conference Papers*

5. Al-Kashoash HAA, Al-Nidawi Y, Kemp AH (2016) Congestion analysis for low power and lossy networks. Published in Wireless telecommunications symposium (WTS). IEEE. 10.1109/WTS.2016.7482027.
6. Al-Kashoash HAA, Al-Nidawi Y, Kemp AH (2016) Congestion-aware RPL for 6LoWPAN networks. Published in Wireless telecommunications symposium (WTS). IEEE. "***Best Student Paper Award***". 10.1109/WTS.2016.7482026.

- *Co-Authored Publications*

7. Kharrufa H, Al-Kashoash HAA, Al-Nidawi Y, Quezada Mosquera M, Kemp AH (2017) Dynamic RPL for multi-hop routing in IoT applications. In: Published in 13th wireless on-demand network systems and services conference. IEEE/IFIP. 10.1109/WONS.2017.7888753.
8. Amer HM, Al-Kashoash HAA, Hawes M, Chaqfeh M, Kemp AH, Mihaylova L, Centralized simulated annealing for alleviating vehicular congestion in smart cities. Submitted in IEEE Trans Mob Comput.

References

1. Shelby Z, Bormann C (2009) 6LoWPAN: The wireless embedded internet. Wiley, New Jersey
2. Li S, Da Xu L, Zhao S (2015) The Internet of things: a survey. Inf Syst Front 17(2):243–259
3. Atzori L, Iera A, Morabito G (2010) The Internet of things: a survey. Comput Netw 54(15):2787–2805
4. Alcaraz C, Najera P, Lopez J, Roman R (2010) Wireless sensor networks and the internet of things: do we need a complete integration? In: Proceedings of 1st international workshop on the security of the internet of things (SecIoT'10), Tokyo (Japan), 29th November, 2010
5. Khalil N, Abid MR, Benhaddou D, Gerndt M (2014) Wireless sensors networks for internet of things. In: Proceedings of IEEE 9th international conference on intelligent sensors, sensor networks and information processing (ISSNIP). IEEE (2014), pp 1–6
6. Ghaffari A (2015) Congestion control mechanisms in wireless sensor networks: a survey. J Netw Comput Appl 52:101–115
7. Kafi MA, Djenouri D, Ben-Othman J, Badache N (2014) Congestion control protocols in wireless sensor networks: a survey. IEEE Commun Surv Tutor 16(3):1369–1390
8. Michopoulos V, Guan L, Oikonomou G, Phillips I (2011) A comparative study of congestion control algorithms in IPv6 wireless sensor networks. In: Proceedings of international conference on distributed computing in sensor systems and workshops (DCOSS). IEEE (2011), pp 1–6
9. Shelby Z, Hartke K, Bormann C (2014) The constrained application protocol (CoAP). In: IETF RFC 7252 (2014)
10. Tanenbaum AS, Wetherall DJ (2013) Computer networks: Pearson New, international edn. University of Hertfordshire, Pearson Higher Education
11. Gomez C, Kim E, Kaspar D, Bormann C (2012) Problem statement and requirements for IPv6 over low-power wireless personal area network (6LoWPAN) routing. In: IETF RFC 6606
12. Chowdhury AH, Ikram M, Cha H-S, Redwan H, Shams S, Kim K-H, Yoo S-W (2009) Route-over vs mesh-under routing in 6LoWPAN. In: Proceedings of the international conference on wireless communications and mobile computing: connecting the world wirelessly. ACM, pp 1208–1212
13. Yoo S, Lee J, Mulligan G (2007) Hierarchical routing over 6LoWPAN (HiLow), draft-daniel-6lowpan-hilow-hierarchical-routing-01. IETF network working group, Technical report, Internet-Draft
14. Kim K, Park SD, Montenegro G, Yoo S, Kushalnagar N (2007) 6LoWPAN Ad Hoc on-demand distance vector routing (LOAD). In: Internet Draft, draft-daniel-6lowpan-load-adhoc-routing-03, IETF
15. Kim K, Montenegro G, Park S, Chakeres I, Perkins C (2007) Dynamic MANET on-demand for 6LoWPAN (DYMO-low) routing. IETF, Internet-Draft
16. Winter T, Thubert P, Brandt A, Hui J, Kelsey R (2012) RPL: IPv6 routing protocol for low-power and lossy networks. In: IETF, RFC 6550

17. Tsvetkov T (2011) RPL: IPv6 routing protocol for low power and lossy networks. In: Sensor nodes–operation, network and application (SN), vol 59, p 2
18. Gaddour O, Koubâa A, Baccour N, Abid M (2014) OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol. In: Proceedings of 12th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt). IEEE, pp 365–372
19. Zhang T, Li X (2014) Evaluating and analyzing the performance of RPL in Contiki. In: Proceedings of the 1st international workshop on mobile sensing, computing and communication. ACM, pp 19–24
20. Levis P, Clausen T, Hui J, Gnawali O, Ko J (2011) The trickle algorithm. In: Internet engineering task force, RFC 6206
21. Hui J, Thubert P (2011) Compression format for IPv6 datagrams over IEEE 802.15. 4-based networks. In: IETF RFC 6282
22. Montenegro G, Kushalnagar N, Hui J, Culler D (2007) Transmission of IPv6 packets over IEEE 802.15.4 networks. In: IETF RFC 4944
23. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) (2003) Specifications for low-rate wireless personal area aetworks (LR-WPANs). IEEE Standard 802(15):4

Chapter 2
Background and Literature Review

2.1 Introduction

This chapter presents a comprehensive literature review on congestion control for WSNs and 6LoWPAN networks. The major contributions of this chapter are as follows:

- It gives a review of performance metrics, operating systems and simulators used to evaluate and test proposed congestion control mechanisms as well as explaining which operating systems and simulators support the 6LoWPAN protocol stack.
- The chapter reviews popular papers designing congestion control approaches and mechanisms for WSNs based on the congestion control method used to solve and mitigate congestion: traffic control, resource control and hybrid schemes.
- This chapter highlights and discusses the differences between congestion control mechanisms in WSNs and 6LoWPAN networks and explains whether congestion control approaches for WSNs are suitable and valid for 6LoWPAN networks.
- Furthermore, this chapter gives some potential directions in future work for designing a novel congestion control mechanism which should build upon the 6LoWPAN protocol stack and its characteristics and take into account the IoT application requirements.
- Recently, a number of survey papers have focused on congestion control approaches for WSNs only [1–10]. To the best of our knowledge, this is the first work that provides a comprehensive review of the existing congestion control algorithms in 6LoWPAN networks.

The remainder of the chapter is organized as follows: Sect. 2.2 gives a mathematical background about the techniques that employed in the thesis, e.g. game theory and optimization theory. Section 2.3 provides an overview of why, how and where congestion occurs and also explains how to solve congestion. Section 2.4 provides information about performance metrics used to evaluate the proposed congestion control schemes. Section 2.5 gives a short review of operating systems and

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_2

simulators used to test and evaluate the proposed algorithms in WSNs and 6LoWPAN networks. In Sects. 2.6 and 2.7, we review numerous congestion control algorithms and mechanisms for WSNs and 6LoWPAN networks, respectively. Section 2.8 discusses key issues addressed in the chapter and gives directions for future work. Finally, Sect. 2.9 draws conclusions.

2.2 Mathematical Background

In this section, we provide some background on mathematical techniques which are used and employed in Chaps. 5 and 6 to solve the congestion problem in 6LoWPAN networks.

2.2.1 Game Theory

Game theory is a branch of applied mathematics and applied sciences. It has been used in a variety of disciplines such as social sciences (e.g. economics), political science, biology, computer science, philosophy and recently communication networks [11]. Game theory can be divided into two major groups: non-cooperative and cooperative. If there is no negotiation or mediation between the players and they select strategies independently from each other, then the game is non-cooperative; otherwise, it is cooperative [12]. Here, we are focusing on non-cooperative game theory. Non-cooperative game theory provides an analytical framework suited for characterizing the interactions among several decision-makers (players) with partially or totally conflicting interests over the outcome of a decision process which is affected by their actions. It has three components: set of players, their strategies and the payoff functions. Formally, non-cooperative game theory is defined as follows:

Definition 2.2.1 A non-cooperative game in strategic form is a triplet $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$, where

- M is a finite set of players, i.e. $M = \{1, \ldots, M\}$.
- S_k is the set of available strategies for player k.
- $\Phi_k : SS \rightarrow \mathbb{R}$ is the payoff function for player k, with $SS = \prod_{k=1}^{m} S_k$ (Cartesian product of the strategy sets).

When a player selects a strategy in a deterministic way with a probability of 1, this strategy called 'pure strategy'. However, a player may be able to select each pure strategy with a certain probability which is the basis of the concept of a 'mixed strategy'. Many concepts are used for solving a non-cooperative game such as Nash equilibrium which is the most accepted solution concept introduced by John Nash. Nash equilibrium gives a strategy choice for all players such that no player can

increase his payoff by changing its current strategy. Formally, Nash equilibrium is defined as follows:

Definition 2.2.2 A pure strategy Nash equilibrium of a non-cooperative game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$ is a strategy profile $\boldsymbol{s}^* \in SS$ such that $\forall k \in M$ we have the following:

$$\Phi(s_k^*, s_{-k}^*) \geq \Phi(s_k, s_{-k}^*), \tag{2.1}$$

$\forall s_k^*, s_k \in S_k, s_k^* \neq s_k, \forall k \in M$ where s_{-k}^* is the vector of strategies of all players except player k.

2.2.2 Multi-attribute Decision-Making

Multi-attribute decision-making (MADM) is a discipline aimed for supporting decision-makers which are faced conflicting alternatives to make an optimal decision. Decision-making process involves a series of steps: identifying the problems, constructing the preferences, evaluating the alternatives and determining the best alternative [13].

In order to deal with an MADM problem, the first step is to determine how many attributes exist in the problem (i.e. identifying the problems). The attributes can be classified into two main categories: cost attributes and benefit attributes. With cost attributes, the lower value is better (e.g. delay); however, with benefit attributes, the higher value is better (e.g. throughput). Next, we need to collect the appropriate information in which the preferences of decision-maker can be correctly reflected and considered (i.e. constructing the preferences). The third step is to build a set of possible alternatives in order to guarantee that the goal will be reached (i.e. evaluating the alternatives). The final step is to select an appropriate method to evaluate and outrank the possible alternatives (i.e. determining the best alternative). There are many common methodologies for MADM such as simple additive weighting (SAW), the technique for order preference by similarity to ideal solution (TOPSIS), analytical hierarchy process (AHP), grey relational analysis (GRA), etc. [14]. An MADM problem with m alternatives and n attributes can be represented by a decision matrix D as follows:

$$D = \begin{array}{c} \begin{matrix} g_1 & \cdots & g_j & \cdots & g_n \end{matrix} \\ \begin{bmatrix} r_1(a_1) & \cdots & r_j(a_1) & \cdots & r_n(a_1) \\ \vdots & & \vdots & & \vdots \\ r_1(a_i) & \cdots & r_j(a_i) & \cdots & r_n(a_i) \\ \vdots & & \vdots & & \vdots \\ r_1(a_m) & \cdots & r_j(a_m) & \cdots & r_n(a_m) \end{bmatrix} \end{array}, \tag{2.2}$$

where $r_j(a_i)$ represents the value of jth attribute for the ith alternative and g_j represents the weight of jth attribute for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

2.2.3 Network Utility Maximization

Network utility maximization (NUM) is a framework used for allocating the fair amount of network resources (e.g. bandwidth) among the nodes in order to maximize the overall network utility. The NUM was introduced by Kelly et al. [15] in 1998 for wired networks and it has already numerous applications in wired and wireless network optimization [16]. NUM studies and formulates the problem of resource allocation in the network as a constrained optimization problem. Consider a set of z nodes shared a link with capacity of c and each node, l, has a certain utility $U_l(\lambda_l)$ when transmitting at rate λ_l. The utility can be interpreted as the level of satisfaction that a node profits when its transmission rate is λ_l. Many types of utility function are commonly used such as exponential, logarithmic, linear and sigmoidal [17]. Using NUM framework, the problem of allocating the link capacity, c, among z nodes can be expressed as follows [18]:

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{maximize}} \quad & \sum_{l=1}^{z} U_l(\lambda_l), \\ \text{subject to} \quad & \sum_{l=1}^{z} \lambda_l \leq c, \\ & \lambda_l \geq 0, \qquad \forall l = 1, 2, \ldots, z, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\lambda}$ is a vector consisting of $\lambda_1, \ldots, \lambda_l, \ldots, \lambda_z$.

2.3 Congestion in WSNs and 6LoWPANs

WSN is a network formed by a large number of sensor nodes that are spatially distributed and organized to monitor physical and environmental conditions, e.g. temperature, sound, vibration, pressure and light. As WSNs are connected to the Internet through 6LoWPAN to form the IoT, the WSN applications are increasingly varied and sensor nodes are everywhere in vehicles, smartphones, factories, building, seas, forests, etc. [19]. Sensor nodes have limited resources with regard to memory, computation capabilities, bandwidth and power supply. Due to these limitations and constraints, the traditional congestion control schemes used on the Internet, i.e. TCP, cannot be applied to WSNs and designing a new congestion control scheme is challenging [1]. Congestion occurs when many sensor nodes start to send their packets concurrently at high data rate or when a node relays many flows across the network. Congestion has a significant impact on quality of service (QoS) parameters and the energy efficiency of sensor nodes [1]. Moreover, congestion increases packet loss, degrades throughput and increases end-to-end delay.

In WSNs and 6LoWPAN networks, congestion occurs and is created at two levels and positions: node-level congestion (buffer overflow) and link-level

congestion (link contention and collision) [1, 2, 5, 10]. When the packet arrival rate is higher than packet departure rate at a sensor node, buffer overflow occurs if there is insufficient space to store the incoming packets. This leads to high packet loss rate at the node and hence increases energy consumption. On the other hand, when multiple nodes located in the same transmission range transmit simultaneously, link congestion occurs where packets are lost due to interference. This reduces throughput and increases the number of retransmission and, therefore, extra energy is consumed due to packet retransmission.

Congestion control in wireless networks is treated differently from the techniques and mechanisms used for wired networks [20]. In wired networks, an end-to-end approach is typically used where source nodes receive congestion feedback from the destination which is responsible for detecting congestion. In the end-to-end approach, the congestion control mechanism exists on a source-to-destination basis and the intermediate nodes do not take any action to alleviate congestion. On the other hand, a hop-by-hop approach is widely used in wireless networks. The hop-by-hop scheme operates on a node-by-node basis where loss recovery and congestion notification are implemented locally at intermediate nodes which react immediately to congestion occurrence [21]. As wireless links are unreliable, it is impractical to support an end-to-end connection to transmit packets in wireless links [22]. Also, the major benefits of the hop-by-hop approach are that it reacts to congestion occurrence much faster than the end-to-end scheme. Therefore, the majority of congestion control algorithms in WSNs and 6LoWPAN networks use the hop-by-hop approach.

The process of congestion control in WSNs and 6LoWPAN networks includes three steps: congestion detection, congestion notification, and congestion control and mitigation [1, 2, 10] as shown in Fig. 2.1.

1. **Congestion detection**: This step refers to the process of detecting congestion and specifying its location. Many congestion detection mechanisms have been proposed and used in WSNs and 6LoWPAN networks, e.g. buffer occupancy, channel load, combination of buffer occupancy and channel load, packet service time, packet loss and delay [23].

    - **Buffer occupancy**: Each sensor node has a buffer which is used to store packets before they are transmitted to the wireless channel. When the buffer occupancy exceeds a threshold value, a congestion alarm is raised. The buffer threshold method is a simple and good indication of congestion. When the buffer occupancy at intermediate nodes exceeds a threshold value, they send back a notification message piggybacked with congestion information to the source nodes.
    - **Channel load**: It measures the packet load on the wireless channel. Channel load or channel busyness ratio is the ratio of time intervals when the channel is busy due to successful transmission or collision to the total time. The channel load is measured by performing CCA function which responds with 0 or 1 when channel is free or busy, respectively. The frequency of busyness measures and reflects the channel load level. However, sampling the wireless channel by performing CCA increases energy consumption.

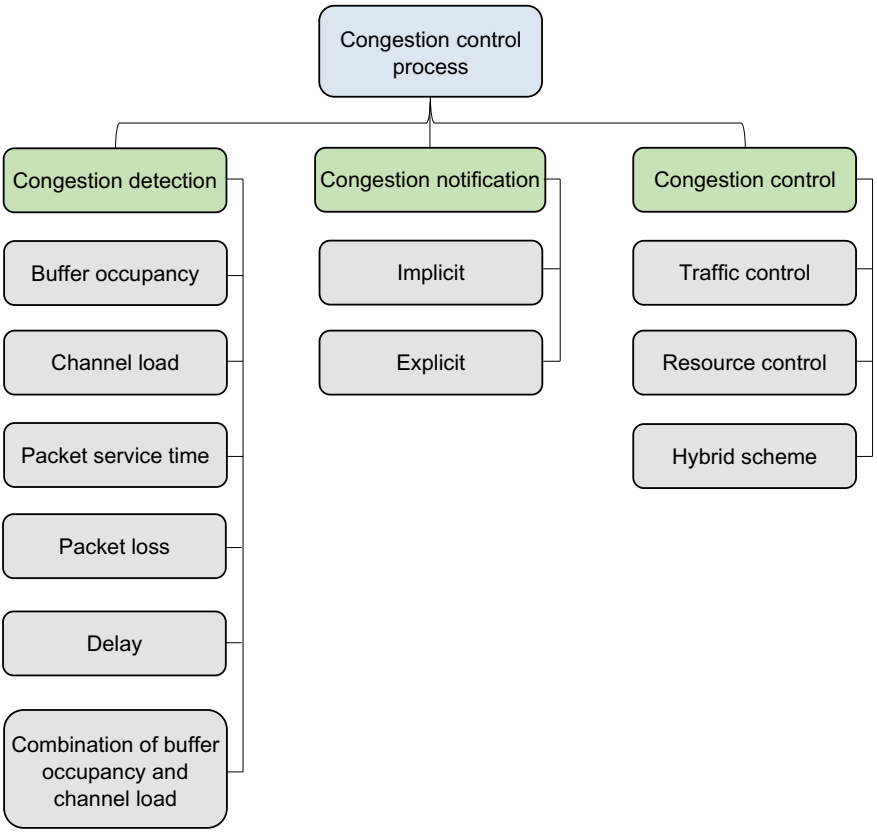

Fig. 2.1 Congestion control steps

- **Combination of buffer occupancy and channel load**: In this method, the above two schemes are combined and congestion is detected either at the node's buffer or in the wireless channel.
- **Packet service time**: It is the time interval between packet arrival at the MAC layer and its successful transmission. It equals one-hop delay and covers packet waiting time at the MAC layer and packet transmission time.
- **Packet loss**: This method is used if ACK is activated. When a sender does not receive an ACK, it assumes that congestion occurs. However, packet loss can be caused by packet errors rather than collision at the wireless channel.
- **Delay**: It is the time since a packet is generated at the sender until its successful reception at the next-hop receiver or the end point receiver. However, using the delay as indicator for congestion may be misleading when radio duty cycle (RDC) is applied at the MAC layer that causes long delay for the packets.
- **Others** such as difference between input and output traffic rates, packet inter-arrival time, weighted moving average of queue length and traffic rate.

2. **Congestion notification**: When congestion is detected, the congested nodes should notify source nodes which nodes cause congestion in the network. The congestion information is sent either implicitly or explicitly.
   - **Implicit notification**: Using this method, the congestion information is piggybacked in a data packet header or in ACK packets. This method avoids injection of unnecessary overhead packets to the network which is already congested.
   - **Explicit notification**: In this method, extra overhead packets are sent by congested nodes to inform other nodes about their congestion state. Using this technique, the congestion condition is increased by injecting more overhead packets into the network.
3. **Congestion control**: After the source nodes receive the congestion information, actions should be taken to reduce and mitigate congestion in the network. Congestion is solved and mitigated using two ways either rate adjustment (traffic control) or selection of an alternative non-congested path (resource control) to forward packets to destination nodes.
   - **Traffic control**: In this method, congestion is controlled by reducing the number of injected packets into the network where source nodes reduce their sending rate to a specific value. There are two approaches for traffic rate adaptation: the window-based method and the rate-based method. In the window-based technique, a source node checks the available bandwidth by slowly increasing the congestion window. When congestion is detected, the congestion window is reduced significantly. An example of this method is additive-increase multiplicative-decrease (AIMD) mechanism where the congestion window is increased linearly and decreased exponentially after congestion occurs. In the rate-based scheme, source nodes check and estimate the available bandwidth. Then, they adjust sending rate based on the calculated available bandwidth. An example of this method is the available bandwidth BW_a equation used in [24] as follows:

$$BW_a = \begin{cases} 0 & \text{if } c_b \geq th_b \\ BW(th_b - c_b)\overline{data}/T_s & \text{if } c_b < th_b \end{cases}, \tag{2.4}$$

     where BW is the transmission rate in bits per second for the data packet, $\overline{data}$ is the average payload size measured by the channel occupancy time (in second), T_s is the average time of successful transmission at the MAC layer (in second), c_b is channel busyness ratio and th_b is channel occupancy threshold (e.g. 92%). However, in case of event-based and time-critical applications where packets carry very important information that should be delivered in time, reducing the valuable data rate is not desirable and impractical.
   - **Resource control**: To avoid the drawback of the traffic control scheme (i.e. reducing the valuable data rate in time-critical applications), an alternative method called resource control is used. In this method, when congestion

occurs, packets are forwarded to destination nodes through alternative uncongested paths without reducing the sending rate. The packet delivery ratio with this scheme is higher than in case of the traffic control method.

- **Hybrid scheme**: Some algorithms combine the above two methods to mitigate congestion in the network. The algorithm first searches for uncongested paths to forward packets using the resource control method. If the uncongested paths are available, then the resource control method is executed. Otherwise, the sending rate is reduced by applying the traffic control method.

2.4 Performance Evaluation Metrics

Performance evaluation is used to determine the effectiveness and efficiency of the proposed algorithms and protocols. The common performance metrics used by congestion control approaches in WSNs and 6LoWPAN networks are energy tax, normalized reliability, energy consumption, throughput, Jain's fairness index, latency, buffer drop rate, packet loss rate, queue length, packet delivery ratio, source rate and fidelity index. A brief description of these metrics is given next.

- **Energy tax** is the ratio between the total number of dropped packets and the total number of received packets at the sink node [25]. As packet transmission and reception consume the main portion of a node's energy, the number of dropped packets per received packet directly indicates the energy efficiency.
- **Normalized reliability** is defined as the ratio between the number of received data packets in an interval at sink node to the number of data packets required for reliable event detection [26].
- **Energy consumption** is the total amount of spent energy due to communication including transmission, reception, idle state and sleep state. This metric is an indication of the energy efficiency of the algorithms [24].
- **Throughput** is the total number of successfully received bits at sink node per unit time (typically every second) [24]. Some papers count the total number of packets received by the server (sink node) and call it **goodput** [27].
- **Jain's fairness index** is an indication of fair allocation of network resources (e.g. bandwidth) among nodes in the network, e.g. the sink node receives equal number of packets from each node [27–29] as follows:

$$\text{Jain's fairness index} = \frac{\left[\sum_{i=1}^{n} th_i\right]^2}{n\sum_{i=1}^{n}(th_i)^2}, \tag{2.5}$$

where th_i is throughput of node i and n is the number of nodes such that Jain's fairness index $\in [0, 1]$.

Some papers use **weighted fairness index** to achieve different throughputs according to nodes' priority and importance [30, 31] as follows:

$$\text{Weighted fairness index} = \frac{\left[\sum_{i=1}^{n} th_i p_i\right]^2}{n \sum_{i=1}^{n} (th_i p_i)^2}, \tag{2.6}$$

where p_i is the priority of node i such that weighted fairness index $\in [0, 1]$.

- **Latency** is the amount of time measured from the application-level packet transmit on the node to the moment at which the final destination receives the packet [29]. Some papers call it **end-to-end delay** [32]. Some algorithms are evaluated using **hop-by-hop delay** which is the time from a child node to its parent (one hop only) [33].
- **Buffer drop rate (queue loss ratio)**: This metric measures the probability that a packet will be dropped due to buffer overflow [29, 34]. Some papers call it **rejection rate** [35]. This metric does not take into account the wireless channel loss [36].
- **Packet loss rate (packet loss ratio)** is the ratio between the total number of lost packets and the total number of sent packets in the network [28]. Some papers call it **loss probability** [35]. This metric takes into account the total number of lost packets due to buffer overflow and wireless channel loss [32, 37].
- **Queue length (queue level)**: This metric shows the average number of packets stored in the nodes' buffer over time [38, 39].
- **Packet delivery ratio** is the ratio between the number of successfully received packets at the sink node to the total number of sent packets in the network [28, 40]. Some papers call it **packet reception rate** [33, 38].
- **Source rate** is the total number of packets generated by source nodes per second [24, 41].
- **Fidelity index** is the ratio between the actual number of delivered packets per unit time to applications and the required (desired) number of packets per unit time received by the applications [42].
- **Others** such as **control overhead packets** [34, 35, 41], **network efficiency** [29, 43] and **hop count** [34, 44, 45].

2.5 Operating Systems and Simulators for WSNs and 6LoWPANs

It is very important to choose an appropriate tool for testing, analysing and evaluating a proposed algorithm performance. Real testbeds provide a better option for studying behaviour of the proposed algorithm in realistic environments and scenarios. TinyOS and Contiki OS are an excellent choice to examine and evaluate the proposed mechanisms as they are real, widely used operating systems supporting

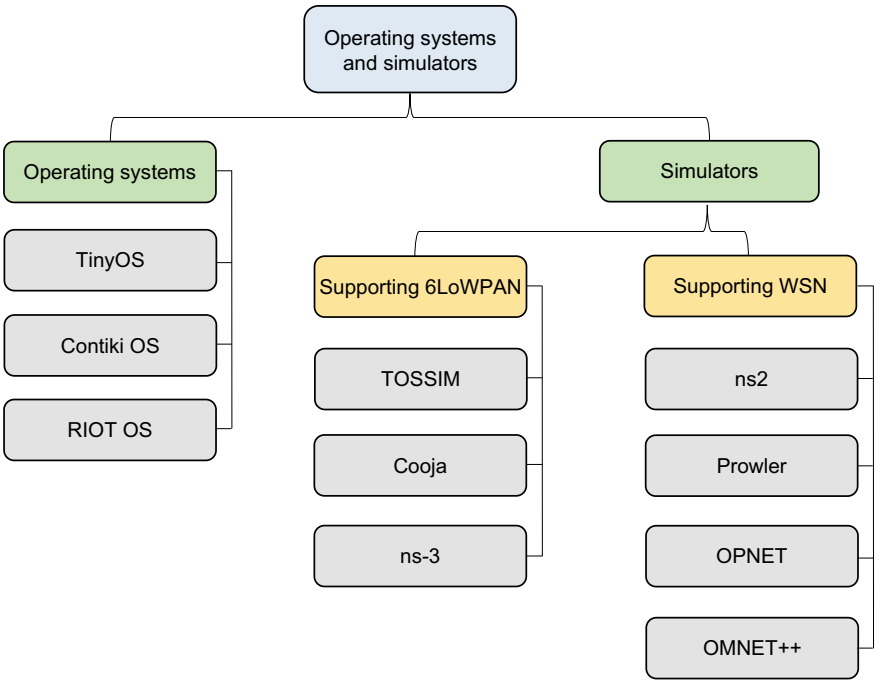

Fig. 2.2 Operating systems and simulators for WSN and 6LoWPAN networks

the 6LoWPAN protocol stack and the IoT. However, testing and evaluating through a real testbed is costly, time-consuming and debugging challenge. Therefore, simulators are good alternatives that provide effective, low-cost, scalable, time-limited and ease-of-implementation tools. It is vital to choose a simulator that supports the 6LoWPAN protocol stack and the IoT, e.g. TOSSIM, Cooja and ns-3. Sometimes, TOSSIM and Cooja are considered emulators as they execute the same code on real motes [46].

Operating systems, testbeds and simulators are effective tools to evaluate the performance of proposed algorithms and mechanisms. Many real operating systems and simulators exist that support WSNs and the 6LoWPAN protocol stack such as TinyOS, Contiki OS, TOSSIM, Cooja, ns2, ns-3, Prowler, OPNET and OMNET++ as shown in Fig. 2.2. A short review of these operating systems and simulators used by researchers to evaluate the performance of congestion control algorithms in WSNs and 6LoWPAN networks is given below.

- **TinyOS** [47] is a tiny, flexible, open-source operating system designed for low-power, embedded, wireless devices. It was developed at the University of California in Berkeley. TinyOS and its programs are written in NesC (network embedded system C). TinyOS uses an event-driven programming model where the user applications are composed of three components: commands, events and tasks. One of the strengths of TinyOS is its support for a wide range of hardware platforms. TinyOS supports the 6LoWPAN protocol stack through BLIP

(Berkeley Low-power IP stack) which is the TinyOS implementation of a number of IP-based protocols, e.g. TinyRPL.

- **Contiki OS** [48] is an open-source operating system for the IoT where Contiki OS connects tiny, low-cost, low-power networked devices to the Internet. Contiki OS was the first operating system that provides IPv4 and IPv6 connectivity for sensor nodes [49]. Contiki was developed at the Swedish Institute of Computer Science by Adam Dunkles. A running Contiki OS consists of an event-driven kernel, libraries, program loader and a set of processes. A Contiki system is partitioned into two parts: the core, which consists of the kernel, the program loader, communication stack and device drives, and loaded programs which are loaded into the system by the program loader. Contiki OS supports three communication stacks: uIP TCP/IP, uIPv6 and Rime. The first two stacks provide IPv4 and IPv6 networking respectively, while the Rime stack is a set of lightweight protocols which are designed for low-power wireless networks. Also, Contiki OS provides a runtime, network-level, power profiling system called Powertrace [50] which uses state tracking to estimate and measure the energy consumption of each node and it is accurate up to 94%.
  Contiki OS has a set of unique features that can be summarized as follows [51]:
  - Contiki provides low-power Internet communication for constrained and limited resource devices, e.g. UDP, TCP and HTTP. Contiki supports IPv4 and IPv6 standards as well as the recent IETF low-power standard protocols such as 6LoWPAN, RPL and CoAP.
  - Contiki runs on a wide range of low-power wireless devices such as micaz, sky, seed-eye, msb430, cc2530dk, z1, win32, sensinode, wismote, etc.
  - Contiki is open-source software and its full source code is free.
  - Contiki OS uses a programming mechanism called protothread which is a low-overhead mechanism for concurrent programming [52].
  - Contiki provides a lightweight flash file system called Coffee where programs can be opened, closed, read and written to an external flash.
  - Contiki has a network simulator called Cooja which provides a simulation environment for large-scale networks with developing and debugging software.
  - Other features are efficient memory allocation, power consumption awareness, supporting dynamic module loading and running in a small amount of memory (10k RAM and 30k ROM).

- **RIOT OS** [53] is an open-source operating system designed and developed by an international community of companies, academia and hobbyists for the particular requirements of the IoT scenarios. It considers devices with minimal resources but eases development across a wide range of devices. RIOT OS implements a microkernel architecture inherited from FireKernel [54] that supports multi-threading with standard application programming interface (API). Also, RIOT OS supports C and C++ programming languages for enabling powerful libraries and providing a TCP/IP network stack. RIOT OS runs on several platforms including embedded devices, e.g. TelosB, Zolertia Z1, Arduino Due, etc., as well as personal computers. RIOT OS supports 6LoWPAN protocol stack, openWSN and Arduino API.

- **TOSSIM** [55] is a discrete-event simulator for TinyOS sensor networks. It is used for compiling a TinyOS application for the TOSSIM simulation framework rather than for a real sensor node implementation. This allows users (researchers) to examine, test and debug their algorithms and mechanisms in a controlled environment. TOSSIM includes models for the CPU, clocks, timers and radio components. However, TOSSIM does not model the real environment and it provides a radio abstraction of directed independent bit errors between two nodes. Independent bit errors mean longer packets have a higher probability of corruption and each packet's loss probability is independent. Also, it does not model the energy consumption and it supports only one hardware platform model (MicaZ).
- **Cooja** [56] is a cross-level, flexible Java-based simulator designed for simulating a network of sensor nodes which run Contiki OS [56]. Cooja simulates the operation of different types of real sensor motes such as Tmote Sky, Z1, WiSMote, MicaZ and ESB (embedded sensor board). Tmote Sky is used in our simulations and also in our demonstration tests. Cooja allows for simultaneous simulation at three different levels: application level, operating system level and machine code instruction level. Cooja implements a number of wireless channel models such as Unit Disk Graph Medium (UDGM)–Distance Loss and Directed Graph Radio Medium (DGRM). UDGM–Distance Loss is used in our simulation since interference is considered [57]. In UDGM–Distance Loss, the transmission range is modelled as a disk where all nodes inside the disk can transmit and receive packets with probability of SUCCESS_RATIO_TX and SUCCESS_RATIO_RX, respectively. Cooja has a very useful tool for development and debugging called TimeLine which shows the timeline for each node [58]. Timeline shows the power state of the node's radio transceiver: white indicates off, grey indicates on, blue indicates transmission and green indicates reception. Also, red is used to show radio interference when two or more simultaneous transmissions of nodes occur. However, the limitation of Cooja is that when the number of nodes exceeds the allowable limit, the simulation time becomes very long. Table 2.1 shows the simulation parameters used in our simulations and tests.
- **ns2** [59] is a discrete-event open-source simulator and it is one of the most popular network simulators. It provides a wide range of IP protocols, e.g. TCP/IP, routing and multicast protocols. It has an object-oriented design which allows users to design and implement new protocols. Also, it has an animation tool called network animator (Nam) used for viewing and visualizing packet traces and protocols behaviour. However, it has not been designed specially for WSNs as well as it does not support the 6LoWPAN protocol stack.
- **ns-3** [60] is an object-oriented open-source simulator similar to ns2. It was developed to replace its predecessor ns2. ns-3 provides a powerful tool for network modelling and optimization. It includes TCP/IP, IPv6, routing, IEEE 802.11, IEEE 802.15.4, WiMAX and Wi-Fi. Also, it supports the 6LoWPAN protocol stack.
- **Prowler** [61] is an event-driven probabilistic simulator developed for wireless networks. As it runs under the MATLAB environment, it provides a fast and easy way for prototyping applications. Prowler can run in two modes: deterministic and probabilistic. Also, it models the important aspects of all levels of the communica-

Table 2.1 Simulation parameters

Parameter	Value
Ratio model	UDGM—distance loss
Node type	Tmote Sky
Transmission range	50 m
Interference range	100 m

tion channel and application, e.g. radio channel and MAC layer. However, it does not consider and support the 6LoWPAN protocol stack.

- **OPNET** [62] is a commercial, generic, event-based simulation tool and it supports the C and Java programming languages. It contains a huge library of accurate models of commercial network hardware and protocols. Also, it supports a wide range of communication systems from local area networks to global satellite networks. OPNET provides powerful tools for building models, executing simulations and analysing output results. However, OPNET does not support the 6LoWPAN protocol stack.
- **OMNET++** [63] is an open-source, modular, discrete-event, C++ based simulator for modelling communication networks. OMNET++ provides deep analysis of network activities at packet level. An OMNET++ model consists of modules which communicate through message passing where simple modules can be grouped into compound modules in a hierarchical fashion with unlimited levels using a high-level language called network description (NED). However, OMNET++ was not designed specially for WSNs and it does not support the 6LoWPAN protocol stack. Recently, Kirsche and Hartwig [64] have developed a 6LoWPAN simulation model for OMNET++ by integrating Contiki's implementation into OMNET++.

2.6 Congestion Control Algorithms for WSNs

Numerous methods and different algorithms have been proposed in the congestion control literature for managing and mitigating congestion in WSNs. In this section, a discussion and review of algorithms according to the congestion control method (traffic control, resource control and hybrid scheme) as well as how each algorithm works are given.

2.6.1 Traffic Control Algorithms

This subsection reviews congestion control mechanisms which are based on the traffic control method where the source traffic rate is adjusted to reduce the number of injected packets into the network and, therefore, congestion can be mitigated. Table 2.2 summarizes these algorithms.

Table 2.2 Traffic control algorithms in WSNs

Algorithm	Congestion detection	Congestion notification	Implementation/(number of nodes)	Evaluation metrics	Compared with
CODA [25]	Buffer occupancy and channel load	Explicit	Simulation and real experiments (TinyOS)/(30–120 nodes)	Energy tax and fidelity penalty	No CC and open-loop CC
ESRT [26]	Buffer occupancy	Implicit	Simulation (ns2) and analytical/(–)	Normalized reliability and power consumption	–
Fusion [29]	Buffer occupancy	Implicit	Real experiments (TinyOS)/(55 nodes)	Throughput, fairness, latency, drop rate and efficiency	No CC, occupancy, channel sampling, rate limiting and occupancy+Delay
IFRC [39]	Weighted moving average of queue length	Explicit	Real experiments (TinyOS)/(40 nodes)	Throughput and instantaneous queue size	–
PCCP [30]	Packet service time/packet inter-arrival time	Implicit	Simulation/(7 nodes)	Normalized throughput, queue length and fairness	CCF [65]
DPCC [31]	Buffer occupancy	Implicit	Simulation (ns2 and MATLAB)/(10 nodes)	Throughput, queue level and delay	CODA [25]
HCCP [41]	Buffer occupancy and flow rate	Explicit	Simulation (ns2)/(5000 nodes)	Total source rate and control overhead packets	AFA [66] and buffer-based congestion avoidance scheme [67]
Multipath CC [68]	Average packet service rate/packet scheduling rate	Implicit	Simulation/(200 nodes)	Queue length and throughput	–

(continued)

Table 2.2 (continued)

Algorithm	Congestion detection	Congestion notification	Implementation/(number of nodes)	Evaluation metrics	Compared with
FACC [24]	Buffer occupancy and channel load	Explicit	Simulation (ns2)/(51 nodes)	Dropped packets, total source rate, throughput and energy expenditure	No CC and CODA [25]
CL-APCC [38]	Buffer occupancy and data flow	Implicit	Simulation (VC++)/(100 nodes)	Packet reception rate, queue length, energy consumption	No CC
UHCC [69]	Buffer occupancy and traffic rate	Implicit	Simulation/(11 nodes)	Normalized throughput, fairness and packet loss ratio	PCCP [30] and CCF [65]
ACT [43]	–	Implicit	Simulation (TOSSIM)/(100 nodes)	Efficiency, fairness, quality of data and energy	CODA [25] and CRRT [70]
Distributed CC [71]	Difference between input and output traffic rates	Implicit	Simulation/(100 nodes)	Goodput, fairness and transmission rate	–
DPCC [72]	Buffer occupancy and traffic rate	Explicit	Simulation/(10 nodes)	Normalized throughput and fairness	PCCP [30]
DRR [28]	Buffer occupancy	Explicit	Simulation (ns2)/(6, 26 nodes)	Packet delivery ratio, packet loss ratio, fairness and energy consumption	–
FBACC [32]	Buffer occupancy and traffic rate	Explicit	Simulation (MATLAB)/(–)	Congestion detection, packet loss, end-to-end delay and energy	ESRT [26], FLCE [73], and CCSFL [74]

In [25], Wan et al. proposed a congestion control algorithm called congestion detection and avoidance (CODA). The proposed scheme consists of three mechanisms: receiver-based congestion detection, open-loop hop-by-hop backpressure and closed-loop multi-source regulation. CODA detects congestion by combining present and past channel loading conditions and buffer occupancy at each receiver. When a node detects congestion, it broadcasts backpressure messages which are propagated upstream towards sources. Every node receives the backpressure message, and it decides whether to broadcast the message again or not, based on network conditions. When a source node receives a backpressure message, it regulates its rate based on the maximum theoretical throughput of the channel. When the source event rate is less than a fraction of channel throughput, the source regulates itself. Otherwise, if the value of source rate is higher than the fraction, the closed-loop control is triggered where the sink node regulates the source rate.

CODA has been tested through real experiments using a small sensor network testbed with TinyOS and through simulations using a packet-level simulation. Real and simulation results show that CODA reduces the average energy tax with minimal fidelity penalty as compared to open-loop congestion control strategy and without congestion control.

In [26], Sankarasubramaniam et al. proposed a new reliable transport scheme for WSN called event-to-sink reliable transport protocol (ESRT). The proposed algorithm includes a congestion control scheme for achieving reliability and saving energy. ESRT defines five characteristic operating regions in the network: No congestion, low reliability (NC, LR), no congestion, high reliability (NC, HR), congestion, high reliability (C, HR), congestion, low reliability (C, LR), and optimal operation region (OOR). The aim of ESRT is to identify the current region and move the network to OOR region. ESRT detects congestion by monitoring sensor nodes' buffer occupancy. Each node, which has buffer overflow, informs the sink node by setting the congestion notification bit in the header of succeeding packets. ESRT operation is based on the achieved reliability and congestion condition in the network. If the reliability is lower than a specific value, the sink adjusts the reporting rate of sensor nodes to achieve the required reliability level. Otherwise, if the reliability is higher than the threshold value, the sink reduces the reporting rate to save energy as much as possible while getting the target reliability.

ESRT has been tested through analytical modelling and simulation using ns2 simulator. Analytical and simulation results show that ESRT satisfies the required reliability and converges to state OOR regardless of the initial network state.

In [29], Hull et al. proposed a congestion control mechanism called Fusion which combines three techniques: hop-by-hop flow control, rate limiting source traffic and a prioritized MAC protocol. Fusion uses the implicit congestion notification scheme by setting a congestion bit in the header of every outgoing packet. The first technique, hop-by-hop flow control, has two components: congestion detection and congestion mitigation. The proposed algorithm detects congestion by monitoring a node's queue size. If the free space in the queue is less than a specific value, the congestion bit of outgoing packet is set. Congestion mitigation is a mechanism which throttles transmission of upstream nodes to prevent the queue of their parent nodes from

overflowing. When a node receives a packet in which the congestion bit is set, it stops sending packets to its next-hop node. In the second technique, rate limiting is used. Here, each node listens to its parent traffic to estimate the total number of sources, N, which are forwarding through its parent. Then, a token bucket scheme is used to regulate each node's sending rate. A node accumulates one token every time it hears its parent forward N packets, up to a maximum number of tokens. The node is allowed to send only when its token count is above zero where each transmission costs one token. The third technique, a prioritized MAC layer, gives congested nodes priority over uncongested nodes for access to the wireless channel.

Fusion has been tested and evaluated under a 55-node network testbed with TinyOS using event-based and periodic data traffic. The proposed algorithm is compared with no congestion control, buffer occupancy-based congestion control, channel sampling-based congestion control, and combined buffer occupancy- and delay-based congestion control. The experimental results show that Fusion achieves high throughput and fairness at high offered load as compared to other algorithms.

In [39], Rangwala et al. proposed an interference-aware fair rate control algorithm (IFRC) to allocate fair and efficient transmission rate to each node. IFRC comprises three components: congestion level measurement, congestion information sharing and rate adaptation using the AIMD scheme. IRFC measures congestion level using an exponentially weighted moving average of the queue length. If the average queue length exceeds a certain threshold value, congestion occurs in the node. When a node detects congestion, it shares its congestion state with other potential interferers by sending its queue length explicitly. After congestion information is shared, the AIMD rate adaptation algorithm is executed where the node halves its rate.

IFRC performance has been evaluated through a 40-sensor node network testbed with TinyOS. The experimental results show that IFRC reduces packet loss rate by 30% and prevents packet drop due to buffer overflow.

In [30], Wang et al. proposed an upstream congestion control scheme called priority-based congestion control protocol (PCCP) that utilizes a cross-layer optimization and imposes a hop-by-hop approach to control congestion. The proposed algorithm comprises three components: intelligent congestion detection, implicit congestion notification and priority-based rate adjustment. PCCP detects congestion periodically based on packet inter-arrival time and packet service time at the MAC layer. After congestion is detected, the congestion information is piggybacked in the header of data packet and sent to other nodes. Each sensor node uses a priority-based rate adjustment where each node is assigned a priority index. The rate adjustment is based on congestion degree and node priority index. PCCP is designed to support single-path routing and multipath routing scenarios.

PCCP has been evaluated through simulation within a 7-node network under single-path and multipath routing scenarios. Also, PCCP is compared with congestion control and fairness algorithm (CCF) [65]. Simulation results show that the proposed algorithm achieves high link utilization and therefore PCCP reduces packet loss, improves energy consumption and reduces packet delay as compared to CCF.

In [31], Zawodniok and Jagannathan developed a decentralized predictive congestion control algorithm (DPCC) for WSNs. The proposed algorithm comprises

two schemes (adaptive flow and adaptive CSMA back-off interval selection) that work in concert with a distributed power control (DPC). DPCC detects congestion using buffer occupancy and channel quality which is predicted by channel estimator algorithm. DPCC uses weights associated with flows to ensure fairness during resources allocation when congestion occurs. The DPCC operation is summarized in the following steps:

1. When congestion is detected, the rate selection algorithm is executed at the receiver to calculate the appropriate rate based on the predicated channel state.
2. The available bandwidth is allocated for the flows based on their weights to ensure fairness.
3. DPC and rate information are exchanged between nodes on every link.
4. At the sender, a CSMA back-off interval is selected based on the assigned outgoing rate.
5. The dynamic weight adaptation algorithm is used for further throughput and fairness enhancement.

DPCC is assessed and evaluated by MATLAB and ns2 simulator under tree topology network and compared with CODA [25]. Simulation results show that DPCC increases throughput, network efficiency and energy saving, and DPCC guarantees the targeted QoS as compared to CODA.

In [41], Sheu and Hu developed a hybrid congestion control protocol that takes into account the packet delivery rate and buffer size as congestion indication. Each node uses its current remaining buffer size and its flow rate to determine its congestion degree which reflects the current congestion level. The congestion information is exchanged among neighbours periodically every a specific period time. When a node receives the congestion degree from its neighbouring nodes, it calculates its traffic rate and updates its congestion degree. If the updated congestion degree is greater than or equal to 0, the node does nothing. Otherwise, it suppresses the data rate of its children nodes.

The proposed algorithm has been tested by ns2 simulations with 5000 nodes, which are randomly placed in an area of 1000 m $\times$ 1000 m, and compared with aggregate fairness algorithm (AFA) [66] and lightweight buffer management based congestion avoidance scheme [67]. Simulation results show that the proposed protocol has better performance in terms of throughput and packet drop rate than others.

In [68], Monowar et al. proposed a multipath congestion control mechanism for heterogeneous data originating from a single node. The proposed algorithm assumes that each node hosts multiple applications where each application has an individual priority. Also, each node has multiple parents at the same time and each application forwards its data packets to a single parent. The proposed algorithm uses the packet service ratio, which is the ratio of average packet service rate and packet scheduling rate, to detect the congestion level. Each node notifies other nodes by piggybacking the congestion information (packet service rate, number of child nodes and packet scheduling rate) in its packet header. A hop-by-hop rate adjustment is used to update the output rate of a node by adjusting the scheduling rate.

The proposed mechanism has been evaluated through simulation with 200 nodes which are randomly deployed in an area of 100 m × 100 m and each node hosts three applications for sensing temperature, pressure and seismic. Simulation results show that the proposed algorithm achieves the desired throughput according to the application priority and reduced packet drop rate.

In [24], Yin et al. proposed an algorithm called fairness-aware congestion control (FACC) which controls congestion and satisfies a fair bandwidth allocation for different flows. The authors categorize all intermediate nodes into near-source nodes and near-sink nodes. The near-source nodes maintain a per-flow state and allocate a fair bandwidth share. On the other hand, the near-sink nodes do not maintain a per-flow state and use a lightweight probabilistic dropping algorithm. When a near-sink node drops a packet, the node sends a warning message (WM) back to the near-source nodes. Once the near-source nodes receive the message, they calculate and allocate the fair rate share for each passing flow. After that, the near-source nodes send a control message (CM) to notify the source nodes of the updated sending rate. The near-source nodes implement fairness-aware transmission rate control based on available channel bandwidth, the arrival rate of each flow and the number of active flows for the node. On the other hand, the near-sink nodes implement a simple transmission control mechanism based on queue occupancy and hit frequency.

FACC has been evaluated by using ns2 simulation and compared with no congestion control and CODA [25]. Simulation results show that the proposed algorithm has better performance than other schemes in terms of packet loss, energy efficiency, channel utilization and fairness.

In [38], Wan et al. proposed a cross-layer active predictive congestion control scheme (CL-APCC) for improving network performance. The proposed algorithm is based on IEEE 802.11 which is revised according to waiting time, the number of neighbouring nodes and the original priority of data packets. The revised IEEE 802.11 dynamically adjusts the sending priority of a node. The CL-APCC operation is based on the node's buffer occupancy, data flow trends of the local network, network condition and node rate within the current period. CL-APCC predicts the input and output rates of node within the next period based on a queuing theory concept to avoid congestion.

CL-APCC has been evaluated and tested through simulation with VC++ under randomly deployed 100-node network. The simulation results show that CL-APCC improves received packet ratio of sink nodes, network lifetime and fairness as compared to no congestion control.

In [69], Wang and Liu proposed a protocol called upstream hop-by-hop congestion control (UHCC) based on cross-layer design. The proposed algorithm comprises two components: congestion detection and rate adjustment. To detect congestion, each node determines its congestion index (CI) based on unoccupied buffer size and traffic rate at the MAC layer. Based on CI value, the traffic transmission rate and local source traffic priority are updated. The congestion information is piggybacked in the header of a packet.

UHCC has been tested under a simple tree topology network within 11 nodes and compared with PCCP [30] and CCF [65]. The simulation results show that the

proposed algorithm achieves higher throughput, better priority-based fairness and reduced packet loss than other algorithms.

In [43], Lee and Jung proposed a new congestion control scheme called adaptive compression-based congestion control technique (ACT) for packet reduction when congestion occurs. The compression methods used in ACT are discrete wavelet transform (DWT), adaptive differential pulse code modulation (ADPCM) and run-length coding (RLC). In the source node, ACT first transforms the data from time domain to frequency domain using ADPCM to reduce data range. Next, RLC is used to reduce the number of packets. Next, DWT is used for priority-based congestion control as DWT classifies data into four different frequency groups. RLC generates a smaller number of packets for low priority data. In the intermediate node, ACT reduces the amount of packets by increasing the quantization step size of ADPCM when congestion occurs. Also, queue is operated adaptively according to congestion state and queue state.

ACT has been evaluated and tested using TinyOS and TOSSIM simulator and compared with CODA [25] and congestion-aware rate-controlled reliable transport algorithm (CRRT) [70]. The simulation results show that ACT increases network efficiency, guarantees fairness to nodes and increases throughput of sink nodes as compared to other algorithms.

In [71], Brahma et al. developed a distributed congestion control algorithm for tree-based communication in WSNs. The proposed algorithm assigns a fair rate to each node where a node monitors its aggregate output and input traffic rates. Based on the difference, the node decides whether to increase or decrease the transmission rates of itself and its children nodes. The proposed algorithm provides fairness among flows in the network using two separated modules to control utility of the network and fairness. The utilization controlling module computes the total increase or decrease in traffic rate. The fairness module decides on how exactly to divide the total change in traffic rate required among flows.

The proposed algorithm is implemented by using an event-driven packet-level simulator and tested under 10 nodes $\times$ 10 nodes grid. The simulation results show that the proposed algorithm achieves high goodput and attains the desired fairness.

In [72], Heikalabad et al. proposed an algorithm called dynamic prediction congestion control (DPCC). The proposed algorithm comprises three components: backward and forward node selection (BFS), predicative congestion detection (PCD) and dynamic priority-based rate adjustment (DPRA). A node selects its forward node based on the received rate adjustment values from its forwarded nodes. The node selects one as a forward node which the received rate value from it is maximum. Then, the node sends notification to the selected forwarded node. DPCC detects congestion by combining the node's unoccupied buffer size and traffic rate at the MAC layer to form Congestion Index (CI). DPCC adjusts the traffic rates of the backward nodes according to CI and total traffic priority.

DPCC has been evaluated through simulation with a network of 10 nodes under IEEE 802.11 MAC protocol. Simulation results show that DPCC improves throughput and fairness as compared to PCCP [30].

In [28], Deshpande et al. proposed an algorithm called differed reporting rate (DRR) that controls congestion in WSNs. They develop a mathematical model to control the flow of data packets through the network. The proposed algorithm has three mechanisms which are congestion detection, congestion notification and reporting rate adjustments.

DRR works as follows: each node periodically checks its buffer occupancy. If the buffer occupancy is above a threshold value which is 80, then it sets a congestion notification bit and sends a choke packet, which contains the current buffer length, to a previous node that forwards its packets through it. The node that receives this message updates its flow rate by using the mathematical equation as updated flow rate $= 51.5 \ln(\text{current buffer length}) - 85.56$. However, when a node records its buffer occupancy below 60, this node resets the congestion notification bit and sends the choke message to its previous node that may increase its flow rate.

DRR has been tested by using the ns2 simulator with a chain and random network topologies in an area of 1000 m $\times$ 1000 m. For the chain topology, there are six sensor nodes with 2 J each and three seconds simulation time whereas 26 sensor nodes with 2 J and 10 s simulation time for the random topology. Simulation results illustrate that DRR has a high packet delivery ratio, low packet loss ratio and low energy consumption for both topologies.

In [32], Jaiswal and Yadav proposed a new algorithm called fuzzy-based adaptive congestion control (FBACC) to detect congestion and regulate it in WSNs. They develop a new fuzzy logic controller for estimating congestion and adapting the traffic rate. The proposed algorithm uses buffer occupancy, participants and traffic rate as inputs for the fuzzy logic controller and transmission rate as output. When a node detects the congestion, the congested node sends a notification message to its neighbouring nodes to regulate the transmission rate.

FBACC has been tested and evaluated using MATLAB. The proposed algorithm is compared with ESRT [26], fuzzy logic-based congestion estimation algorithm (FLCE) [73], and congestion control scheme based on fuzzy logic (CCSFL) [74] in terms of congestion detection, packet loss, end-to-end delay and energy. Simulation results show that FBACC has a better performance than these algorithms. However, as the sensor node has very limited computation capabilities, it is very difficult to implement and execute the fuzzy logic controller on the sensor node.

2.6.2 Resource Control Algorithms

In this category of algorithms, resource control is applied to alleviate congestion by distributing network traffic through different paths or forwarding data packets to their final destination through less congested paths. Table 2.3 summarizes these mechanisms.

In [42], Kang et al. proposed a resource control-based algorithm called topology-aware resource adaptation strategy (TARA) to alleviate congestion. The proposed scheme detects congestion by combining buffer occupancy and channel load. TARA

Table 2.3 Resource control algorithms in WSNs

Algorithm	Congestion detection	Congestion notification	Implementation/ (number of nodes)	Evaluation metrics	Compared with
TARA [42]	Buffer occupancy and channel load	Explicit	Simulation (ns2)/(81 nodes)	Fidelity index and energy consumption	No CC, traffic control and resource control
TADR [75]	Buffer occupancy	–	Simulation (TOSSIM)/(999 nodes)	Receiving packets rate, throughput ratio and energy efficiency	MintRoute algorithm of TinyOS [76]
QoS adaptive cross-layer CC [77]	Packet inter-arrival time/packet service time	Implicit	Simulation/(50 nodes)	Average queue length and energy	No CC and CCF [65]
HTAP [33]	Buffer occupancy	Implicit	Simulation (Prowler)/(100 nodes)	Received packets ratio, throughput, hop-by-hop delay and energy consumption	No CC, TARA [42] and SenTCP [78]
DAIPaS [79]	Buffer occupancy and channel load	Implicit	Simulation (Prowler)/(100 nodes)	Received packets ratio, throughput, hop-by-hop delay and end-to-end delay	No CC, TARA [42] and HTAP [33]
CATree [80]	–	–	Simulation (OPNET)/(60 nodes)	End-to-end delay, sink bit error rate, sink packet loss ratio and sink bit errors per packet	Star, tree and mesh topologies

activates appropriate sensor nodes whose radio is off (sleeping nodes) to construct a new topology that has enough capacity to handle the increased traffic. A channel capacity model has developed to estimate the end-to-end throughput of different topologies and the model is based on a graph-colouring problem. When a node detects that its congestion level is higher than a threshold value, it should quickly locate two important nodes: distributor and merger. Then, an alternative path can be established that starts at the distributor and ends at the merger. The distributor shares the incoming traffic between the original path and the alternative path, whereas the merger combines the two flows.

TARA has been evaluated through simulation using ns2 simulator on an 81 node network and compared with no congestion control, traffic control and resource control. Simulation results show that TARA performs very close to an ideal offline resource control in terms of energy saving and fidelity satisfaction as compared to other schemes.

In [75], He et al. proposed a traffic-aware dynamic routing algorithm (TADR) to forward packets around the congestion areas and distribute heavy traffic along multiple paths. The basis of TADR is to construct two independent potential fields using depth and queue length. These two fields are combined into a hybrid potential field to dynamically make routing decisions. The potential queue length field provides a traffic-aware solution and the depth field provides the basic routing backbone to route the packets to the sink. When a queue length is higher than a certain threshold (i.e. congestion occurs), the packets are routed along other suboptimal paths.

TADR has been evaluated through simulation by using TinyOS and TOSSIM simulator. Simulation results show that TDRA achieves its objectives and improves network throughput as compared to a benchmark routing protocol with minimum overhead packets.

In [77], Rahman et al. proposed a new QoS adaptive cross-layer congestion control approach to support QoS guarantee for different application data. The proposed scheme detects congestion based on the ratio between packet inter-arrival time and packet service time at the MAC layer; the ratio is called congestion scale. An implicit congestion notification method is used to notify other nodes about congestion status. Two congestion control mechanisms are proposed to mitigate congestion: short-term and long-term congestion control. The short-term congestion control is used to remove short-term congestion; when a node detects congestion, its child node distributes the real-time traffic into its alternative parent (path). If the short-term scheme cannot avoid congestion, the long-term congestion control is carried out where intermediate nodes periodically send congestion information as a back-pressure message. When a source node receives the message, it applies the short-term congestion control mechanism.

The proposed algorithm has been evaluated through simulation on a 50-node network and compared with no congestion control and CCF [65]. Simulation results show that the proposed scheme improves network throughput, average queue occupancy and energy consumption as compared to others.

In [33], Sergiou et al. developed a new algorithm called hierarchical tree alternative path (HTAP) for congestion control in WSNs. The proposed algorithm uses the resource control method and solves the congestion problem by creating dynamic alternative paths from the source node to the sink node. The main features of HTAP are its topology control scheme where each node builds its local minimum spanning tree and each node is able to recognize deadlocks.

HTAP has four steps which are topology control, hierarchical tree creation, alternative path creation and handling of powerless nodes. In the first step, each node builds its neighbouring table by using the local minimum spanning tree algorithm (LMST). Each node broadcasts periodically a “Hello” message which contains the ID and location of the node with its maximum transmission power level. Each node, which receives the “Hello” message, applies Prim’s algorithm in order to build a power efficient minimum spanning tree where the node selects six nearest neighbours. Then, the node determines and adjusts its transmission power level to a level that can reach to its farthest neighbour.

The next step runs when a source node starts to send packets. In this step, each source node assigns itself as a level 0 and sends a level_discovery message to all its neighbours which are selected during the topology control step. The nodes that receive this message consider themselves as level 1 and again they send the level_discovery message to their neighbours. This process continues until this message reaches the sink node. During this step, if a node becomes unable to forward packets a level up, it broadcasts a negative acknowledge (NACK) message. Therefore, the nodes know that they cannot forward packets through this node. Also, a connection between nodes is established by using a two-way handshake where the nodes can exchange the congestion state.

The alternative path creation step is executed when a node becomes nearly congested. In this step, each node monitors its buffer, when the buffer starts to fill where a number of receiving packets are more than a number of sending packets. In this case, this node sends a backpressure message to the nodes that send their packets through it to notify them that it is congested. Therefore, these nodes update their table and avoid sending packets through the congested node. Also, they should select another node to forward packets. Finally, the last step runs when the power of a node exhausts where this node broadcasts a message to notify other nodes to remove it from their neighbouring table. So, the alternative path creation step is executed again to select an alternative path.

HTAP has been evaluated by using the Prowler simulator and compared with three other algorithms. HTAP is tested under different scenarios with 100 nodes which are uniformly deployed in an area of 500 m $\times$ 500 m. Simulation results show that the proposed algorithm is a more efficient and simple solution for the congestion problem than TARA [42] and the hop-by-hop congestion control protocol (SenTCP) [78]. However, HTAP consumes energy by using overhead packets (hello, level_discovery and backpressure messages).

In [79], Sergiou et al. proposed an algorithm called dynamic alternative path selection (DAlPaS). The proposed algorithm uses the resource control method by creating a dynamic alternative path to mitigate congestion in WSNs. The main feature of DAlPaS is a flag algorithm that uses several factors such as buffer occupancy, remaining power and hop count to select the most appropriate path. The proposed algorithm has good performance in terms of hop-to-hop delay and throughput.

DAlPaS has one phase and three schemes which are the setup phase, the topology control scheme, the soft stage and the hard stage scheme, respectively. The setup phase is executed only once during the network initialisation. In this phase, the sink node broadcasts a "hello" message within its level (level 0). Every node that receives this message responds to the sink node by sending an ACK message. When the sink node receives this ACK message, it resends a "connect" message to the nodes that sent the ACK message. Then, these nodes make themselves as level 1 and update their neighbouring table. After that, the level 1 nodes broadcast again the hello message and this process continues as above until all nodes discover each other.

In the topology control scheme, each node uses its neighbouring table that has been built during the setup phase to choose only nodes that are located in a lower level than its own level in order to forward its packet through them. The soft stage

scheme is executed when a node receives packets from more than one flow (node). This node sends a backpressure message to one of these nodes to notify it to stop transmitting packets and find an alternative path. If the node which receives this message cannot find the alternative path, the hard stage scheme is executed to force the node to change its path. This scheme has two steps which are a flag decision algorithm and alternative path creation. In the first step, each node updates a flag field in its neighbouring table either to 0 when a neighbour node becomes unavailable or to 1 when the neighbour node is available. The calculation of the flag is based on three factors: buffer occupancy, remaining power and level node unavailability. In the second step, each node sorts its available nodes (their flag is 1) in the table according to their number of hops, remaining power and buffer occupancy. The node selects a neighbour node which is located in the table in order to forward its packets.

DAlPaS has been evaluated and compared with no congestion control, TARA [42] and HTAP [33]. DAlPaS is tested by using the Prowler simulator with 100 nodes which are deployed uniformly in an area of 50 m $\times$ 50 m. Simulation results show that DAlPaS improves the average throughput of the network and the average end-to-end delay more than other algorithms. However, DAlPaS uses many overhead packets (hello, ACK, connect and backpressure messages) during the setup phase that increase the consumed energy. Moreover, the limitation of the proposed algorithm is that each node should be aware of its position and the position of the sink node.

In [80], Dasgupta et al. proposed a congestion avoidance scheme called CATopology or CATree. The proposed algorithm uses a Karnaugh map to create a tree topology which is free from congestion at the link level. At first, the sink node stores a table that represents the relationship among nodes in the form of a Karnaugh map. Then, a depth-first traversal strategy is used to create the collision avoidance tree. In this tree, each node has a level which represents a communication round which the node can transmit its data packets. Also, two or more nodes from the same parent cannot be with the same level to ensure the collision avoidance state. The data transmission is triggered by the sink node that sends data request packets to the nodes which start to transmit a large number of data packets where each node takes its own communication round.

CATree has been evaluated by using the OPNET simulator within 60 nodes which are uniformly distributed in an area of 100 $\times$ 100 scale. The proposed algorithm is tested and compared with three other topologies which are star, mesh and tree. The simulation results show that CATree improves sink packet loss ratio, network end-to-end delay, energy consumption and network lifetime. However, the proposed algorithm is valid only with the query driven application. Also, CATree does not have a strategy that deals with the occurrence of congestion.

2.6.3 Hybrid Schemes

This subsection reviews congestion control mechanisms which combine the traffic control method and resource control to mitigate the network congestion. Table 2.4 summarizes these algorithms.

Table 2.4 Hybrid algorithms in WSNs

Algorithm	Congestion detection	Congestion notification	Implementation/ (number of nodes)	Evaluation metrics	Compared with
TALONet [36]	Buffer occupancy	Implicit	Simulation (ns2)/ (50–200 nodes)	Dropped packets and power consumption	No CC, TARA [42] and backpressure
Multipath routing CC [81]	Buffer occupancy	Explicit	Simulation (ns2)/(1000 nodes)	Throughput and packet delivery ratio	No CC, buffer-based congestion avoidance scheme [67] and PCCP [30]
CADA [40]	Buffer occupancy and channel load	Implicit	Simulation (ns2)/(500–5000 nodes)	End-to-end delivery ratio, bit energy consumption, per-hop delay and throughput	No CC, TARA [42]
HRTC [82]	Buffer occupancy	Explicit	Simulation (Prowler) /(30 nodes)	Throughput	No CC, traffic control and resource control

In [36], Huang et al. proposed an energy-efficient grid-based traffic congestion avoidance scheme called TALONet. The proposed algorithm uses three approaches to avoid congestion: two different transmission power levels are used to mitigate link-level congestion, an efficient buffer management method is used to avoid node-level congestion and a multipath detouring technique is used to increase the channel capacity for congested flows. TALONet comprises three phases: network formation, data dissemination and framework updating. The first phase is used to create a virtual grid topology where the sink node broadcasts a control message which contains its location and its distance from other nodes. A node located in intersections of grid is called a talon node which is responsible for collecting and relaying data packets during the second phase. After the grid topology network is formed, a normal node transmits its data to its neighbouring talon node at a minimum power level. Then, the talon node forwards the packets with maximum power level to another close to sink talon node until the data reaches to the sink. A node with maximum free buffer space is selected as the forwarding node to avoid congestion. When the buffer occupancy is higher than a threshold value, the transmission rate is reduced. The last phase is used to update the network topology either conditionally or periodically to avoid exhausting the talon nodes as they use the maximum transmission power level.

TALONet has been evaluated through simulation using ns2 simulation and compared with no congestion control, TARA [42] and backpressure method. Simulation results show that TALONet improves packet delivery rate, increases network lifetime and saves energy as compared to others.

In [81], Razzaque and Hong proposed a congestion control mechanism for multipath data forwarding in WSNs. The proposed algorithm supposes that each source

node has to establish multiple paths to the sink using a multipath routing algorithm. A source node sends data packets through two different paths at a specific loading rate. The buffer occupancy method is used to detect congestion by using an exponential weighted moving average. If the average is higher than a certain threshold, an intermediate node sends a congestion notification message to the source node. When the source node receives the message, it stops sending packets over the two paths. Then, it reduces the loading rate and waits for a specific time. If the source node does not receive another notification message during the wait time, it sends packets with the updated loading rate.

The proposed algorithm has been evaluated through simulation using ns2 and compared with no congestion control, lightweight buffer management based congestion avoidance scheme [67] and PCCP [30]. Simulation results show that the proposed scheme increases packet generation rates and throughput by a factor of 1.5 as well as improving packet delivery ratio as compared to other schemes.

In [40], Fang et al. proposed a congestion control scheme called congestion avoidance, detection and alleviation (CADA). The proposed algorithm consists of three main mechanisms for avoiding, detecting and alleviating congestion. First, when an event occurs, subnet nodes in the event area are chosen to become data sources. The other nodes are suppressed from reporting data to the sink. Thus, the traffic load from the event area is reduced. Second, every node periodically measures the congestion level in hotspot areas by checking the buffer occupancy and channel utilization. Lastly, if congestion cannot be avoided in the first step and congestion is detected, two methods are used for alleviating congestion: resource control and traffic control. The resource control method tries to redirect some traffic away from the traffic hotspot by establishing detour routes. If alternative paths are not available, the traffic control strategy is executed by reducing the traffic rate at source nodes by using an AIMD-like policy.

CADA has been evaluated in ns2 and compared with no congestion control and TARA [42] using a variable number of nodes (500–5000). Results show that the proposed algorithm has better performance in terms of throughput, energy consumption and average per-hop delay than others.

In [82], Sergiou and Vassiliou proposed a new algorithm called hybrid algorithm for efficient congestion control (HRTC) that controls congestion in WSNs. They develop a hybrid algorithm by combining two methods which are traffic control method and resource control method where the proposed algorithm utilizes the positive aspects of both methods. HRTC improves the efficiency of the network in terms of packet delivery ratio and network lifetime.

HRTC works as follows: when a node faces congestion, it sends a backpressure message to the source node to notify it that congestion has occurred and its data rate should be decreased to a minimum. When intermediate nodes, which are located between the source node and the congested node (receiver), receive this message, they check if the resource control method can be applied to solve the congestion problem. Then, this method is executed and the backpressure message is eliminated. Otherwise, they forward the message to the source node. When the source node receives this message, it applies the traffic congestion method and decreases its data

rate to minimum. Next, whenever the source node sends a data packet, it sets the throttle bit in the header of the sending packet to indicate that it is throttled now. Any node which receives this data packet checks if the congestion can be solved by applying the resource control method. Then, it runs this method and sends a subsequent backpressure message to the source node that can now send packets at its maximum transmission rate.

The Prowler simulator is used to evaluate the performance of HRTC where the proposed algorithm is compared with two schemes which are a pure resource control and a pure traffic control. HRTC is tested under two scenarios with 30 nodes which are deployed in an area of 100 m $\times$ 100 m. Simulation results show that HRTC improves throughput of the network and extends the network lifetime more than the pure traffic and resource control schemes.

2.7 Congestion Control Algorithms for 6LoWPAN Networks

Recently, a number of articles suggest new congestion control mechanisms for 6LoWPAN networks. A review of these mechanisms as well as how each algorithm works are given next. In this section, the algorithms are classified according to congestion control method into traffic control algorithms (Sect. 2.7.1) and resource control algorithms (Sect. 2.7.2). Table 2.5 summarizes these mechanisms, and Table 2.6 shows the advantages and disadvantages of the algorithms.

2.7.1 Traffic Control Algorithms

In [27], Michopoulos et al. proposed a new congestion control algorithm called duty cycle-aware congestion control (DCCC6) for control congestion in 6LoWPAN networks. The proposed algorithm detects the presence of duty cycle and adjusts its operation accordingly. The proposed protocol uses the buffer occupancy as a congestion detection method as well as traffic control strategy to reduce the congestion in the network.

DCCC6 works as follows: every node monitors its buffer occupancy. If the buffer occupancy exceeds a threshold value, the congested node sends a notification back to the sources of congestion. The congested node adjusts the threshold value dynamically to avoid high rate of notification messages. If the node uses RDC scheme, the notification is sent inside unicast frames. Otherwise, if the radio is always on, the node sends the notification with broadcast packets. When a node receives the notification, it adapts its data rate by using a modified AIMD scheme.

DCCC6 is implemented using the Cooja simulator as well as a testbed network and compared with HCCP [41], AFA [66], IFRC [39] and CSMA. In the simulation,

Table 2.5 Traffic and resource control algorithms in 6LoWPAN networks

Algorithm	Congestion detection	Congestion notification	Congestion control	Implementation/ (number of nodes)	Evaluation metrics	Compared with
DCCC6 [27]	Buffer occupancy	Implicit and explicit	Traffic control	Simulation (Cooja) and real experiments (Contiki OS)/(15, 25 nodes)	Goodput, end-to-end delay, energy consumption and Jain's fairness index	HCCP [41], AFA [66], IFRC [39] and CSMA
Griping, deaf and fuse [35]	Buffer occupancy	Implicit and explicit	Traffic control	Simulation (ns-3)/(14 nodes)	Reception rate, multi-hop delay, loss probability, rejection rate and transmission overhead	Backpressure [83] and UDP
Bird flocking CC [84]	Buffer occupancy	–	Resource control	Simulation (Cooja)/(50 nodes)	Duplicate messages and transmission time	CoAP [85]
QU-RPL [34, 86]	Buffer overflow	Explicit	Resource control	Real experiments (TinyOS)/(30 nodes)	Packet delivery, packet loss ratio, hop distance and routing overhead packets	RPL [87]
GTCC [44, 45]	Difference between packet generation rate and packet service rate	Explicit	Resource control	Simulation (Cooja)/(22, 26 nodes)	Packet loss rate, throughput and hop count	RPL with OF0 [88] and RPL with ETX-OF [89]
CA-RPL [90]	–	Implicit	Resource control	Simulation (Cooja)/(21 nodes)	Throughput, packet loss rate and average end-to-end delay	Original RPL [87]
Lodhi's M-RPL [91]	Packet delivery ratio	Implicit	Resource control	Simulation (Cooja)/(113 nodes)	Throughput, end-to-end latency and energy consumption	RPL [87]

(continued)

Table 2.5 (continued)

Algorithm	Congestion detection	Congestion notification	Congestion control	Implementation/ (number of nodes)	Evaluation metrics	Compared with
MLEq [92]	–	–	Resource control	Simulation (ns2)/(100 nodes)	Throughput, Jain's fairness index and control packet overhead	RPL [87]
LB-RPL [93, 94]	–	–	Resource control	Simulation (ns2)/(1000 nodes)	Packet delivery ratio and end-to-end delay	RPL [87]
Tang's M-RPL [95]	–	–	Resource control	Simulation (Cooja)/(20 nodes)	Packet reception rate, packet loss rate and end-to-end delay	RPL [87]
CA-OF RPL [Chap. 4]	Buffer occupancy	Implicit	Resource control	Simulation (Cooja)/(19, 35 nodes)	Number of lost packets, throughput, packet delivery ratio and energy consumption	RPL with OF0 [88], RPL with ETX-OF [89] and RPL with ENERGY-OF [96]
GTCCF [Chap. 5]	Ratio of forwarding rate to receiving rate	Explicit	Traffic control	Simulation (Cooja)/(5, 21 nodes)	Packet loss, throughput, delay, weighted fairness index and energy consumption	DCCC6 [27]
OHCA [Chap. 6]	Ratio of forwarding rate to receiving rate	Explicit	Hybrid scheme	Simulation (Cooja)/(10, 25 nodes)	Packet loss, throughput, delay, weighted fairness index and energy consumption	DCCC6 [27] and QU-RPL [34, 86]

Table 2.6 Pros and cons of congestion control algorithms in 6LoWPANs

Algorithm	Advantages	Disadvantages
DCCC6 [27]	• Aware of RDC mechanism • Improves fairness, delay and energy consumption	• Does not support the hybrid application type • Does not utilize non-congested paths (nodes) to forward packets to sink
Griping, deaf and fuse [35]	• No control overhead packets • Improves packet reception rate and buffer overflowed packets	• ACK packet loss does not mean that receiver's buffer is overflowed • Does not support the hybrid application type • Does not utilize non-congested paths (nodes) to forward packets to sink
Bird flocking CC [84]	• Avoid congestion areas by using bird flocking concept • Improves transmission time and duplicate packets	• Radio is always ON • Waste extra energy by passive listening • Calculation of the proposed algorithm parameters is not accurate • Does not support RDC mechanism • Does not support the hybrid application type
QU-RPL [34, 86]	• Provides network traffic load balancing • Improves queue losses and packet delivery ratio	• Increases control overhead packets • Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not support the hybrid application type
GTCC [44, 45]	• Selects alternative less congested paths by using game theory • Improves throughput and packet loss ratio	• Increases control overhead packets • Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not support the hybrid application type
CA-RPL [90]	• Mitigates congestion by distributing heavy traffic to different paths • Improves packet loss and delay	• Does not aware when high packet overflow occurs at nodes' queue • Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not support the hybrid application type
Lodhi's M-RPL [91]	• Splits the forwarding rate among multiple paths • Improves throughput, latency and energy consumption	• Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not the support hybrid application type

(continued)

Table 2.6 (continued)

Algorithm	Advantages	Disadvantages
MLEq [92]	• Achieves load balancing and distribution based on water flow behaviour working principle • Supports and is aware of multiple gateways in the network • Improves throughput, fairness and control overhead	• Does not have a strategy to detect congestion when it occurs • Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not the support hybrid application type
LB-RPL [93, 94]	• Distributes source node's heavy workload among k parents • Improves packet delivery ratio and end-to-end delay	• Does not have a strategy to detect congestion when it occurs • Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not the support hybrid application type
Tang's M-RPL [95]	• Uses dynamic adaptive routing scheme to alleviate congestion • Improves packet reception rate, packet loss rate and end-to-end delay	• Does not have a strategy to detect congestion when it occurs • Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not the support hybrid application type
CA-OF RPL [Chap. 4]	• Selects less congested nodes (paths) by using buffer occupancy as a routing metric • Improves packet loss due to buffer drops, throughput, packet delivery ratio and energy consumption	• Does not have a policy to reduce source rate when non-congestion nodes (paths) are not available • Does not support the hybrid application type
GTCCF [Chap. 5]	• Formulates congestion problem as a non-cooperative game • Adapts nodes' sending rate using Nash equilibrium • Supports node priorities and application priorities awareness • Improves packet loss, throughput, delay, weighted fairness index and energy consumption	• Does not utilize non-congested paths (nodes) to forward packets to sink
OHCA [Chap. 6]	• Combines traffic control and resource control strategies into a hybrid solution • Uses GRA as a resource control strategy and NUM as a traffic control strategy • Supports node priorities and application priorities awareness • Improves packet loss, throughput, delay, weighted fairness index and energy consumption	–

DCCC6 has been tested with 25 emulated Tmote Sky nodes which are distributed randomly. On a real testbed, DCCC6 has been evaluated by using 15 nodes with Contiki OS. The simulation and real results show that the proposed algorithm has good performance in terms of energy consumption, average delay time and a higher degree of fairness than other algorithms. However, DCCC6 does not support hybrid application types which are common in IoT and 6LoWPAN. Also, it does not use a resource control strategy to mitigate congestion.

In [35], Castellani et al. proposed three different congestion control schemes called Griping, Deaf and Fuse for control unidirectional and bidirectional data flows in CoAP/6LoWPAN networks. The proposed algorithms are based on a distributed backpressure concept which is proposed in [83], and implemented at layer 3 of each sensor node. The proposed algorithms use a buffer occupancy strategy to detect congestion as well as traffic control method to mitigate congestion by adjusting the transmission rate to reduce the rate of injected packets into the network.

In Griping, when a node receives a new datagram, it checks its layer 3 queue length. If the queue length is greater than a threshold, Q_{thr}, the node sends back a BP (backpressure) control message to the sender of the datagram. However, the receiver cannot send more than one BP message to the same sender during K seconds. Whenever the sender receives the BP message, it halves its transmission rate. The sender can send W datagrams during T seconds (time slot). If no BP control message has been received during T seconds, the sender increments its transmission rate, W.

In Deaf, when a receiver receives a datagram, it checks its layer 3 buffer length. If the length is above a threshold, Q_{thr}, it stops sending layer 2 acknowledgement to the sender of the datagram. The sender waits T_{wait} seconds from the transmission of the datagram until it retransmits. The sender updates the T_{wait} as follows: $T_{wait} = 2^n T$ where T is a layer 3 time slot and n is the number of transmissions of the same datagram that limits to a maximum value of 4. According to the above formula, whenever the sender does not receive the acknowledgement message during T_{wait}, it doubles the value of T_{wait} after each failure of sending the same datagram.

The last scheme, Fuse, combines the action of both Griping and Deaf. If a buffer length of a receiver is less than a maximum threshold, the behaviour of the receiver is the same as in Griping. Also, when the receiver's buffer length is full, the receiver combines the actions of Griping and Deaf by sending BP control message as well as stopping transmission of acknowledgement. Whenever the sender receives the BP message, it acts as in Griping.

The proposed algorithms have been simulated using ns-3 and compared with a pure backpressure scheme and UDP. They are tested within a tree topology network which contains nine leaf nodes, four routers and one border router, and under two scenarios: unidirectional flows and bidirectional CoAP traffic. Simulation results show that Fuse is the best performing scheme for both scenarios in terms of packet reception rate, packet loss rate and transmission overhead. The transmission overhead includes the number of transmissions for successfully received single packet and BP control messages, and the rate of rejects due to buffer overflow. Conversely, the Deaf scheme is simple and does not require control message transmission but its throughput is 5–10% smaller than the Fuse scheme. However, in both Deaf and

Fuse algorithms, a sender assumes that lack of reception of an acknowledgement message means that the buffer is overflowed but there are other reasons for missing the acknowledgement message such as packet error in the wireless channel.

In Chap. 5, we formulated the congestion problem in 6LoWPAN networks as a non-cooperative game framework where the nodes (players) behave uncooperatively and demand high data rate in a selfish way. Based on this framework, we proposed a simple congestion control mechanism called game theory based congestion control framework (GTCCF). The proposed algorithm adapts the nodes' sending rate using Nash equilibrium solution concept such that congestion is mitigated. GTCCF is aware of node priorities and application priorities to support the IoT application requirements.

The proposed framework has been tested and evaluated through two different scenarios by using Contiki OS and compared with comparative algorithms. Simulation results show that GTCCF improves performance in the presence of congestion by an overall average of 30.45, 39.77, 26.37, 91.37 and 13.42% in terms of throughput, end-to-end delay, energy consumption, number of lost packets and weighted fairness index, respectively, as compared to DCCC6 algorithm.

2.7.2 Resource Control Algorithms

In [84], Hellaoui and Koudil proposed a congestion control solution for CoAP/RPL/6LoWPAN networks. The proposed algorithm is based on a bird flocking concept to pass packets through uncongested areas and avoid congested ones. Birds display a structured and organized order during their migration without collisions even when obstacles are encountered. The proposed mechanism uses the buffer occupancy strategy to detect congested nodes in the network as well as the resource control method to mitigate the congestion by selecting the least congested routes to deliver the packets to the destination (sink node).

The authors define two areas: 'zone of repulsion' (ZoR), which is an area that contains the sending node, its parents and children (one hop), and 'zone of attraction' (ZoA), which contains parents and children of next-hop nodes of the sending node (two hops). The least congested node in each ZoR and ZoA is selected as next two hops to route a packet through them. Also, the proposed algorithm uses two parameters, Q_s^{ZoR} and Q_s^{ZoA}, to estimate the buffer filling ratio of nodes in ZoR and ZoA, respectively. The calculation of Q_s^{ZoR} and Q_s^{ZoA} is done by using the wireless transmission medium where the sending node always eavesdrops (passive listening) the number of UDP messages sent and received by the nodes in the ZoR.

The proposed solution has been implemented by using the Contiki OS simulator, Cooja, and compared with Confirmable (CON) and Non-confirmable (NON) transactions of CoAP. The proposed mechanism is tested within 50 nodes which are distributed in an area of 201 m $\times$ 201 m during 300 s simulation time. The simulation results show that the proposed algorithm has a good performance in terms of duplicate messages and average transmission time more than CON trans-

actions. However, the proposed technique is executed even when the network is not congested. Therefore, packets may not pass through the best route in terms of energy consumption and end-to-end delay. As a result, the proposed algorithm might not be good in terms of energy saving and packet delay. Also, the calculation of Q_s^{ZoR} and Q_s^{ZoA} is not accurate since the sending node cannot always be aware of sending and receiving UDP packets in ZoR nodes. Moreover, the node always eavesdrops (passive listening) to the wireless channel. Thus, the radio is always on and therefore energy consumption is wastefully increased.

In [34, 86], Kim et al. proposed an effective queue utilization-based RPL algorithm called (QU-RPL). The proposed algorithm reduces the queue losses in case of congestion. QU-RPL uses the queue utilization (QU) factor in parent selection process to satisfy the traffic load balancing. When a node experiences a certain number of consecutive buffer overflows, it broadcasts a DIO message which contains the congestion information. The node changes its parent on experiencing congestion with one that has less buffer occupancy and lower hop distance to LLN border router. Otherwise, without congestion, the node chooses its best parent based on the same parent selection mechanism of the default RPL.

QU-RPL has been implemented and tested under 30 nodes and one LLN border router real testbed network with TinyOS. The proposed algorithm is compared with the default RPL in terms of packet delivery, queue loss ratio, hop distance and routing overhead packets. The experimental results show that QU-RPL alleviates the packet loss problem at queues and achieves improvement in end-to-end packet delivery performance.

In [44, 45], the authors proposed a congestion control mechanism called game theory congestion control (GTCC) for 6LoWPAN networks. The proposed algorithm is based on game theory over RPL to mitigate the effect of congestion. GTCC detours the traffic flow to an alternative path by using parent-change procedure. The proposed protocol detects congestion by using the network packet flow rate which is packet generation rate subtracted by packet service rate. When a parent node detects congestion, it sends a congestion message to its children through a DIO control packet. When the children nodes receive the DIO packet, they start the parent-change procedure. In this procedure, the node uses the potential game theory method to decide whether to change its parent or not. When the node changes its parent, it broadcasts a new DIO message to notify other nodes and update their information.

GTCC has been implemented and tested by using Contiki OS and Cooja simulator under two scenarios. Also, the proposed algorithm is compared with two others: RPL with OF0 (objective function zero) and RPL with ETX-OF (expected transmission count objective function). Simulation results show that GTCC has two times improvement in throughput and packet loss rate as compared to RPL protocols.

In [90], Tang et al. proposed a congestion avoidance multipath routing algorithm based on RPL called CA-RPL. Also, the authors propose a routing metric for RPL called DELAY_ROOT which minimizes the average delay towards the root node. CA-RPL mitigates network congestion by distributing a large amount of traffic to different paths. The proposed algorithm uses the DELAY_ROOT and three other metrics: ETX, rank and number of received packets for parent selection process.

CA-RPL has been tested over a 21-node network with Contiki OS and Cooja simulator and compared with RPL which uses the ETX metric. Simulation results show that CA-RPL reduces the number of lost packets and the time delay from original RPL by an average of 20% and 30%, respectively.

In [91], Lodhi et al. proposed a multipath extension of RPL routing protocol called M-RPL which provides a temporary multipath routing when congestion occurs. In M-RPL, intermediate (forwarding) nodes are responsible for detecting congestion by using packet delivery ratio. When the packet delivery ratio is lower than a certain threshold called the Congestion Interval (CI), the congested node sends a congestion notification to the source node through a DIO message. Once the source node receives the DIO packet, it forwards packets through multiple paths to the sink by splitting its forwarding rate into two halves. One half is forwarded to the original parent, while the other half is forwarded to another parent selected for the parent table.

M-RPL has been tested over a random topology by using Contiki OS and Cooja simulator and compared with the original RPL. Simulation results show that M-RPL supports higher data rates as compared to RPL. Also, M-RPL improves overall throughput, reduces end-to-end latency and decreases energy consumption.

In [92], Ha et al. proposed a dynamic and distributed load balancing scheme called multi-gateway load balancing scheme for equilibrium (MLEq) for 6LoWPAN network with multiple gateways. The working principle of MLEq is based on water flow behaviour such that water flows downwards and finds its own level. The proposed scheme models all the traffic flows to each gateway in the network as a three-dimensional terrain in a dynamic and distributed way. Each node maintains a parameter called virtual height level (VL) which reflects the present conditions of traffic load, link quality and hop distance. Initially, each gateway sends multicast VL information object (VIO) messages to its neighbours. Every intermediate (router) node receives the VIO message, it updates its VL value and sends multicast VIO messages to its neighbours. This process continues until all nodes successfully update their VL values. Each node selects a neighbor as its parent with the lowest VL value to deliver packets to the gateway through the optimal path in terms of load balancing and path quality.

MLEq has been evaluated through simulation under randomly deployed 100 node network by using ns2 simulator and compared to RPL. Simulation results show that MLEq has better performance in terms of throughput, fairness and control message overhead as compared to the native RPL.

In [93, 94], the authors proposed a load balanced routing protocol based on RPL called LB-RPL for 6LoWPAN network to achieve balanced heavy traffic load distribution. The proposed protocol takes into account the workload differences and distributes the data traffic among different parent nodes. LB-RPL modifies the DODAG construction procedure in the native RPL such that a node will not send a new DIO packet immediately. Instead, the node starts a timer, which is proportional to its workload, and transmits the DIO packet after the timer expires. The authors define a parameter called buffer utilization counter to quantify the workload. This parameter can be defined as the average number of packets in the buffer within a time period or the total number of new packets pushed into the buffer. In LB-RPL, a source node

selects a number of top parents from its parent table to distribute and forward its traffic load.

LB-RPL has been evaluated through simulation over a 1000-node network by using ns2 simulator. Simulation results show that the proposed protocol performs better as compared to RPL in terms of traffic load distribution, packet delivery rate and end-to-end delay.

In [95], Tang et al. proposed a multipath routing optimization strategy for RPL called M-RPL which relieves network congestion and decreases packet loss rate. The proposed mechanism uses a dynamic adaptive routing scheme which combines ETX metric and number of sent packets at a node to dynamically adjust the selection of paths. M-RPL has been evaluated through simulation over 20 node network by using Cooja simulator. Simulation results show that M-RPL performs better in the presence of congestion, reduces packet loss rate and decreases end-to-end delay.

In Chap. 4, we proposed a new RPL-based objective function called congestion-aware objective function (CA-OF) that works efficiently when congestion occurs. The proposed objective function combines two metrics: buffer occupancy and ETX and forwards packets to sink node through less congested nodes. CA-OF reflects how much the nodes are congested by using buffer occupancy metric and how much the wireless link is congested by using ETX metric.

The proposed objective function has been tested and evaluated under two scenarios with 19 node and 35 node networks by using Contiki OS and Cooja simulator. Also, CA-OF is compared with three other objective functions: RPL with OF0, RPL with ETX-OF and RPL with ENERGY-OF. Simulation results show that CA-OF improves performance in the presence of congestion by an overall average of 37.4% in terms of number of lost packets, throughput, packet delivery ratio and energy consumption as compared to others.

In Chap. 6, we proposed a novel congestion control algorithm called optimization based hybrid congestion alleviation (OHCA) which combines traffic and resource control strategies into a hybrid solution. OHCA utilizes the positive aspects of each strategy and efficiently uses the network resources. The proposed algorithm uses a multi-attribute optimization methodology called grey relational analysis for resource control by combining three routing metrics (buffer occupancy, expected transmission count and queuing delay) and forwarding packets through non-congested parents. Also, OHCA uses optimization theory and network utility maximization (NUM) framework to achieve traffic control when the non-congested parent is not available. The proposed algorithm is aware of node priorities and application priorities to support the IoT application requirements where the applications sending rate allocation is modelled as a constrained optimization problem.

The proposed algorithm has been tested and evaluated through simulation by using Contiki OS and compared with comparative algorithms. Simulation results show that OHCA improves performance in the presence of congestion by an overall average of 28.36, 28.02, 48.07, 31.97 and 90.35% in terms of throughput, weighted fairness index, end-to-end delay, energy consumption and buffer dropped packets as compared to DCCC6 and QU-RPL.

Recently, some papers have modelled and analysed TCP performance over 6LoWPAN network. In [97], Zheng et al. studied TCP on two scenarios: single-hop and multi-hop in terms of throughput, energy consumption and number of end-to-end retransmissions. The authors evaluated TCP through a testbed with a seven-node network by using Contiki OS. In [98], Ayadi et al. developed a mathematical model to predict energy consumption due to TCP in 6LoWPAN network. The authors used the OMNET++ simulator to validate the proposed model. The model estimates TCP energy consumption based on bit error rate, maximum number of retransmissions at the MAC layer, number of hops, amount of forward error correction (FEC) and TCP maximum segment size. Also, the proposed model studies the effect of the segment size, the FEC redundancy ratio and the maximum MAC retransmissions on the total energy consumption. In [99], Kim et al. presented a comprehensive experimental study on the performance of TCP over RPL in 6LoWPAN network by using TinyOS and a multi-hop testbed of 30-node network. The experimental results show that TCP sacrifices significant throughput to maintain its reliability. Also, TCP has unfairness among nodes in terms of throughput and TCP does not affect the operation of RPL in terms of control overhead and parent changes.

2.8 Discussion and Future Direction

Several mechanisms and algorithms have been proposed to solve congestion problems in WSNs. Nevertheless, the question remains of whether the WSN congestion control mechanisms are suitable and valid for 6LoWPAN networks.

1. Two methods are used to solve or mitigate congestion problem in WSNs: traffic control and resource control. Many congestion control mechanisms have been proposed based on resource control strategy such as [33, 36, 40, 42, 75, 77, 79–82, 100] where the congestion control algorithm is responsible to construct the network topology by selecting a non-congested path from source to destination. However, in 6LoWPAN networks the RPL routing protocol, which is expected to be the standard routing protocol for 6LoWPAN, is completely responsible for network topology construction by using an objective function (e.g. OF0, ETX-OF, etc.). Therefore, a conflict occurs between RPL protocol operation and the resource control strategy-based congestion control mechanisms in traditional WSNs.
2. In contrast to the traditional WSN, 6LoWPAN networks might host a variety of applications at the same time as they connect to the Internet, i.e. hybrid application types which are common in the IoT. These different applications have various packet sizes and different priorities. So, we need a congestion control algorithm that supports different applications and is aware of packets priorities as well as nodes priorities. To the best our knowledge, there is no proposed congestion control mechanism in 6LoWPAN that supports hybrid application types.

3. In [101], Michopoulos et al. have demonstrated that RDC mechanisms (e.g. ContikiMAC which is used in Contiki OS) have an impact on the performance of the congestion control algorithm. This effect is neglected when designing and implementing congestion control in traditional WSN.
4. The protocol stack of 6LoWPAN is different from the traditional WSN one. Sensor nodes in 6LoWPAN implement the Internet protocol (IP) stack as they are connected to the Internet. Also, a new layer is developed between the data link layer and the network layer, called the adaptation layer, to support IPv6 packet transmission over IEEE 802.15.4 links. Moreover, the majority of congestion control algorithms in traditional WSNs are built and evaluated on IEEE 802.11 standard such as [24–26, 28, 30, 31, 36, 38, 40–42, 68, 72, 77, 81, 100]. IEEE 802.11 is significantly different from IEEE 802.15.4 in many aspects such as data rate of IEEE 802.11 is up to 54 Mbps and it was designed for wireless local area network (WLAN) not for WSN. On the other hand, IEEE 802.15.4 can support a maximum data rate of 250 kbps and it is designed for low-cost, low-power and constrained resources devices such as 6LoWPAN motes.
5. In [29], Hull et al. analysed congestion through testbed experiments in a traditional WSN protocol stack with TinyOS where B-MAC and single-destination destination-sequenced distance vector (DSDV) are used. They concluded that wireless channel losses dominate buffer overflow and increase quickly with increasing offered load. On the other hand, in Chap. 3, we analyse congestion through analytical modelling and simulation in 6LoWPAN protocol stack by using Contiki OS and Cooja simulator. In contrast to Hull's conclusion, we have concluded that the majority of packets are lost due to buffer overflow as compared to channel loss. Also, we have concluded that the number of lost packets due to buffer drops increases with increasing offered load while the channel losses remain constant with different offered loads.
6. In the 6LoWPAN protocol stack, when the IPv6 packet size does not fit into a single 802.15.4 frame size, it must be fragmented into two or more fragments at the adaptation layer. When a node receives an initial (first) fragment, it stores the fragment in a buffer called the reassembly buffer and starts a parameter value called 'reassembly timeout' countdown. When the reassembly timeout expires and the node does not receive all fragments that belong to the same IPv6 packet, the received fragments are discarded. In Chap. 3 , we do congestion analysis for 6LoWPAN networks and we have demonstrated that the reassembly timeout parameter has a significant effect on network performance when congestion occurs. However, this parameter does not exist in the traditional WSN protocol stack.

For the reasons stated above (1–6), it is very important to design and build a novel congestion control mechanism based on the unique characteristics of the IEEE 802.15.4 standard, IPv6 and 6LoWPAN. Designing a congestion control algorithm should consider the 6LoWPAN protocol stack, i.e. the RPL routing protocol, the adaptation layer, IEEE 802.15.4 MAC and PHY layers. Also, it should consider the 6LoWPAN protocol stack parameters which impact on network performance when

congestion occurs such as the reassembly timeout parameter and RDC mechanism which is vital to save energy in power-constrained sensor nodes. The existing congestion control algorithms in 6LoWPAN networks use either traffic control or resource control to alleviate the congestion problem. It is important to use the positive aspects of both methods through the hybrid scheme where each strategy has advantages and disadvantages with different scenarios and network conditions.

As sensor nodes are connected to the Internet through the 6LoWPAN protocol stack to form the IoT, the applications of 6LoWPAN networks become ever wider. Also, the sensor nodes will be all around us in vehicles, smartphones, factories, building, seas, forests, etc. An estimate by Bell Labs is that from 50 to 100 billion things are expected to be connected to the Internet by 2020 [102], and the number of the wireless sensor devices will account for the majority of these [103]. Therefore, the sensor nodes may host many different application types simultaneously (event-based, continuous and query-based) with varied requirements. Some of them are real-time applications where the application data is time-critical and delay-constrained, while others are non-real-time applications. Some applications send very important data and losing this data is not permitted, e.g. medical applications (i.e. data may be important information about a patient case) and fire detection applications where data is very important and time constrained. This brings new challenges to the congestion control algorithms and mechanisms designed to be aware of data importance, packet priorities and application priorities as well as node priorities. To the best of our knowledge, none of the existing congestion control literature in WSNs and 6LoWPAN networks supports awareness of both application priorities and node priorities.

2.9 Conclusion

The 6LoWPAN protocol stack is one of the most important standards for the IoT where 6LoWPAN motes will account for the majority of the IoT 'things'. In this chapter, we have presented a survey of congestion control mechanisms in WSNs and 6LoWPAN networks to provide the state of art for the IoT. We have briefly overviewed the 6LoWPAN protocol stack. We gave a short review of the performance metrics, operating systems and simulators used to test and evaluate the proposed congestion control schemes. Also, we have presented an overview of congestion in WSNs and 6LoWPAN networks with respect to congestion detection, congestion notification and congestion control. Then, a review and summary of popular congestion control algorithms and mechanisms in WSNs is given. Also, a comparative review and summary of all the existing congestion control mechanisms in 6LoWPAN networks up to August 2016 is given. We have discussed these algorithms and explained the differences between congestion control in WSNs and 6LoWPAN networks. Also, we have explained the suitability and validity of WSN congestion control schemes for 6LoWPAN networks. Finally, we have derived some potential directions for conges-

tion control in 6LoWPAN networks in future work. In conclusion, we believe that a novel congestion control algorithm should

(i) Build upon the 6LoWPAN protocol stack and its characteristics.
(ii) Take into account the application requirements such as time constraint and reliability to support the IoT applications.
(iii) Support the hybrid application type which will be common in the IoT.
(iv) Be lightweight to support memory and processing capability-constrained sensor nodes.
(v) Support and be aware of RDC schemes to reduce energy consumption in energy-constrained sensor motes.
(vi) Apply the hybrid scheme for congestion control to utilize the benefits of using both traffic control and resource control strategies.
(vii) Be aware of data packet priority, application priority as well as node priority to support the IoT application requirements.

References

1. Ghaffari A (2015) Congestion control mechanisms in wireless sensor networks: a survey. J Netw Comput Appl 52:101–115
2. Kafi MA, Djenouri D, Ben-Othman J, Badache N (2014) Congestion control protocols in wireless sensor networks: a survey. IEEE Commun Surv Tutor 16(3):1369–1390
3. Flora DJ, Kavitha V, Muthuselvi M (2011) A survey on congestion control techniques in wireless sensor networks. In: Proceedings of international conference on emerging trends in electrical and computer technology (ICETECT). IEEE, pp 1146–1149
4. Yuan H, Yugang N, Fenghao G (2014) Congestion control for wireless sensor networks: a survey. In: Proceedings of the 26th Chinese control and decision conference (2014 CCDC). IEEE, pp 4853–4858
5. Pant N, Singh M, Kumar P (2014) Traffic and resource based methods for congestion control in wireless sensor networks: a comparative analysis. In: Proceedings of 6th international conference on adaptive science and technology (ICAST). IEEE, pp 1–6
6. Zhao J, Wang L, Li S, Liu X, Yuan Z, Gao Z (2010) A survey of congestion control mechanisms in wireless sensor networks. In: Proceedings of 6th international conference on intelligent information hiding and multimedia signal processing (IIH-MSP). IEEE, pp 719–722
7. Gowthaman P, Chakravarthi R (2013) Survey on various congestion detection and control protocols in wireless sensor networks. Int J Adv Comput Eng Commun Technol (IJACECT) 2(4):15–19
8. Chakravarthi R, Gomathy C, Sebastian SK, Pushparaj K, Mon VB (2010) A survey on congestion control in wireless sensor networks. Int J Comput Sci Commun 1(1):161–164
9. Budhwar P (2015) A survey of transport layer protocols for wireless sensor networks. J Emerg Technol Innov Res (JETIR) 2:985–991
10. Sergiou C, Antoniou P, Vassiliou V (2014) A comprehensive survey of congestion control protocols in wireless sensor networks. IEEE Commun Surv Tutor 16(4):1839–1859
11. Han Z, Niyato D, Saad W, Başar T, Hjørungnes A (2012) Game theory in wireless and communication networks: theory, models, and applications. Cambridge University Press, Cambridge
12. Matsumoto A, Szidarovszky F (2016) Game theory and its applications. Springer, Berlin

13. Tzeng G-H, Huang J-J (2011) Multiple attribute decision making: methods and applications. CRC Press, Boca Raton
14. Kuo Y, Yang T, Huang G-W (2008) The use of grey relational analysis in solving multiple attribute decision-making problems. Comput Ind Eng 55(1):80–93
15. Kelly FP, Maulloo AK, Tan DK (1998) Rate control for communication networks: shadow prices, proportional fairness and stability. J Oper Res Soc 49(3):237–252
16. Tychogiorgos G, Leung KK (2014) Optimization-based resource allocation in communication networks. Comput Netw 66:32–45
17. Wang L, Kuo G-S (2013) Mathematical modeling for network selection in heterogeneous wireless networks—a tutorial. IEEE Commun Surv Tutor 15(1):271–292
18. Srikant R, Ying L (2013) Communication networks: an optimization, control, and stochastic networks perspective. Cambridge University Press, Cambridge
19. Yinbiao S et al (2014) Internet of things: wireless sensor networks. White paper. International Electrotechnical Commission (IEC)
20. Reena P, Jacob L (2007) Hop-by-hop versus end-to-end congestion control in wireless multi-hop UWB networks. In: Proceedings of international conference on advanced computing and communications (ADCOM 2007). IEEE, pp 255–261
21. Heimlicher S, Nuggehalli P, May M (2007) End-to-end vs. hop-by-hop transport. SIGMETRICS Perform Eval Rev 35(3):59–60
22. Heimlicher S, Karaliopoulos M, Levy H, May M (2007) End-to-end vs. hop-by-hop transport under intermittent connectivity. In: Proceedings of the 1st international conference on autonomic computing and communication systems. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p 20
23. Kafi MA, Djenouri D, Othman JB, Ouadjaout A, Badache N (2014) Congestion detection strategies in wireless sensor networks: a comparative study with testbed experiments. Procedia Comput Sci 37:168–175
24. Yin X, Zhou X, Huang R, Fang Y, Li S (2009) A fairness-aware congestion control scheme in wireless sensor networks. IEEE Trans Veh Technol 58(9):5225–5234
25. Wan C-Y, Eisenman SB, Campbell AT (2003) CODA: congestion detection and avoidance in sensor networks. In: Proceedings of the 1st international conference on embedded networked sensor systems. ACM, pp 266–279
26. Sankarasubramaniam Y, Akan ÖB, Akyildiz IF (2003) ESRT: event-to-sink reliable transport in wireless sensor networks. In: Proceedings of the 4th ACM international symposium on mobile ad hoc networking and computing. ACM, pp 177–188
27. Michopoulos V, Guan L, Oikonomou G, Phillips I (2012) DCCC6: duty cycle-aware congestion control for 6LoWPAN networks. In: Proceedings of international conference on pervasive computing and communications workshops (PERCOM workshops). IEEE, pp 278–283
28. Deshpande VS, Chavan PP, Wadhai VM, Helonde JB (2012) Congestion control in wireless sensor networks by using differed reporting rate. In: Proceedings of world congress on information and communication technologies (WICT). IEEE, pp 209–213
29. Hull B, Jamieson K, Balakrishnan H (2004) Mitigating congestion in wireless sensor networks. In: Proceedings of the 2nd international conference on embedded networked sensor systems. ACM, pp 134–147
30. Wang C, Li B, Sohraby K, Daneshmand M, Hu Y (2007) Upstream congestion control in wireless sensor networks through cross-layer optimization. IEEE J Sel Areas Commun 25(4):786–795
31. Zawodniok M, Jagannathan S (2007) Predictive congestion control protocol for wireless sensor networks. IEEE Trans Wirel Commun 6(11):3955–3963
32. Jaiswal S, Yadav A (2013) Fuzzy based adaptive congestion control in wireless sensor networks. In: Proceedings of 6th international conference on contemporary computing (IC3). IEEE, pp 433–438
33. Sergiou C, Vassiliou V, Paphitis A (2013) Hierarchical tree alternative path (HTAP) algorithm for congestion control in wireless sensor networks. Ad Hoc Netw 11(1):257–272

34. Kim H-S, Paek J, Bahk S (2015) QU-RPL: queue utilization based RPL for load balancing in large scale industrial applications. In: Proceedings of 12th annual IEEE international conference on sensing, communication, and networking (SECON). IEEE, pp 265–273
35. Castellani AP, Rossi M, Zorzi M (2014) Back pressure congestion control for CoAP/6LoWPAN networks. Ad Hoc Netw 18:71–84
36. Huang J-M, Li C-Y, Chen K-H (2009) TALONet: a power-efficient grid-based congestion avoidance scheme using multi-detouring technique in wireless sensor networks. In: Proceedings of wireless telecommunications symposium (WTS). IEEE, pp 1–6
37. Al-Kashoash HAA, Al-Nidawi Y, Kemp AH (2016) Congestion-aware RPL for 6LoWPAN networks. In: Proceedings of wireless telecommunications symposium (WTS 2016). IEEE, pp 1–6
38. Wan J, Xu X, Feng R, Wu Y (2009) Cross-layer active predictive congestion control protocol for wireless sensor networks. Sensors 9(10):8278–8310
39. Rangwala S, Gummadi R, Govindan R, Psounis K (2006) Interference-aware fair rate control in wireless sensor networks. ACM SIGCOMM Comput Commun Rev 36(4):63–74
40. Fang W-W, Chen J-M, Shu L, Chu T-S, Qian D-P (2010) Congestion avoidance, detection and alleviation in wireless sensor networks. J Zhejiang Univ Sci C 11(1):63–73
41. Sheu J-P, Hu W-K (2008) Hybrid congestion control protocol in wireless sensor networks. In: Proceedings of vehicular technology conference (VTC). IEEE, pp 213–217
42. Kang J, Zhang Y, Nath B (2007) TARA: topology-aware resource adaptation to alleviate congestion in sensor networks. IEEE Trans Parallel Distrib Syst 18(7):919–931
43. Lee J-H, Jung I-B (2010) Adaptive-compression based congestion control technique for wireless sensor networks. Sensors 10(4):2919–2945
44. Sheu JP, Hsu CX, Ma C (2015) A game theory based congestion control protocol for wireless personal area networks. In: Proceedings of 39th annual computer software and applications conference (COMPSAC), vol 2
45. Ma C, Sheu J-P, Hsu C-X (2015) A game theory based congestion control protocol for wireless personal area networks. J Sens
46. Fahmy HMA (2016) Simulators and emulators for WSNs. Wireless sensor networks. Springer, Berlin, pp 381–491
47. Levis P, Madden S, Polastre J, Szewczyk R, Whitehouse K, Woo A, Gay D, Hill J, Welsh M, Brewer E et al (2005) TinyOS: an operating system for sensor networks. Ambient intelligence. Springer, Berlin, pp 115–148
48. Dunkels A, Grönvall B, Voigt T (2004) Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of 29th annual IEEE international conference on local computer networks. IEEE, pp 455–462
49. Dunkels A (2009) Contiki: bringing IP to sensor networks. ERCIM News 76:2009
50. Dunkels A, Eriksson J, Finne N, Tsiftes N (2011) Powertrace: network-level power profiling for low-power wireless networks. Swedish Institute of Computer Science (SICS), Technical report
51. Thingsquare (2016) Why choose Contiki. http://www.contiki-os.org/
52. Dunkels A, Schmidt O, Voigt T, Ali M (2006) Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In: Proceedings of the 4th international conference on embedded networked sensor systems. ACM, pp 29–42
53. Baccelli E, Hahm O, Gunes M, Wahlisch M, Schmidt TC (2013) RIOT OS: towards an OS for the internet of things. In: Proceedings of the 32nd international conference on computer communications (INFOCOM). IEEE, pp 79–80
54. Will H, Schleiser K, Schiller J (2009) A real-time kernel for wireless sensor networks employed in rescue scenarios. In: Proceedings of the 34th conference on local computer networks (LCN). IEEE, pp 834–841
55. Levis P, Lee N (2003) TOSSIM: a simulator for TinyOS networks. UC Berkeley, vol 24
56. Osterlind F, Dunkels A, Eriksson J, Finne N, Voigt T (2006) Cross-level sensor network simulation with COOJA. In: Proceedings of 31st IEEE conference on local computer networks. IEEE, pp 641–648

57. Stehlık M (2011) Comparison of simulators for wireless sensor networks. Master's thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic
58. Österlind F, Eriksson J, Dunkels A (2010) COOJA TimeLine: a power visualizer for sensor network simulation. In: Proceedings of the 8th ACM conference on embedded networked sensor systems. ACM, pp 385–386
59. Downard IT (2004) Simulating sensor networks in NS-2. DTIC document.Technical report
60. Henderson TR, Lacage M, Riley GF, Dowell C, Kopena J (2008) Network simulations with the NS-3 simulator. SIGCOMM Demonstr 15:17
61. Simon G, Volgyesi P, Maróti M, Lédeczi Á (2003) Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In: Proceedings of IEEE aerospace conference, vol 3, pp 1339–1346
62. Chang X (1999) Network simulations with OPNET. In: Proceedings of the 31st conference on winter simulation: simulation—a bridge to the future-volume 1. ACM, pp 307–314
63. Varga A (2001) The OMNeT++ discrete event simulation system. In: Proceedings of the European simulation multiconference (ESM'2001), pp 185–192
64. Kirsche M, Hartwig J (2013) A 6LoWPAN model for OMNeT++: poster abstract. In: Proceedings of the 6th international ICST conference on simulation tools and techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp 330–333
65. Ee CT, Bajcsy R (2004) Congestion control and fairness for many-to-one routing in sensor networks. In: Proceedings of the 2nd international conference on embedded networked sensor systems. ACM, pp 148–161
66. Chen S, Zhang Z (2006) Localized algorithm for aggregate fairness in wireless sensor networks. In: Proceedings of the 12th annual international conference on mobile computing and networking. ACM, pp 274–285
67. Chen S, Yang N (2006) Congestion avoidance based on lightweight buffer management in sensor networks. IEEE Trans Parallel Distrib Syst 17(9):934–946
68. Monowar MM, Rahman MO, Hong CS (2008) Multipath congestion control for heterogeneous traffic in wireless sensor network. In: Proceedings of 10th international conference on advanced communication technology (ICACT), vol 3. IEEE, pp 1711–1715
69. Wang G, Liu K (2009) Upstream hop-by-hop congestion control in wireless sensor networks. In: Proceedings of 20th international symposium on personal, indoor and mobile radio communications. IEEE, pp 1406–1410
70. Alam MM, Hong CS (2009) CRRT: congestion-aware and rate-controlled reliable transport in wireless sensor networks. IEICE Trans Commun 92(1):184–199
71. Brahma S, Chatterjee M, Kwiat K (2010) Congestion control and fairness in wireless sensor networks. In: Proceedings of 8th IEEE international conference on pervasive computing and communications workshops (PERCOM workshops). IEEE, pp 413–418
72. Heikalabad SR, Ghaffari A, Hadian MA, Rasouli H (2011) DPCC: dynamic predictive congestion control in wireless sensor networks. IJCSI Int J Comput Sci Issues 8(1)
73. Munir SA, Bin YW, Biao R, Jian M (2007) Fuzzy logic based congestion estimation for QoS in wireless sensor network. In: Proceedings of wireless communications and networking conference (WCNC 2007). IEEE, pp 4336–4341
74. Wei J, Fan B, Sun Y (2012) A congestion control scheme based on fuzzy logic for wireless sensor networks. In: Proceedings of 9th international conference on fuzzy systems and knowledge discovery (FSKD). IEEE, pp 501–504
75. He T, Ren F, Lin C, Das S (2008) Alleviating congestion using traffic-aware dynamic routing in wireless sensor networks. In: Proceedings of 5th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks (SECON'08). IEEE, pp 233–241
76. Woo A, Tong T, Culler D (2003) Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the 1st international conference on embedded networked sensor systems. ACM, pp 14–27

77. Rahman MO, Monowar MM, Hong CS (2008) A QoS adaptive congestion control in wireless sensor network. In: Proceedings of 10th international conference on advanced communication technology (ICACT), vol 2. IEEE, pp 941–946
78. Wang C, Sohraby K, Li B (2005) SenTCP: a hop-by-hop congestion control protocol for wireless sensor networks. In: Proceedings of IEEE INFOCOM, 2005, pp 107–114
79. Sergiou C, Vassiliou V, Paphitis A (2014) Congestion control in wireless sensor networks through dynamic alternative path selection. Comput Netw 75:226–238
80. Dasgupta R, Mukherjee R, Gupta A (2015) Congestion avoidance topology in wireless sensor network using Karnaugh map. In: Proceedings of applications and innovations in mobile computing (AIMoC). IEEE, pp 89–96
81. Razzaque MA, Hong CS (2009) Congestion detection and control algorithms for multipath data forwarding in sensor networks. In: Proceedings of 11th international conference on advanced communication technology (ICACT), vol 1. IEEE, pp 651–653
82. Sergiou C, Vassiliou V (2014) HRTC: a hybrid algorithm for efficient congestion control in wireless sensor networks. In: Proceedings of 6th international conference on new technologies, mobility and security (NTMS). IEEE, pp 1–5
83. Tassiulas L, Ephremides A (1992) Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Trans Autom Control 37(12):1936–1948
84. Hellaoui H, Koudil M (2015) Bird flocking congestion control for CoAP/RPL/6LoWPAN networks. In: Proceedings of the workshop on IoT challenges in mobile and industrial systems. ACM, pp 25–30
85. Shelby Z, Hartke K, Bormann C (2014) The constrained application protocol (CoAP). IETF RFC 7252
86. Kim H-S, Kim H, Paek J, Bahk S (2016) Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks. IEEE Trans Mob Comput
87. Winter T, Thubert P, Brandt A, Hui J, Kelsey R (2012) RPL: IPv6 routing protocol for low-power and lossy networks. IETF, RFC 6550
88. Thubert P (2012) Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552
89. Gnawali O, Levis P (2010) The ETX objective function for RPL. Internet draft: draft-gnawali-roll-etxof-00
90. Tang W, Ma X, Huang J, Wei J (2015) Toward improved RPL: a congestion avoidance multipath routing protocol with time factor for wireless sensor networks. J Sens 2016
91. Lodhi MA, Rehman A, Khan MM, Hussain FB (2015) Multiple path RPL for low power lossy networks. In: Proceedings of Asia Pacific conference on wireless and mobile (APWiMob). IEEE, pp 279–284
92. Ha M, Kwon K, Kim D, Kong P-Y (2014) Dynamic and distributed load balancing scheme in multi-gateway based 6LoWPAN. In: Proceedings of international conference on internet of things (iThings), green computing and communications (GreenCom) and cyber, physical and social computing (CPSCom). IEEE, pp 87–94
93. Liu X, Guo J, Bhatti G, Orlik P, Parsons K (2013) Load balanced routing for low power and lossy networks. In: Proceedings of wireless communications and networking conference (WCNC). IEEE, pp 2238–2243
94. Guo J, Liu X, Bhatti G, Orlik P, Parsons K (2013) Load balanced routing for low power and lossy networks, 21 January 2013, US Patent Application 13/746,173
95. Tang W, Wei Z, Zhang Z, Zhang B (2014) Analysis and optimization strategy of multipath RPL based on the COOJA simulator. Int J Comput Sci Issues (IJCSI) 11(5):27–30
96. Kamgueu PO, Nataf E, Ndié TD, Festor O (2013) Energy-based routing metric for RPL. [Research report] RR-8208, INRIA, p 14
97. Zheng T, Ayadi A, Jiang X (2011) TCP over 6LoWPAN for industrial applications: an experimental study. In: Proceedings of 4th IFIP international conference on new technologies, mobility and security (NTMS). IEEE, pp 1–4

98. Ayadi A, Maillé P, Ros D (2011) TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption. In: Proceedings of 4th IFIP international conference on new technologies, mobility and security (NTMS). IEEE, pp 1–5
99. Kim H-S, Im H, Lee M-S, Paek J, Bahk S (2015) A measurement study of TCP over RPL in low-power and lossy networks. J Commun Netw 17(6):647–655
100. Antoniou P, Pitsillides A, Blackwell T, Engelbrecht A, Michael L (2013) Congestion control in wireless sensor networks based on bird flocking behavior. Comput Netw 57(5):1167–1191
101. Michopoulos V, Guan L, Oikonomou G, Phillips I (2011) A comparative study of congestion control algorithms in IPv6 wireless sensor networks. In: Proceedings of international conference on distributed computing in sensor systems and workshops (DCOSS). IEEE, pp 1–6
102. Weldon M (2016) The future X network: a Bell Labs perspective. CRC Press, Boca Raton
103. Dunlap J (2011) From billing and technology convergence to ecosystem convergence: Why M2M matters to your business. Pipeline: Technol Serv Provid 8(7):14

Chapter 3
Comprehensive Congestion Analysis for 6LoWPANs

3.1 Introduction

This chapter presents a comprehensive congestion analysis for 6LoWPAN network through analytical modelling, simulations and testbed results. Congestion occurs when multiple sensor nodes start to send packets concurrently at high data rate or when a node relays many flows across the network. Thus, link collision on the wireless channel and packet overflow at buffer nodes occur in the network [1]. Recently, a few papers have been presented to address congestion in 6LoWPAN networks [2–5], but none considered congestion assessment and analysis. In [6], Hull et al. did a testbed experiment in a traditional WSN protocol stack with TinyOS where B-MAC and the single destination DSDV (Destination Sequenced Distance Vector) routing protocol are used. In this chapter, experiments in 6LoWPAN wireless sensor networks using the 6LoWPAN protocol stack and Contiki OS are considered.

The remainder of the chapter is organized as follows: An analytical modelling of congestion for 6LoWPAN is developed in Sect. 3.2. In this section, we did simulations to validate our proposed model with different scenarios. Section 3.3 presents an extensive congestion analysis for 6LoWPAN through simulations with different scenarios and various parameters. Section 3.4 presents a testbed-based congestion analysis for 6LoWPAN with different scenarios (indoor and outdoor) and various parameters. Finally, Sect. 3.5 concludes this chapter.

3.2 Analytical Modelling of Congestion for 6LoWPAN

In this section, we propose an analytical model to study the 6LoWPAN network performance in the presence of congestion (e.g. how many packets are lost due to buffer overflow and the average number of packets received by a sink node) using Markov chain analysis and queuing theory. Queuing theory is one of the most

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_3

important tools for studying and analysing computer network performance [7, 8]. Queuing analysis is considered as a special case of Markov chains. It deals with queues (in nodes' buffers) where packets compete to be processed by servers (e.g. sensor nodes). Also, we calculate the IEEE 802.15.4 effective channel capacity based on Contiki OS implementation with and without the occurrence of wireless channel collisions. Finally, we validate our modelling with different parameters, i.e. number of nodes, buffer sizes and offered loads, through simulation using Contiki OS [9] and Cooja simulator [10].

3.2.1 System Model

In 6LoWPAN networks, RPL [11] is responsible for constructing the network topology. Three types of nodes are defined: sink (root) nodes which provide connectivity to other networks, intermediate nodes which forward packets to the sink and leaf nodes. Consider a network of M leaf nodes, $L_1, \ldots, L_k, \ldots, L_M$, one intermediate node, I_l, and one sink node, S. The topology of the network is shown in Fig. 3.1. Each node in the network has a buffer of size B packets. We assume that the wireless channel capacity (CC_b) in bits per second is distributed among nodes as the intermediate node has half portion of the leaf node (i.e. the service (forwarding) rate of intermediate node is half of the forwarding rate of leaf node). The reason is that the radio of the intermediate node is receiving and transmitting at the same time whereas the leaf node's radio is just sending traffic. Also, we assume that the sensor nodes run the contention-based IEEE 802.15.4 MAC protocol with unslotted CSMA/CA as access mechanism.

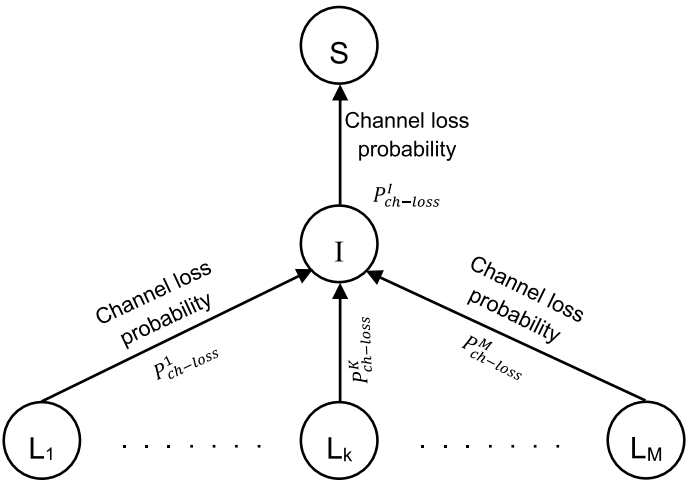

Fig. 3.1 Network topology

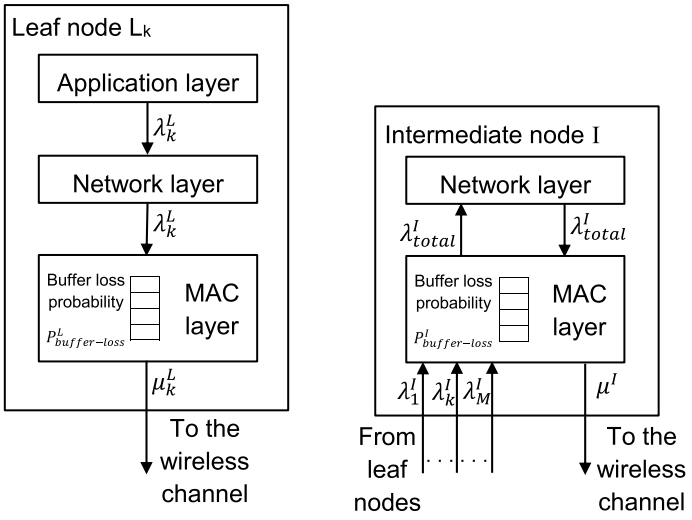

Fig. 3.2 Leaf and intermediate nodes model

When congestion occurs, the packets are lost either at the sensor node (buffer overflow) or on the wireless channel. Figure 3.2 shows the node model for the leaf and intermediate nodes. In Fig. 3.2, the applications in the leaf nodes, $L_1, L_2, \ldots, L_M$, generate packets at an average data rate of $\lambda_1, \lambda_2, \ldots, \lambda_M$, respectively. Then the packets are stored in the MAC's buffer to be transmitted by the MAC protocol to the intermediate node I_l. We assume that the leaf nodes $L_1, L_2, \ldots, L_M$, transmit the packets with an average departure rate of $\mu_1^L, \mu_2^L, \ldots, \mu_M^L$ respectively. Before the packets arrive at the intermediate node I_l, a number of packets are lost on the wireless channel with a probability $P_{ch-loss}^j$ where $j = 1, 2, \ldots, k, \ldots, M$. Then, packets arrive at the node I_l with an average rate of $\lambda_1^I, \lambda_2^I, \ldots, \lambda_M^I$ form nodes $L_1, L_2, \ldots, L_M$ respectively as:

$$\lambda_j^I = (1 - P_{ch-loss}^j)\mu_j^I, \tag{3.1}$$

where $j = 1, 2, \ldots, k, \ldots, M$, and the total arrival packets at node I_l is $\lambda_{total}^I = \sum_{j=1}^{M} \lambda_j^I$. When node I_l receives the packets, it stores them in its buffer to forward them later to the sink node S with an average departure rate of μ^I.

In the following subsections, we develop a model to calculate the probabilities of packet buffer loss and channel loss in the network.

3.2.2 Buffer Loss Probability

In this subsection, we perform Markov chain analysis to calculate the buffer loss probability ($P_{buffer-loss}$). The states of the Markov chain represent the number of packets stored in the buffer. Consider the packet arrivals to be Poisson distributed with a mean rate of λ (packet/s) and a mean service time of each packet is assumed $1/\mu$. The buffer can be modelled as an M/M/1/B model where B represents the buffer size. We take the time step of state transitions equal to the inverse of the maximum data rate, i.e. the channel capacity in packet per second (CC_p), as follows:

$$T = \frac{1}{CC_p}, \tag{3.2}$$

where $CC_p = CC_b/PL$ and PL is packet length in bits.

At a given time step a maximum one packet could arrive at or leave the buffer. We assume that at a certain time step, the probability of packet arrival is P_{arr} and the probability that a packet leaves the queue is P_{dep}. The state transition diagram for the M/M/1/B queue is shown in Fig. 3.3 and the transition matrix, which is $(\mathrm{B}+1)\times(\mathrm{B}+1)$ dimensions, is given by:

$$P = \begin{bmatrix} u & z & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ x & y & z & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & x & y & z & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & x & y & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & y & z & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & x & y & z & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & x & y & z \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & x & v \end{bmatrix}$$

where $v = 1 - x, u = 1 - z, x = (1 - P_{arr})P_{dep}, y = P_{arr}P_{dep} + (1 - P_{arr})(1 - P_{dep})$ and $z = P_{arr}(1 - P_{dep})$.

The equilibrium (steady state) distribution vector of the transition matrix is

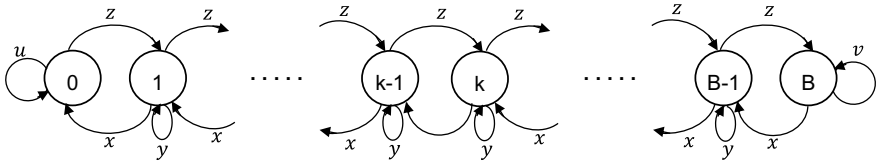

Fig. 3.3 State transition diagram

$$\pi = [\pi_0 \quad \pi_1 \quad \pi_2 \quad \dots \quad \pi_B]$$

To simplify the analysis, consider that the applications at the leaf nodes generate packets with equal data rate of an average of λ where $\lambda = \lambda_1 = \lambda_2 = \cdots = \lambda_M$.

For the leaf nodes, the probability of packet arrival at the buffer is $P^L_{arr} = \lambda / CC_P$ and the probability of packet departure is $P^L_{dep} = \mu^L_{max} / CC_p$.

As the channel capacity is distributed among nodes with leaf node has twice portion of intermediate node, the maximum departure rate at a leaf node is $\mu^L_{max} = 2CC_p/(2M+1)$. Packets are lost when the leaf node's buffer is full and a packet arrives but does not leave the buffer. Thus, the average number of lost packets per time step T at each leaf node's buffer is as follows:

$$L^T_{leaf} = \pi^L_B \times P^L_{arr} \times (1 - P^L_{dep}), \tag{3.3}$$

and the average number of lost packets per second at each leaf node's buffer is as follows:

$$L_{leaf} = \pi^L_B \times P^L_{arr} \times (1 - P^L_{dep}) \times CC_p. \tag{3.4}$$

Thus, the probability of packet loss at the leaf node's buffer is given by:

$$P^L_{buffer-loss} = \frac{L_{leaf}}{\lambda}. \tag{3.5}$$

For the intermediate node I_l, the probability of packet arrival at the buffer is $P^I_{arr} = \lambda^I_{total} / CC_p$ and the probability of packet departure is $P^I_{dep} = \mu^I_{max} / CC_p$ where μ^I_{max} is as follows:

$$\mu^I_{max} = \begin{cases} CC_p/(2M+1) & \text{if } \mu^L = \mu^L_{max} \\ CC_p - M\,\mu^L & \text{if } \mu^L < \mu^L_{max} \end{cases},$$

where $\mu^L = (1 - P^L_{buffer-loss}) \times \lambda$.

Packets are lost when the intermediate node's buffer is full and a packet arrives but does not leave the buffer.Thus, the average number of lost packets per time step T at the intermediate node's buffer is as follows:

$$L^T_{inter.} = \pi^I_B \times P^I_{arr} \times (1 - P^I_{dep}), \tag{3.6}$$

and the average number of lost packets per second at the intermediate node's buffer is as follows:

$$L_{inter.} = \pi^I_B \times P^I_{arr} \times (1 - P^I_{dep}) \times CC_p. \tag{3.7}$$

Thus, the probability of packet loss at the intermediate node's buffer is given by:

$$P^I_{buffer-loss} = \frac{L_{inter.}}{\lambda^L_{total}}. \tag{3.8}$$

The total average number of lost packets per second at the buffers in the network is:

$$L_{buffer-loss} = M \times L_{leaf} + L_{inter.}, \tag{3.9}$$

and the total probability of packet loss at nodes' buffers in the network is:

$$P_{buffer-loss} = \frac{L_{buffer-loss}}{M \times \lambda}. \tag{3.10}$$

Substituting Eqs. (3.4), (3.7) and (3.9) in Eq. (3.10), we get:

$$P_{buffer-loss} = \frac{[M \times \pi_B^L \times (CC_p - \mu^L)] + [\pi_B^I \times \lambda_{total}^I \times (CC_p - \mu^I)]}{M \times \lambda \times CC_p}. \tag{3.11}$$

The average number of received packets per second at sink node (λ^S) is:

$$\lambda^S = (1 - P_{ch-loss}^I)(1 - P_{buffer-loss}^I) \times \lambda_{total}^I. \tag{3.12}$$

From Eq. (3.11), we can notice that the probability of packet loss due to buffer overflow depends on number of leaf nodes, buffer size (which is implicit included in π_B), sending rate of leaf nodes and most significant the channel capacity.

3.2.3 Channel Loss Probability

In the IEEE 802.15.4 CSMA/CA, packets are assumed to be lost in the wireless channel due to two reasons:

(1) Channel access failure: when a node tries to transmit a packet, it performs a CCA to sense the wireless channel. If the channel is idle, then the node begins to transmit. Otherwise, it increments the value of two parameters; the number of backoffs (NB) and the backoff exponent (BE). After that, the node waits for a random time in the range $[0, (2^{BE} - 1)]$ backoff unit periods before it does the CCA again. Each backoff unit period equals 20 symbols $\times$ 16 µs/symbol [12]. This process continues until the value of BE exceeds the value of *macMaxCSMABackoffs* parameter. Then, the packet is discarded due to channel access failure.
(2) Maximum number of retransmission limit: when the node sends a packet, it waits for an ACK packet. If the node does not receive the ACK packet due to a collision or an ACK timeout expires, then, it increments the retransmission count and tries to retransmit the packet. If the number of retransmissions reaches the maximum number of the retransmissions parameter *macMaxFrameRetries*, then, the packet is dropped. Channel access failure happens when a packet fails to obtain clear channel within $(m + 1)$ backoffs. Furthermore, a packet is discarded if the transmission fails due to repeated collisions after $(n + 1)$ attempts. Thus, the probability of channel loss for node j ($P_{ch-loss}^j$) is:

$$P^{j}_{ch-loss} = P^{j}_{caf} + P^{j}_{mrl}, \tag{3.13}$$

where P^{j}_{caf} is the probability of packet loss due to channel access failure, P^{j}_{mrl} is the probability of packet loss due to the maximum number of retransmission limit, m is the maximum number of backoffs and n is the maximum number of retransmissions.

In [13], Di Marco et al. have developed an analytical model to calculate P^{j}_{caf} and P^{j}_{mrl} for unslotted IEEE 802.15.4 as follows:

$$P^{j}_{caf} = \frac{P^{m+1}_{cca,j}(1 - (P_{coll,j}(1 - P^{m+1}_{cca,j}))^{n+1})}{1 - P_{coll,j}(1 - P^{m+1}_{cca,j})}, \tag{3.14}$$

$$P^{j}_{mrl} = (P_{coll,j}(1 - P^{m+1}_{cca,j}))^{n+1}, \tag{3.15}$$

where $P_{cca,j}$ is the probability that CCA is busy and $P_{coll,j}$ is the probability that a transmitted packet encounters a collision for node j. For more details about P^{j}_{caf} and P^{j}_{mrl}, please refer to [13].

3.2.4 Contiki-Based IEEE 802.15.4 Effective Channel Capacity

The developed buffer loss probability model depends on a set of parameters; one of them is the actual channel capacity. We do validation of our proposed modelling with Contiki OS and Cooja simulator. In this section, we estimate the actual channel capacity based on Contiki 3.0 OS implementation. The IEEE 802.15.4 standard supports a maximum data rate at the physical layer of 250 kbps in the 2.4 GHz band. In reality, the effective data rate is smaller than 250 kbps and its actual value varies with time due to operation of the channel access algorithm, overhead of ACK packet transmission, collisions and number of active nodes. The Contiki OS uses unslotted CSMA/CA as a channel access mechanism and it implements the data link layer as three sublayers: framer, RDC and medium access control (MAC). In the simulation, these are 802.15.4 (framer), nullrdc (RDC) and CSMA (MAC).

When the application generates packets, they are passed down to the MAC layer through the network layer and sicslowpan layer. When the MAC layer receives the packets, it queues them in its buffer. After that, the MAC layer sends the packets to the nullrdc layer which calls a function called 'NETSTACK_RADIO.prepare' to prepare the packet with the radio. While preparing, if the radio is currently receiving a packet or it has already received a packet that needs to be read before sending an ACK packet, then, the radio returns 'TX_COLLISION'. Otherwise, the nullrdc calls another function called 'NETSTACK_RADIO.transmit' to send the already prepared packet. Next, the nullrdc layer waits for ACK packet for a time called *macAckWaitDuration*. If the ACK is received during the wait time, the nullrdc waits

again for a time called 'AFTER _ACK_DETECTED_WAIT_TIME' ($T_{A\_A\_D}$) and then returns 'TX_OK' to the CSMA layer. Otherwise, if *macAckWaitDuration* time ends and ACK is not received, the nullrdc returns 'TX_NOACK'. When the CSMA layer gets 'TX_OK', it dequeues the successfully transmitted packet and sends the next packet. Otherwise, if it gets 'TX _COLLISION' or 'TX_NOACK', the CSMA waits for a random backoff time in the range $[time, time + 2^{BE} \times time]$ where *time* is channel check interval which equals $1/channel\ check\ rate$. Then, after the backoff time ends, the CSMA layer retransmits the packet again. The MAC layer makes *macMaxFrameRetries* retransmission attempts and if unsuccessful, the packet is dropped. When the packet is received, it waits for a time called *turnaroundtime* and then it sends the ACK packet. Figure 3.4 shows the flow chart of this process.

Figure 3.5 illustrates the TimeLine of 6 motes where one of them sends packets to others. Clearly, we can see the packet transmission time, turnaround time, ACK transmission time and wait time which includes $T_{A\_A\_D}$ as well as other times. Also, we can see that when a collision occurs (two nodes transmit at the same time), the collided nodes wait for *macAckWaitDuration* plus a random backoff time before they try to transmit again.

Considering that a collision does not occur, the maximum effective data rate, EDR_{max}, that Contiki 3.0 OS can support is as follows:

$$EDR_{max} = \frac{N}{T_{nocoll}}, \tag{3.16}$$

where N is the data packet length (in bits) and T_{nocoll} is the actual time needed to transmit one data packet without collision. The maximum data packet length that IEEE 802.15.4 link can support is 127 bytes and T_{nocoll} is calculated as follows:

$$T_{nocoll} = T_{data} + turnaround\ time + T_{ACK} + T_{wait}, \tag{3.17}$$

where T_{data} and T_{ACK} are the amount of time required to transmit data packet and ACK packet respectively.

Turnaround time is the time required to switch between transmit and receive, or vice versa and T_{wait} includes $T_{A\_A\_D}$ and other times as shown in Fig. 3.5. Thus, EDR_{max} is calculated as:

$$EDR_{max} = \frac{127 \times 8}{(4.256 + 0.192 + 0.288 + 3.7)\,\text{ms}} \approx 120\,\text{kbps}.$$

In practice, transmissions typically suffer collisions. When a collision does occur, the actual data rate goes down as the probability of collision increases in the network. When a collision occurs, the mote retransmits the collided packet. This takes extra time which includes transmission time of the collided data packet, *macAckWaitDuration* and random backoff time as shown in Fig. 3.5. Thus, the actual data rate, *ADR*, with a probability of collision of $P_{collision}$ is as follows:

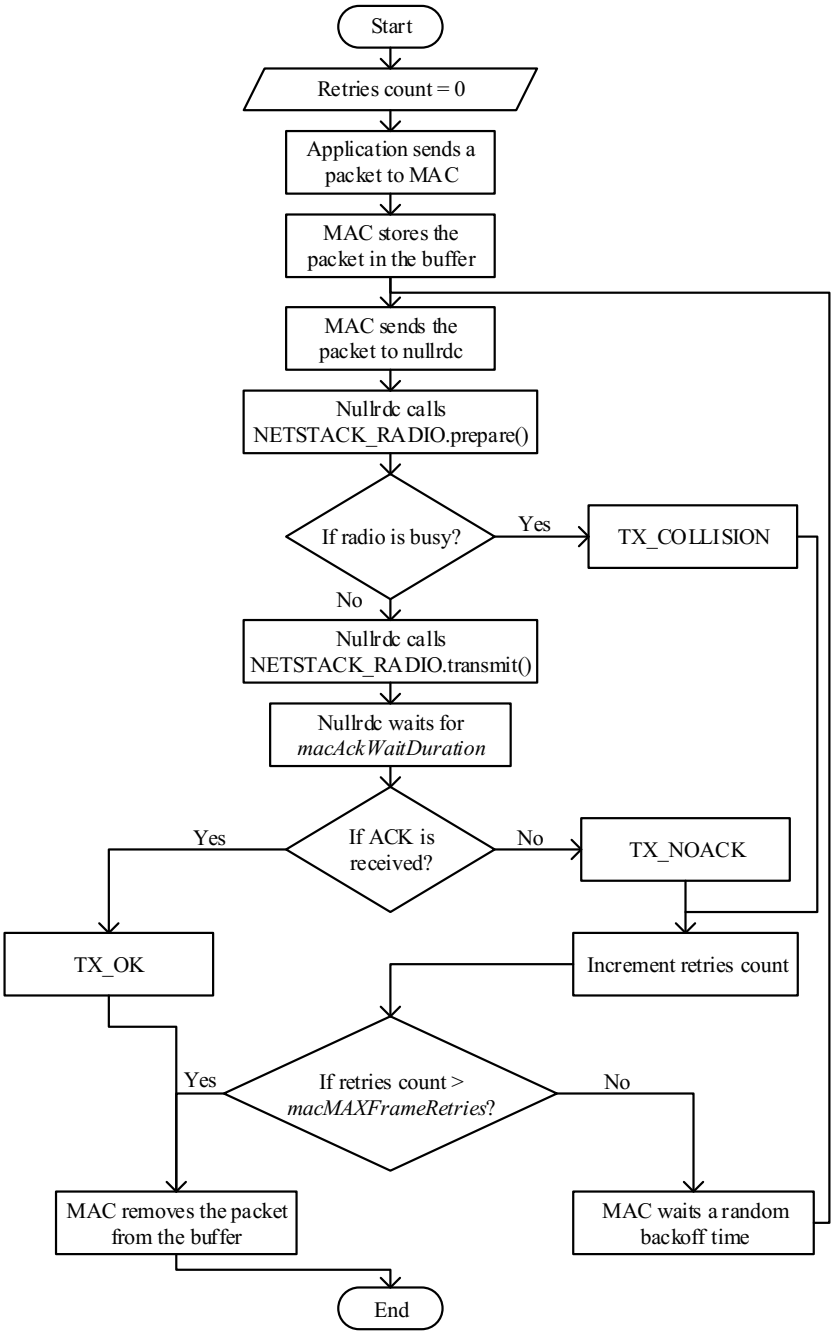

Fig. 3.4 Packet transmission process in Contiki OS

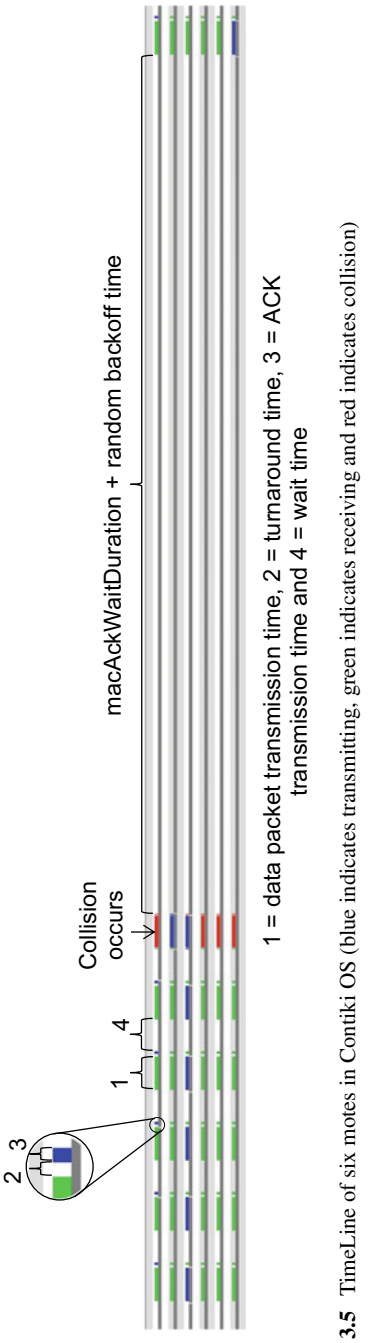

Fig. 3.5 TimeLine of six motes in Contiki OS (blue indicates transmitting, green indicates receiving and red indicates collision)

$$ADR = \frac{N}{(1 - P_{collision})T_{nocoll} + P_{collision}T_{coll}}, \tag{3.18}$$

where T_{coll} is the actual time needed to transmit one data packet within collision and one retransmission attempt and is calculated as:

$$T_{coll} = T_{data} + macAckWaitDuration + T_{backoff} + T_{nocoll}. \tag{3.19}$$

For example, with a probability of collision of 5%, the actual data rate can be calculated as follows:

$$ADR = \frac{127 \times 8}{[0.95 \times 8.436 + 0.05(4.256 + 0.4 + 125 + 8.436)]\,\text{ms}} \approx 68\,\text{kbps}.$$

When a node enters a backoff period and there are other active nodes located in the transmission range. The active nodes can utilize this period by sending their data packets. Thus, as there are active nodes during a backoff period of a collided node, the channel time is utilized and therefore, the actual channel capacity increases. Also, as the active nodes can detect the idle time of the wireless channel quickly, the channel utilization will be high and therefore, the actual channel capacity increases. Overall, the actual channel capacity is not constant and it varies according to network circumstances. The actual channel capacity is affected by many factors such as probability of collision, number of active nodes and the utilization rate of idle wireless channel time. In [14, 15], Sun et al. have developed effective channel capacity estimation of IEEE 802.15.4 beaconless mode without taking account of the random backoff time and collision occurrence. Also, in [16], Latré et al. have determined throughput of unslotted IEEE 802.15.4 with unreal assumptions (no losses due to collisions, no packets are lost due to buffer overflow, perfect channel with bit error rate of zero). In our modelling validation, we estimate the actual channel capacity value based on our simulation results.

3.2.5 Simulation Results

In this subsection, we present simulation results obtained through using Contiki 3.0 OS and Cooja simulator to validate our buffer loss probability modelling for varying number of leaf nodes, buffer sizes and various offered loads. The protocols and simulation parameters used in the simulation are shown in Table 3.1. In the simulation, we use Minimum Rank with Hysteresis Objective Function (MRHOF) as an objective function that uses ETX routing metric. Also, we use HC06 as a compression method which uses IPHC and NHC methods. The total duration time of each simulation is set to be 60 s and during the simulation time, each leaf node sends data packets periodically to the sink node at an offered load of 32 packet/s.

Table 3.1 Protocol stack and simulation parameters

Layer	Protocol	Parameter value
Application	Every leaf node periodically send packets to sink node	Offered load = 32 packet/s
Transport	UDP	
Network	uIPv6 + RPL	Objective function = MHROF
Adaptation	SICSlowpan layer	Compression method = HC06
Data link	CSMA (MAC layer) nullrdc (RDC layer) 802.15.4 (framer)	Buffer size = 10 packets *macMaxFrameRetries* = 3 *Channel check rate* = 8 Hz *macAckWaitDuration* = 0.4 ms $T_{A\_A\_D}$ = 0.6667 ms *macMinBE* = 0 *macMaxBE* = 3 Frame size = 127 bytes MAC reliability (ACK) = enabled
Physical	CC2420 RF transceiver	*Turnaround time* = 0.192 ms

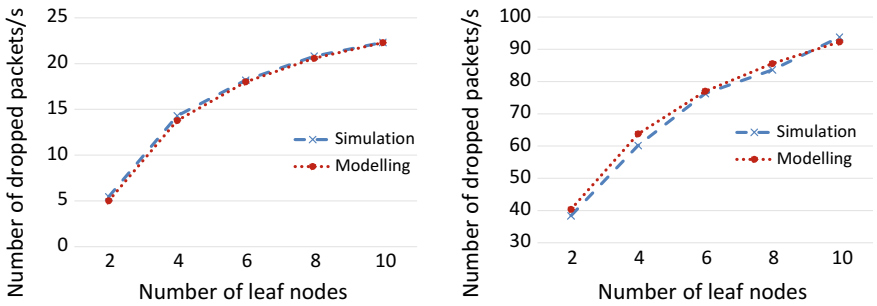

Fig. 3.6 Average number of dropped packets with different number of leaf nodes

In the first scenario, we change the number of leaf nodes from 2 to 4, 6, 8 and 10 where each leaf node's application generates packets with an average rate of 32 packet/s. Figure 3.6 shows the average number of dropped packets per second due to buffer overflow in the leaf node and the intermediate node estimated by simulation and using analytical modelling. From this figure, we can observe a good agreement between simulation and analytical results. Also, we can see that as the number of leaf nodes increases in the network, the number of lost packets due to buffer overflow increases in both leaf node and intermediate node. The reason is that as the number of leaf nodes increases, the portion of channel capacity for each node is reduced and therefore the departure rate of each node becomes lower and the probability of buffer loss increases.

Next, in the second scenario, we set the number of leaf nodes to 5 and we change the buffer size from 5 to 10, 15 and 20 packets. Figure 3.7 shows the average rate of

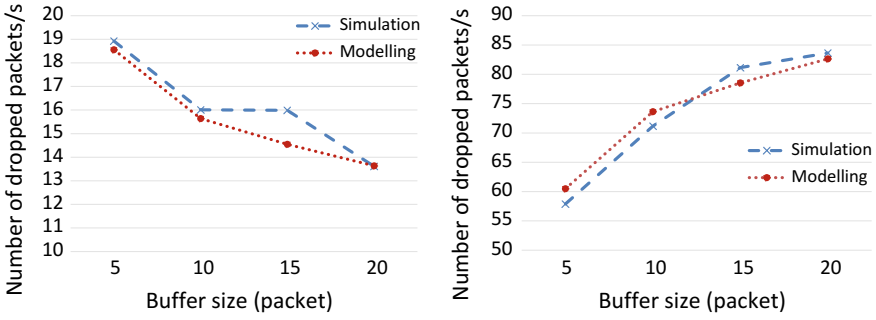

Fig. 3.7 Average number of dropped packets with different buffer sizes

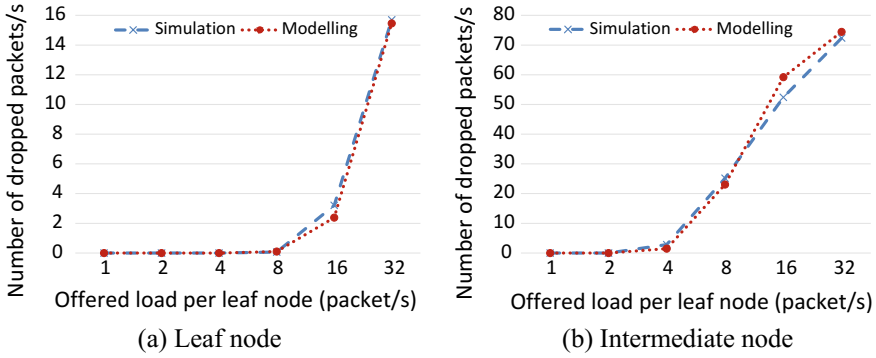

Fig. 3.8 Average number of dropped packets with various offered loads

packet loss due to buffer overflow in the leaf node and the intermediate node. We notice close correlation between simulation and modelling results. It is clear that as the buffer size increases, the average number of lost packets due to buffer overflow at leaf node decreases while it increases in the intermediate node. The reason is that when the buffer size is increased in the leaf node, the probability of buffer loss decreases and the departure rate of leaf node increases. As the departure rate of leaf node is increased, the arrival rate at the intermediate node is increased and therefore the probability of packet lost due to buffer overflow (i.e. buffer loss) becomes higher.

In the last scenario, we set the offered load to 1, 2, 4, 8, 16 and 32 packet/s and the number of leaf nodes to 5. Figure 3.8 shows the number of dropped packets at the buffers of the leaf and intermediate nodes every second with different offered loads. From this figure, we can see the similarity between the simulation and analytical modelling results. Also, we can see that when the offered load is increased, the average number of buffer dropped packets increases in the leaf and intermediate nodes.

Finally, Fig. 3.9 shows the average number of received packets at the sink node every second for the three scenarios above. From the figure, we notice that the

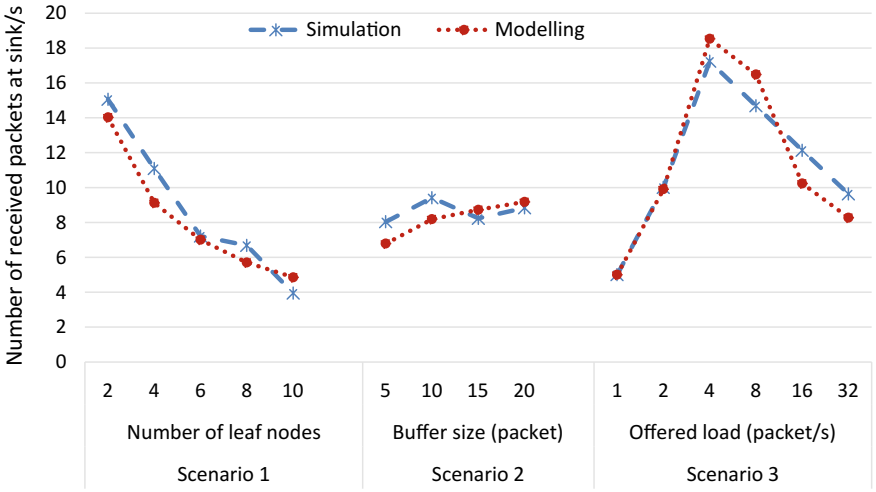

Fig. 3.9 Average number of received packets at sink node

simulation and analytical results have the same trend and a good consistency. Also, we can see that the number of received packets at the sink increases with the decreasing number of leaf nodes, increasing the buffer size and increasing the offered load until it reaches a certain rate (4 packet/s) after that the number of received packets at the sink starts decreasing.

Overall, the scenarios show that the analytical modelling results have a good agreement with the simulation results. Also, the simulation results show that our analytical modelling of congestion accurately models the buffer loss probability and the average number of received packets at the sink node. However, the derived analytical model is valid only for a network topology with one sink node, one intermediate node and a set of M leaf nodes as shown in Fig. 3.1. To make the analysis is valid for a general network topology (e.g. multi-sink and multi-intermediate nodes), we need to extend the modelling by considering how the channel capacity is distributed among multi-intermediate nodes and their leaf nodes.

3.3 Simulation-Based Congestion Analysis for 6LoWPAN

In order to assess the number of lost packets at the wireless channel as compared to at the sensor node (i.e. due to buffer overflow), experiments using Contiki OS and Cooja simulator with different network sizes and various offered loads were performed. In the networks, an average number of nodes per personal operating space (POS) is 4. The POS is defined as a physical space (coverage area) of a node since other nodes inside this area can communicate with the node [17]. These experiments have been executed with and without fragmentation which is implemented at the SICSlowpan

Table 3.2 Protocol stack

Layer	Protocol	Parameter value
Application	Every node periodically send packet to sink node	Simulation time = 30 min for each simulation
Transport	UDP	
Network	uIPv6 + RPL	Objective function = MHROF
Adaptation	SICSlowpan layer	Compression method = HC06
Data link	CSMA (MAC layer) Contikimac (RDC layer) 802.15.4 (framer)	Buffer size = 8 packets CCA count = 2 MAC retransmission = 3 Channel check rate = 8 Hz
Physical	CC2420 RF transceiver	Max. packet length = 127 byte

(adaptation) layer. The SICSlowpan layer performs two main functions: IPv6 header compression [18] and IPv6 fragmentation and reassembly [19]. Before an IPv6 packet is transmitted over an 802.15.4 link, the IPv6 header must be compressed by using a header compression mechanism. After compression, if the IPv6 packet size does not fit into a single 802.15.4 frame size, it must be fragmented. In each network, every node sends packets periodically to a single sink node. The protocol stack and simulation parameters which have been used in the experiments are shown in Table 3.2. In Contiki OS, all sending and receiving packets are stored in a common buffer called the Rime buffer which contains the application data and packet attributes such as RSSI value [20]. Some protocols, which need to queue packets, can allocate a queue buffer to store waiting packets such as the MAC protocol that cannot send packets until the wireless channel becomes free.

3.3.1 Without Fragmentation

In this subsection, we present the congestion analysis results without fragmentation where every single IPv6 packet is sent in one 802.15.4 frame without fragmentation. The network sizes are set to be 15, 25 and 50 nodes as shown in Fig. 3.10 and the offered loads (packet/second) are 8/1, 4/1, 2/1, 1/1, 1/2, 1/4, 1/8, 1/16, 1/32, and 1/64.

First, we set the MAC buffer size to 8 packets (8 * 127 bytes) which is the default setting of Contiki OS. Figure 3.11 shows the packet loss in 15-node, 25-node and 50-node networks respectively. Clearly, as offered load and number of nodes increase, the packet loss rises in the network. For example, with an offered load of one packet every second, the packet loss increases from 37 to 76% as the number of nodes in the network increases from 15 to 50. Figure 3.12 shows the number of lost packets, which are measured at the MAC layer, due to buffer drops and channel loss. In this figure, a logarithmic scale is used due to the big difference between packet buffer drops and packet channel loss. It is clear that with high offered load, the number of packets which are lost at sensor nodes (due to buffer overflow) is much higher than packet loss in the wireless channel. For instance, when the offered load is 8 packets

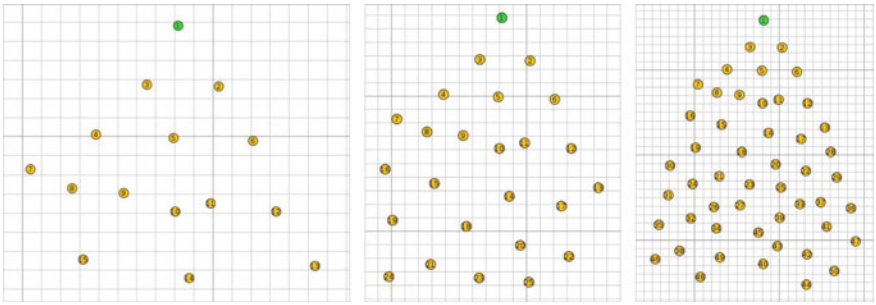

Fig. 3.10 Network 1, 2 and 3 topologies

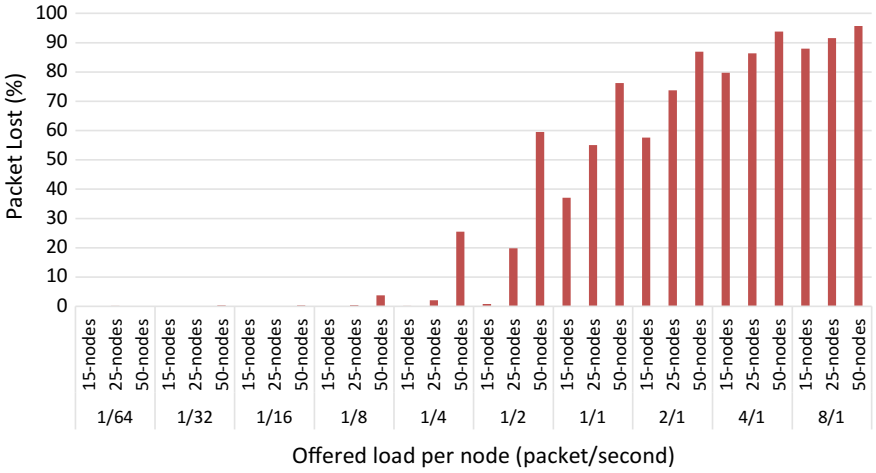

Fig. 3.11 Packet loss [buffer size = 8 packets]

per second and network size of 50 nodes, the total number of lost packets are up to 600,000 due to buffer overflow compared with only 2,000 due to channel loss. However, with low offered load, the number of lost packets in the wireless channel is slightly higher than due to buffer drops, e.g. with network traffic of 1 packet per 16 s and 50-node network, the packet loss due to buffer overflow and channel loss are 3 and 18 respectively. From Fig. 3.12, the number of dropped packets at the buffer increases as the offered load and number of nodes in the network increase. We can see that with high traffic, the majority of packets are lost at the buffer, i.e. more than 90% of the total lost packets are lost due to buffer overflow.

Second, we increase the buffer size to 16 packets to see the impact of buffer size on packet loss at the buffer. Figure 3.13 shows packet loss while varying network size and offered load. By comparing Fig. 3.11 with Fig. 3.13, it can be seen that by doubling the buffer size, the packet loss decreases with different offered loads. Similarly, the number of dropped packets at the buffer decreases by a small amount as can be seen by comparing Figs. 3.12 and 3.14. With the eight-packet buffer size

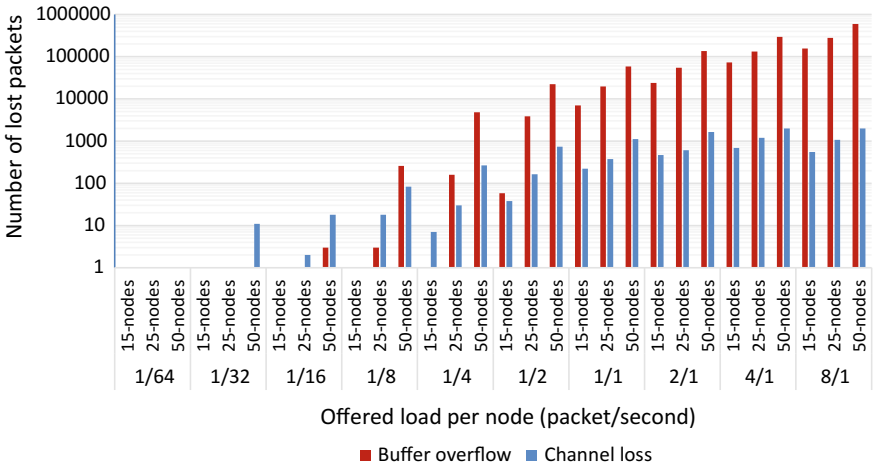

Fig. 3.12 Number of lost packets due to buffer overflow and wireless channel loss [buffer size = 8 packets]

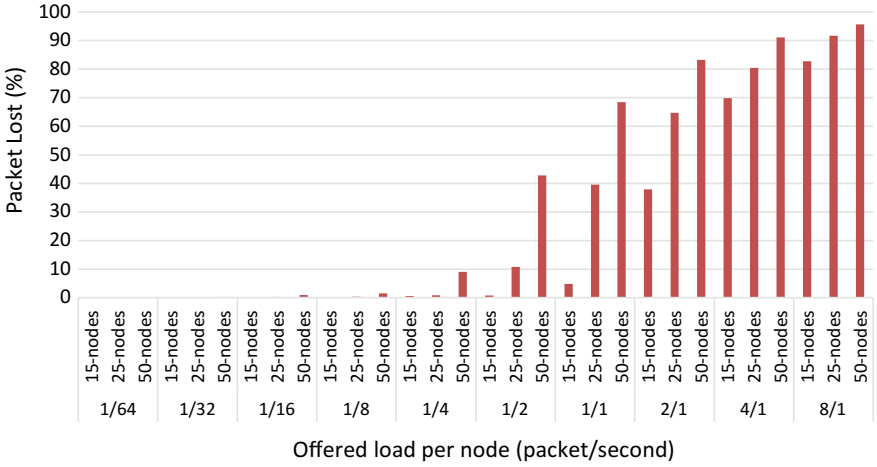

Fig. 3.13 Packet loss [buffer size = 16 packets]

scenario, more than 90% of packet loss occurs at the buffer with offered load 8/1, 4/1, 2/1 and 1/1. However, even when the buffer size is increased, the number of lost packets due to buffer overflow is still dominant as compared with channel loss.

Finally, we increase the node density with an average number of nodes per POS to 12, as node density has an impact on contention among nodes to access the wireless channel. We simulate a new network with 50 nodes and POS of 12 and compare it with the 50-node network which has POS of 4. From Fig. 3.15, we can see that packet loss increases with high POS. Also, the number of lost packets in both the buffer and the wireless channel increases as shown in Fig. 3.16.

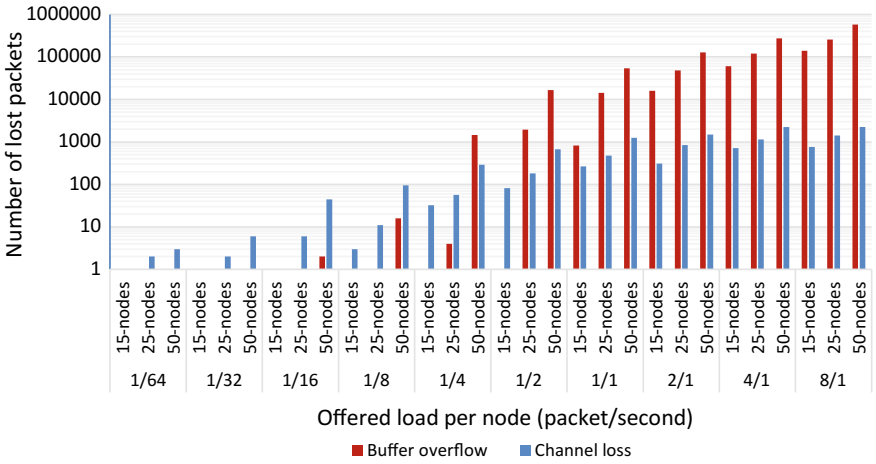

Fig. 3.14 Number of lost packets due to buffer overflow and wireless channel loss [buffer size = 16 packets]

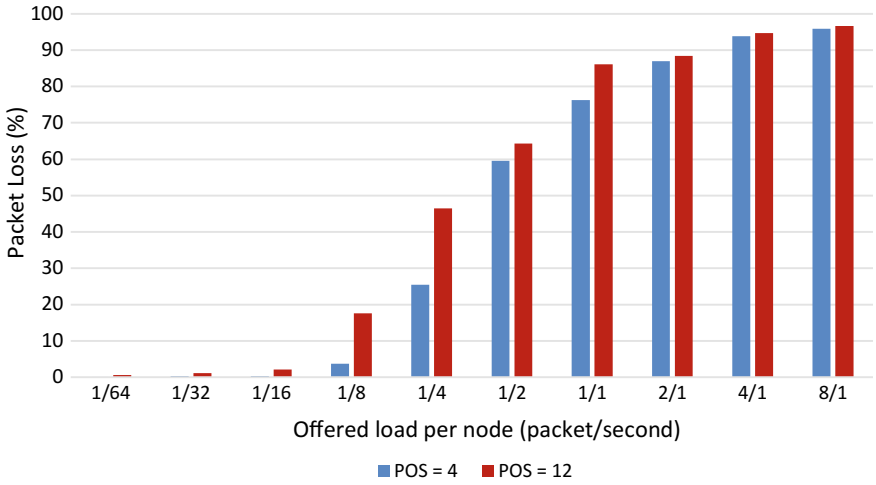

Fig. 3.15 Packet loss with varying POS

3.3.2 With Fragmentation

Next, in order to see the impact of increasing the application payload length (i.e. every IPv6 packet is fragmented into two or more fragments where each fragment is sent over a single IEEE 802.15.4 frame) on network performance within congestion, the application payload length is increased. In these experiments, 5-node, 15-node

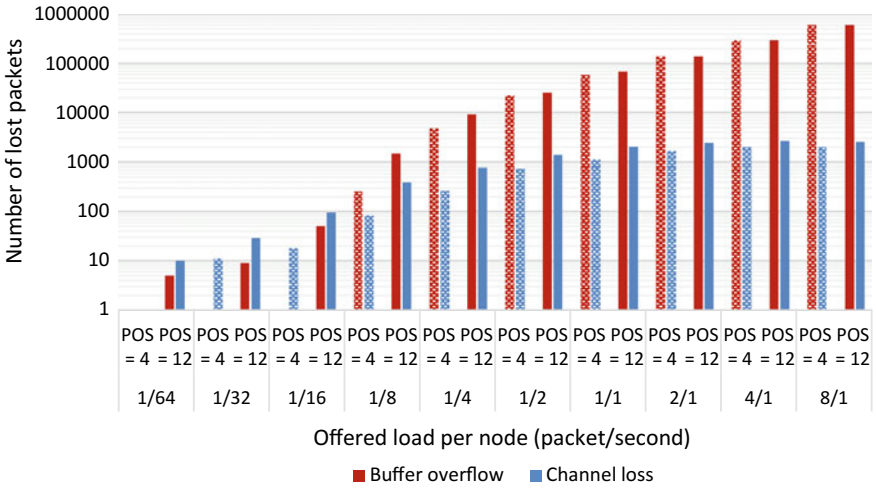

Fig. 3.16 Number of lost packets due to buffer overflow and wireless channel loss with varying POS

and 25-node networks as well as 8/1, 4/1, 2/1, 1/1, 1/2, 1/4, 1/8, 1/16, 1/32, and 1/64 offered loads (packet/second) are used.

Similarly to, without fragmentation with high offered load, the majority of packets are lost due to buffer overflow. On the other hand, with low traffic load, the channel loss packets are small and slightly higher than the buffer dropped packets. However, there is an important parameter that should be considered within fragmentation called 'reassembly timeout'. When a node receives an initial fragment, it stores the fragment in a buffer called the reassembly buffer and starts the reassembly timer countdown. When the reassembly timeout expires and the node has not received all fragments that belong to the same IPv6 packet, the received fragments are discarded. According to the standard protocol RFC 4944 [19], if the reassembly timer does not expire and a new fragment, which does not belong to the same packet that is being reassembled, is received, this fragment is dropped. However, with SICSlowpan, which is the 6LoWPAN implementation in Contiki, the previous packet fragments are discarded and the received new fragment is stored in the buffer to start reassembling a new packet. According to the standard protocol, the reassembly timeout must be set to a maximum value of 60 s. To notice the impact of the reassembly timeout parameter on network performance, we have used different values (0.001, 0.005, 0.05, 0.5, 1, 10, 30 and 60 s) in the experiments.

In 6LoWPAN, the routing schemes can be divided into two categories: 'route over' and 'mesh under' [21]. With route over, the routing decision is made at the routing layer and the fragmentation and reassembly process is implemented at each node through path from source to destination. In mesh under, the routing decision is taken at SICSlowpan layer (adaptation layer) as well as fragmentation and reassembly being executed at source and destination nodes only. Contiki OS supports the 'route

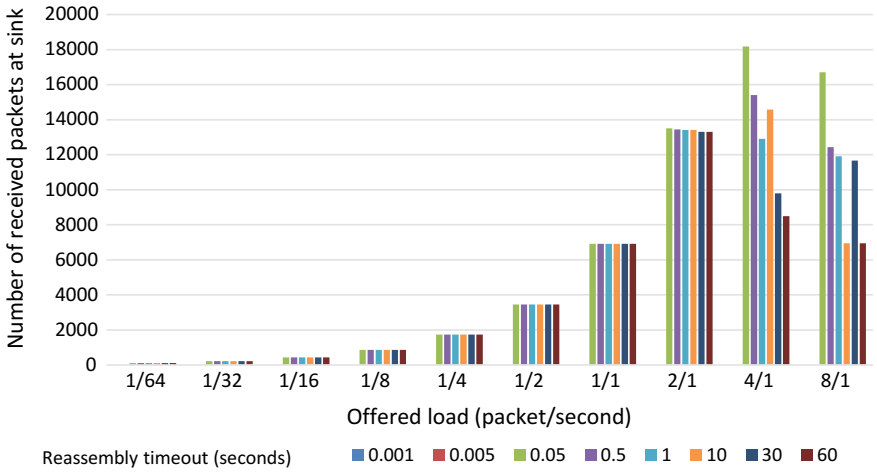

Fig. 3.17 Number of received packets (when broken into two fragments) at sink in 5-node network

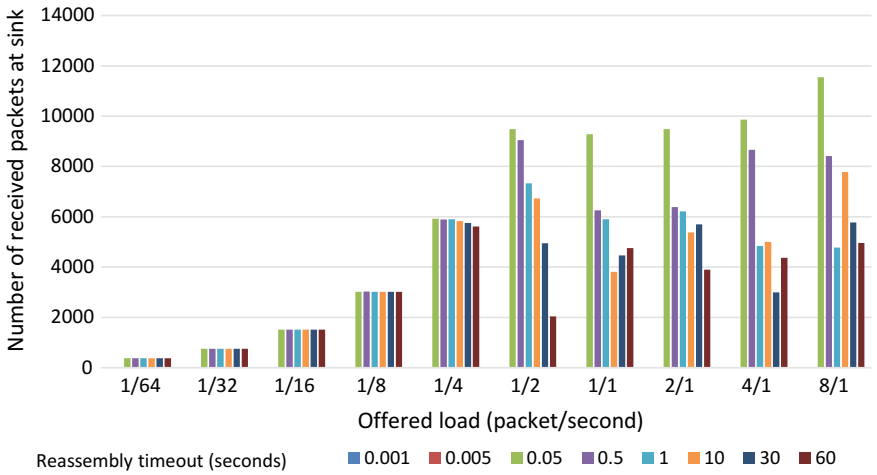

Fig. 3.18 Number of received packets (when broken into two fragments) at sink in 15-node network

over' scheme, which is used in these experiments, where RPL performs the routing decision.

First, the application payload length is set to 100 bytes since every IPv6 packet is divided into two fragments. Figures 3.17, 3.18 and 3.19 show the number of received packet at the sink node with different offered loads and various reassembly timeouts in 5-node, 15-node and 25-node networks respectively. From these figures, it can be seen that with reassembly timeout values of 1 and 5 ms, the number of received packets at the sink is zero since the reassembly timeout is too short and it expires early. In this case, a node drops the first fragment before the second fragment arrives.

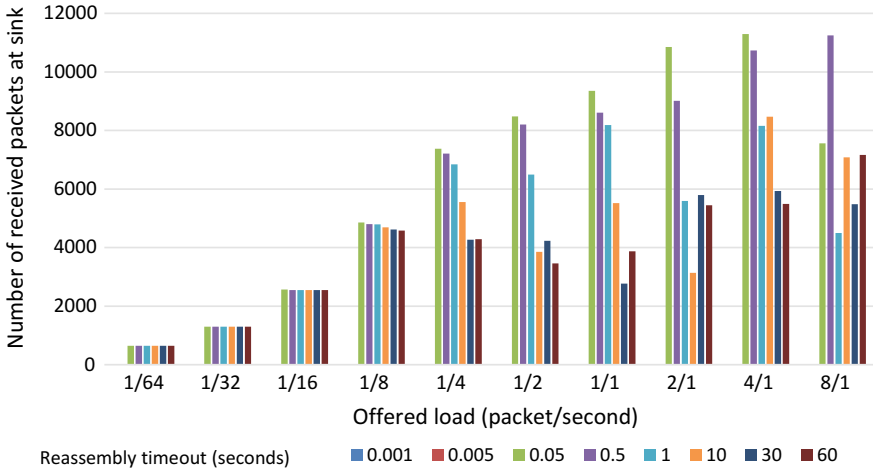

Fig. 3.19 Number of received packets (when broken into two fragments) at sink in 25-node network

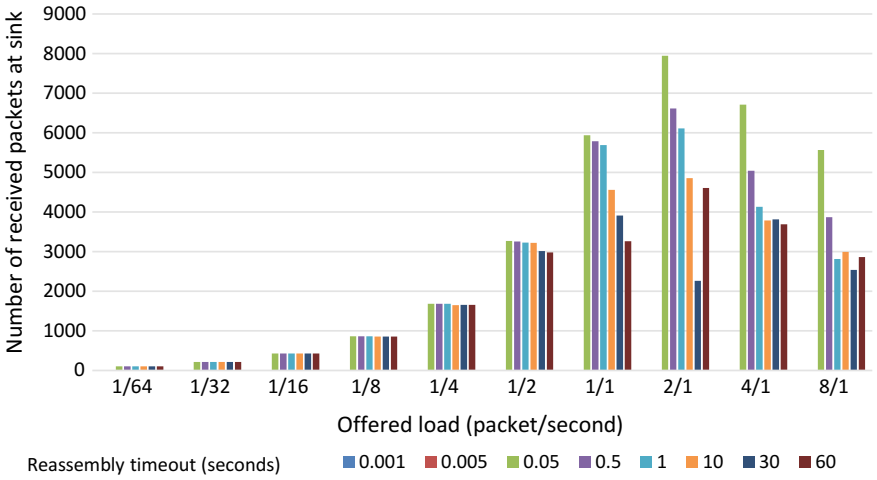

Fig. 3.20 Number of received packets (when broken into four fragments) at sink in 5-node network

It is clear that with low data rate, the reassembly timeout has no or a tiny impact on the number of received packets at the sink. On the other hand, with high network traffic where congestion occurs, the reassembly timeout value of 0.05 s has the highest number of received packets as compared to others except for the case of an offered load of 8 packets per second with a 25-node network, when 0.5 s is the best in terms of the number of received packets.

Meanwhile, in order to determine the impact of the number of fragments on the reassembly timeout parameter, we increase the application payload length to 300 bytes where every IPv6 packet is fragmented into four fragments. Figures 3.20,

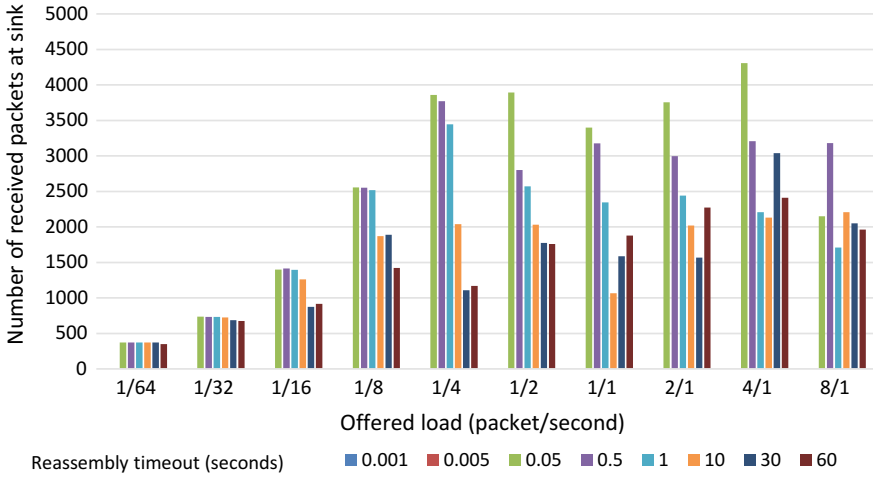

Fig. 3.21 Number of received packets (when broken into four fragments) at sink in 15-node network

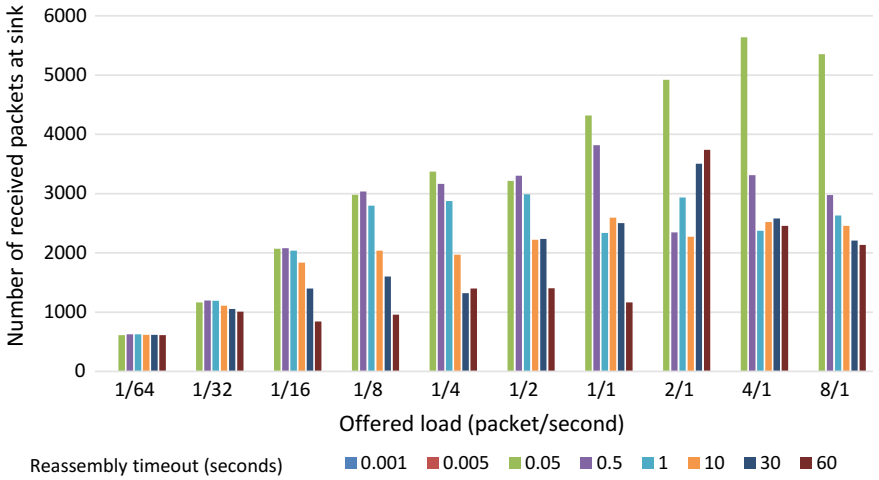

Fig. 3.22 Number of received packets (when broken into four fragments) at sink in 25-node network

3.21 and 3.22 show the number of received packets at the sink in 5-node network, 15-node network and 25-node network respectively. As in the case of two fragments, the number of received packets is zero for 1 and 5 ms reassembly timeouts. Also, we can see that with low offered load; the reassembly timeout has a little impact on the number of received packets. However, with high offered load, a reassembly timeout of 0.05 s has better performance in term of packet delivery ratio than others in all scenarios except in a scenario of 8 packets per second with 15-node network where 0.5 reassembly timeout is best.

3.3.3 Discussion

First, in the scenarios without fragmentation, the simulation results show that with different: network size, offered load, buffer size and node density, the majority of packets across the network are lost at the sensor node due to buffer drops as compared to channel loss when congestion occurs in 6LoWPAN networks. Also, as expected with high offered loads (i.e. 1/1, 2/1, 4/1 and 8/1 packet/s), the number of lost packets in the wireless channel remains approximately constant whereas, the number of lost packets at node buffers increases as the offered load increases as shown in Figs. 3.12, 3.14 and 3.16. It should be stressed that the network is designed to operate in low congestion conditions. This means that during normal operation packet loss across the network will predominantly be due to channel loss. When congestion does occur due to periods of high traffic, buffer overflow loss at nodes will predominate. This occurs because all neighbouring nodes are forwarding packets to their parent without checking buffer occupancy. In addition, all layers above the MAC layer send packets to the MAC layer without checking the available MAC's buffer space. On the other hand, the MAC layer cannot send the packets directly to the wireless channel without checking its availability.

However, in the scenarios considering fragmentation, it is obvious that the value of the reassembly timeout parameter has a significant effect on network performance when congestion occurs while no impact has been noticed with low offered load. Also, we can see that as the number of nodes in the network and number of fragments increase, the reassembly timeout has more effect on network performance. For example with two fragments and the 5-node network scenario, the offered loads 1/1, 1/2, 1/4, 1/8, 1/16, 1/32 and 1/64 have the same number of received packets with different reassembly timeouts whereas, in 25-node network with four fragments, the only offered load 1/64 has the same received packets with different values of reassembly timeout. Generally, it is clear that with low data rate; as the probability of packet loss (fragments) is low, the reassembly timeout parameter has no impact on network performance. It is known that with low data rate, high competition among nodes on the wireless channel does not exist as well as buffer overflow does not occur. Thus, when an intermediate node receives the first fragment of an IPv6 packet, it would successfully receive the next incoming fragments shortly afterwards. Also, sending the next IPv6 packet would happen after a long time (e.g. one minute or more). In this case, the value of reassembly timeout is inconsequential (e.g. 10, 20, 30 s etc.) since the reassembly timer expires for the current assembled IPv6 packet before receiving the next IPv6 packet. On the other hand, for high data rate, the reassembly timeout value has an effect since the probability of packet loss is high and the next IPv6 packet will arrive after a very short time from receiving the current packet. Therefore, it is important that the value of reassembly timeout should be short (e.g. 50 ms) for high data rate. In [22], Teo et al. propose a new reassembly mechanism called Multi-Reassemblies Buffer Management System (MR-BMS) for 6LoWPAN networks. MR-BMS consists of three components: buffer manager, list of reassembly buffer and IP packet buffer. When a new fragment arrives at a node, the buffer

manager creates a new reassembly buffer to store the incoming fragment and starts a reassembly timer. After that, if the next incoming fragment belongs to the packet of the first fragment, it is stored in the same buffer. Otherwise, a new reassembly buffer is created to store the incoming fragment. However, the authors do not show the importance of the reassembly timeout parameter value in heavy traffic conditions.

In conclusion, it is clear that the majority of packets are lost due to buffer overflow when there is congestion. Therefore, buffer occupancy should be considered in protocol designs such as RPL. This will tackle the issue of congestion by reducing buffer overflow and improving network performance. Also, when the application payload size is increased, since IPv6 packets are divided into two or more fragments, the reassembly timeout parameter needs careful consideration. The reassembly timeout value should be small during periods of congestion or if possible be adaptive according to network conditions.

3.4 Testbed-Based Congestion Analysis for 6LoWPAN

Here, we investigate and evaluate the impact of congestion on 6LoWPAN networks through a testbed using 10 CM5000 TelosB sensor nodes. The testbed experiments are carried out on different scenarios (indoor and outdoor) and various parameters (varying buffer size and with or without packet fragmentation). The distribution of the sensor nodes is shown in Fig. 3.23. For indoor experiments, the nodes are distributed in a house of three floors with dimensions of 3.70 m × 2.60 m × 8 m (length × height × width). The sink node, intermediate nodes and leaf nodes are placed in the basement, ground floor and first floor respectively. For outdoor tests, the nodes were distributed in an open space on Kirkstall Road, Leeds city where the node distribution is shown in Fig. 3.23. Finally, the simulation tests are implemented in Contiki OS with Cooja simulator. The CM5000 TelosB sensors is IEEE 802.15.4 compliant wireless sensor node based on the original open-source TelosB/Tmote Sky platform design

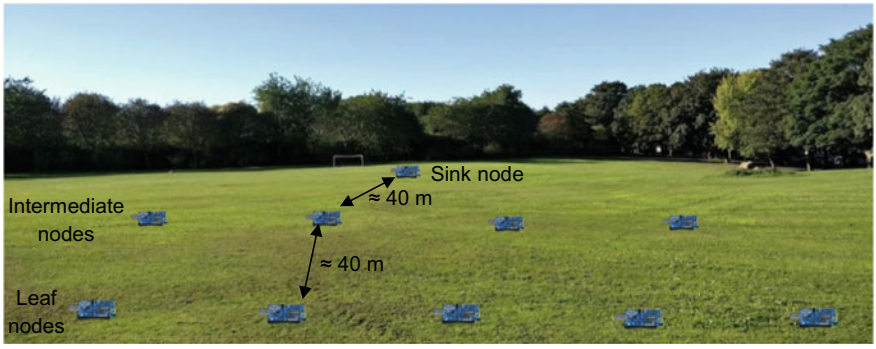

Fig. 3.23 Real sensor nodes distribution in testbed experiments

developed and published by the University of California, Berkeley [23]. The main technical specifications of CM5000 TelosB sensor node are as follows:

- Processor model: Texas Instruments® MSP430F1611.
- Memory: 48 kB (Program flash), 10 kB (Data RAM) and 1 MB (External Flash).
- RF chip: Texas Instruments® CC2420 (IEEE 802.15.4 2.4 GHz Wireless Module).
- Frequency band: 2.4–2.485 GHz.
- Transfer rate: 250 kbps.
- Range: ∼120 m (outdoor) and 20–30 m (indoor).

3.4.1 Without Fragmentation

In this subsection, we present the congestion analysis results without fragmentation where every single IPv6 packet sends over a one IEEE 802.15.4 frame without fragmentation. The offered loads (packet/second) are set to be 8/1, 4/1, 2/1, 1/1, 1/2, 1/4, 1/8, 1/16, 1/32, and 1/64.

First, the MAC buffer size is set to 8 packets (8 * 127 bytes) which is the default setting of Contiki OS. Figure 3.24 illustrates the percentage of packets lost in the network for the three scenarios, i.e. indoor, outdoor and simulation tests. As expected, the figure shows an upward trend. As the data rate per node increases, the percentage of packet loss increases. The graph can be divided into two categories: low offered loads from 1/64 to 1/2 packets per second and high offered loads from 1 to 8 packets per second. For the first category, the percentage of packet loss is less than 5%. From this point onwards, the percentage increases significantly reaching its highest point of 91% when an offered load of 8 packets/s in the indoor environment.

Also, Fig. 3.24 shows good correlation between indoor, outdoor and simulation results. In the indoor experiments, there is the highest loss because of the high impact of the environment on the radio signals, e.g. interferences with other signal sources (e.g. Wi-Fi) and signal attenuation and reflection due to walls and ceilings. The largest difference between hardware tests and simulation is at the offered load of one packet per second. Simulation results present 10% of packet loss whereas the outdoor tests present around 20%. However, when increasing the offered load per node, the hardware tests results are gradually similar to simulations.

Next, the buffer size is increased to 16 packets to see the impact of buffer size on packet loss in the network. Figure 3.25 shows the percentage of packet loss for indoor, outdoor and simulation tests. Again the results show similar trends among indoor, outdoor and simulation results but with slightly different values. By comparing Fig. 3.24 with Fig. 3.25, it can be seen that by doubling the buffer size, the packet loss decreases with different offered loads. For instance, when the data rate is 8 packets per second, the percentage of packet loss improved around 20% compared to the case when the buffer size is 8 packets.

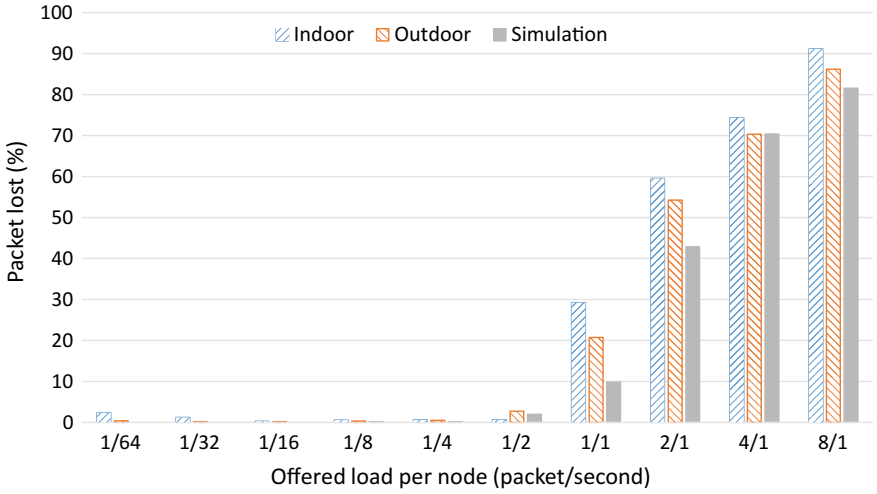

Fig. 3.24 Packet loss [buffer size = 8 packets]

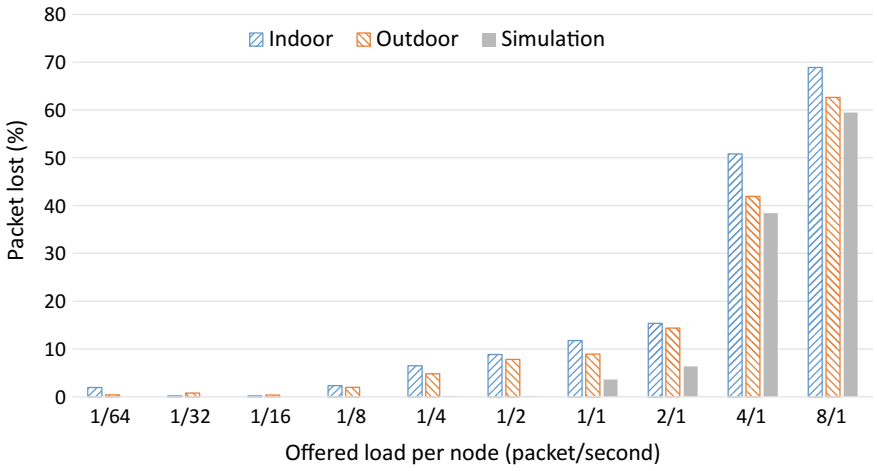

Fig. 3.25 Packet loss [buffer size = 16 packets]

3.4.2 With Fragmentation

Next, we investigate the impact of increasing the application payload length on the network performance with congestion. The application payload length is increased so that every IPv6 packet is fragmented into two or more fragments where each fragment is sent over a single IEEE 802.15.4 frame. In these experiments, offered

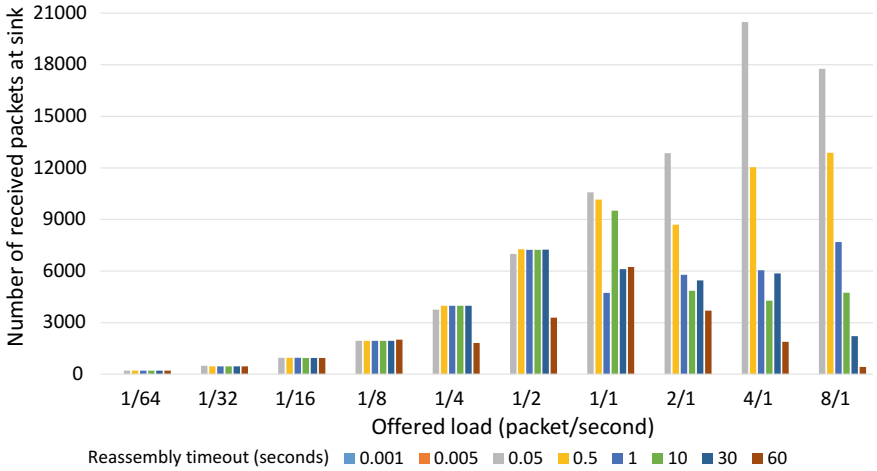

Fig. 3.26 Number of received packets (when broken into two fragments) for indoor test

loads of 8/1, 4/1, 2/1, 1/1, 1/2, 1/4, 1/8, 1/16, 1/32, and 1/64 (packet/second) are used. Similarly to without fragmentation, with high offered load the majority of packets are lost in the network due to congestion. However, there is an important parameter that should be considered within fragmentation called the 'reassembly timeout'. To notice the impact of this parameter on network performance, we have used different values (0.001, 0.005, 0.05, 0.5, 1, 10, 30 and 60 s) in the experiments.

First, the application payload length is set to 100 bytes which means every IPv6 packet is divided into two fragments. Figures 3.26, 3.27 and 3.28 show the number of received packets at the sink node for indoor, outdoor and simulation tests respectively. Results obtained for the three scenarios show that no packet is received by the sink node when the reassembly timeout is set to 1 and 5 ms. This means that this time is too short to wait for the next fragment of a packet at the adaptation layer. Therefore, all fragments are dropped as they cannot be reassembled into a single IPv6 packet. It is clear that with low data rate, the reassembly timeout has no impact or only a marginal impact on the number of received packets at the sink. On the other hand, with high network traffic where congestion occurs, the reassembly timeout value of 0.05 s has the highest number of received packets as compared to others. Testbed outdoor results shown in Fig. 3.27 perform similarly to simulation results shown in Fig. 3.28. In contrast, testbed indoor results shown in Fig. 3.26 perform similar trend but slightly different values from simulation and outdoor results for reasons stated previously (in Sect. 3.4.1).

Next, in order to determine the impact of the number of fragments on the reassembly timeout parameter, we increase the application payload length to 300 bytes where

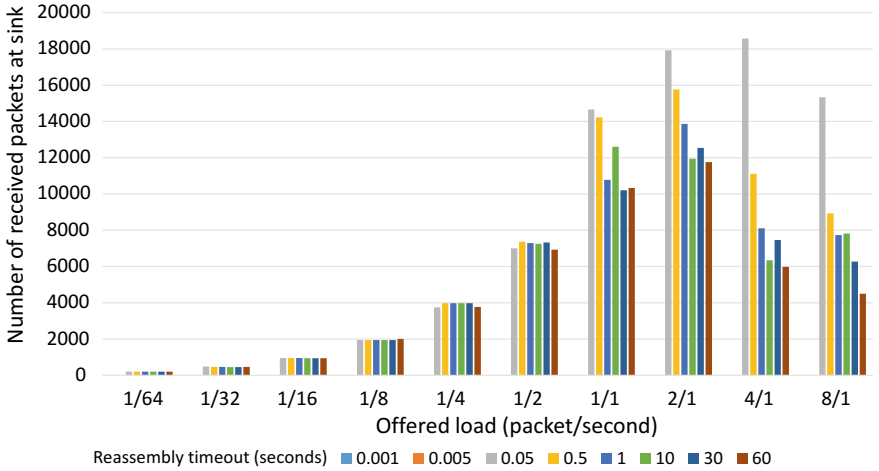

Fig. 3.27 Number of received packets (when broken into two fragments) for outdoor test

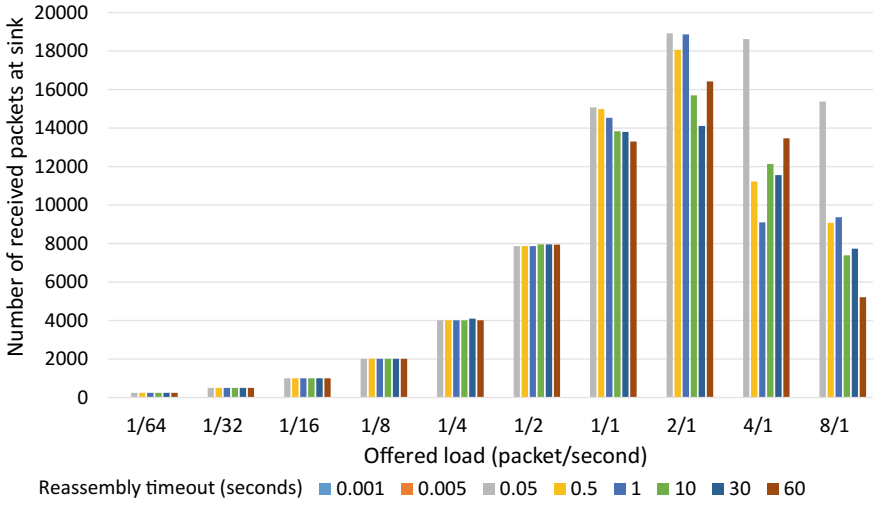

Fig. 3.28 Number of received packets (when broken into two fragments) for simulation test

every IPv6 packet is fragmented into four fragments. Figures 3.29, 3.30 and 3.31 show the number of received packets at the sink for indoor, outdoor and simulation test scenarios, respectively. As in the case of two fragments, the number of received packets is zero for 1 and 5 ms reassembly timeout values since the node drops the first fragment before the second fragment arrives. Also, we can see that with low offered load; the reassembly timeout has a little impact on the number of received

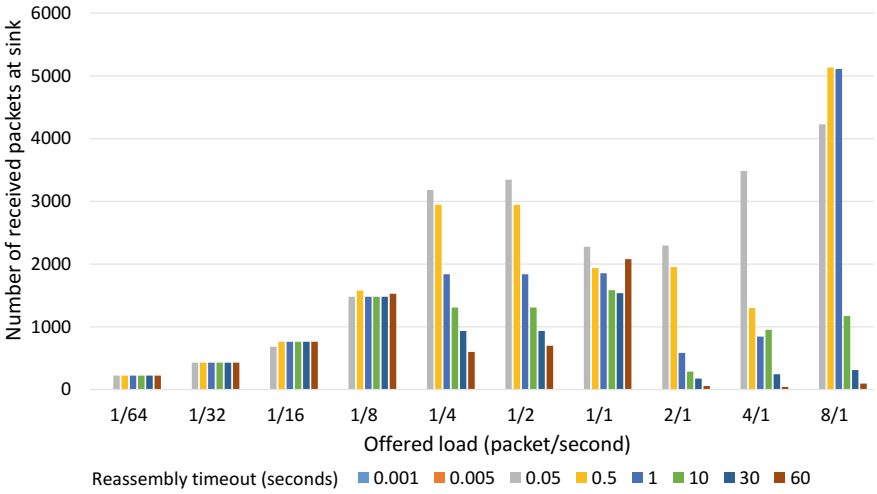

Fig. 3.29 Number of received packets (when broken into four fragments) for indoor

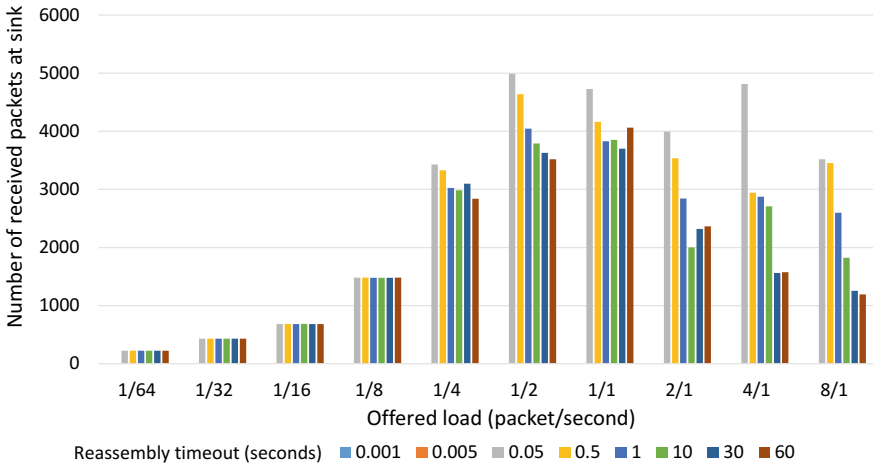

Fig. 3.30 Number of received packets (when broken into four fragments) for outdoor

packets. However, with high offered load, a reassembly timeout of 0.05 s has better performance in term of packet delivery ratio than others in all scenarios except in a scenario of 8 packets per second for simulation where 0.5 s reassembly timeout is best.

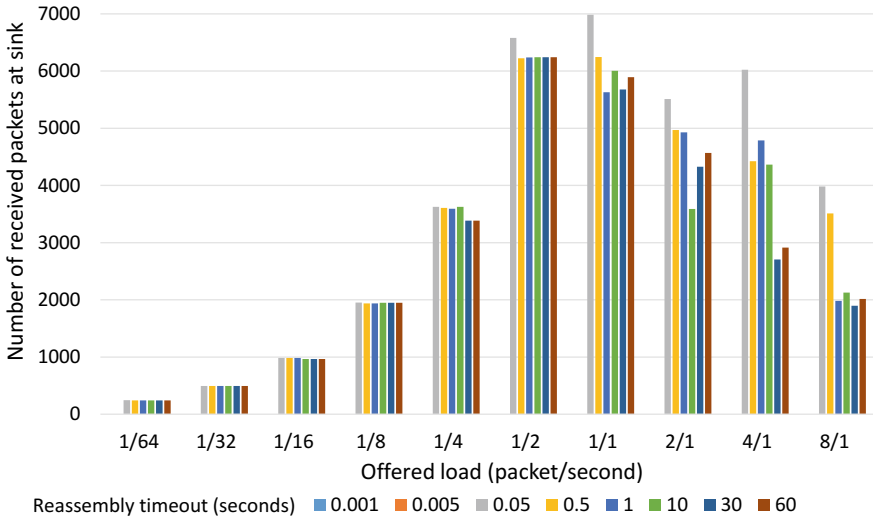

Fig. 3.31 Number of received packets (when broken into four fragments) for simulation

3.5 Conclusion

In this chapter, we have presented an analytical model for 6LoWPAN network in the presence of congestion using Markov chain analysis and queuing theory. We have derived the expressions for the buffer loss probability and throughput at the sink. Also, we have calculated the IEEE 802.15.4 actual channel capacity under an unslotted CSMA-CA with and without collisions based on Contiki 3.0 implementation. Simulation results show a good match with the analytical modelling of congestion for different scenarios and various parameters. Also, simulation results show that: (i) As the number of leaf nodes increases, buffer overflow increases in the network, (ii) As buffer size is increased, buffer overflow at the leaf node decreases while it increases at the intermediate node, and (iii) As offered load is increased, the number of dropped packets increases in the leaf and intermediate nodes.

Also, in this chapter, we have presented a comprehensive analysis of the impact of congestion in 6LoWPAN networks with Contiki OS and Cooja simulator with different parameters: network size, network traffic load, buffer size, node density, and number of fragments (application payload size). Simulation results show that when congestion does occur, the majority of packets are lost at sensor nodes due to buffer overflow. Also, the reassembly timeout parameter value has an impact on network performance when IPv6 packets are fragmented at the adaptation layer. In order to improve network performance, the buffer occupancy should be considered in different protocol designs such as an objective function of RPL which is responsible for network topology construction.

Finally, real experiments have been implemented using 10, CM5000 TelosB sensor nodes in a network to evaluate the impact of congestion on 6LoWPAN networks in a real environment. The real experiments are carried out with different scenarios (indoor and outdoor) and various parameters. The results show similar performance between simulation and outdoor experiments. However, indoor experimental results have slightly different values but with the same trend to simulation and outdoor experiments since parameters such as nodes position, enclosed environment and interfering wireless signals (e.g. Wi-Fi) influenced negatively.

References

1. Ghaffari A (2015) Congestion control mechanisms in wireless sensor networks: a survey. J Netw Comput Appl 52:101–115
2. Michopoulos V, Guan L, Oikonomou G, Phillips I (2011) A comparative study of congestion control algorithms in IPv6 wireless sensor networks. In: Proceedings of international conference on distributed computing in sensor systems and workshops (DCOSS). IEEE, pp 1–6
3. Michopoulos V, Guan L, Oikonomou G, Phillips I (2012) DCCC6: duty cycle-aware congestion control for 6LoWPAN networks. In: Proceedings of international conference on pervasive computing and communications workshops (PERCOM workshops). IEEE, pp 278–283
4. Castellani AP, Rossi M, Zorzi M (2014) Back pressure congestion control for CoAP/6LoWPAN networks. Ad Hoc Netw 18:71–84
5. Hellaoui H, Koudil M (2015) Bird flocking congestion control for CoAP/RPL/6LoWPAN networks. In: Proceedings of the workshop on IoT challenges in mobile and industrial systems. ACM, pp 25–30
6. Hull B, Jamieson K, Balakrishnan H (2004) Mitigating congestion in wireless sensor networks. In: Proceedings of the 2nd international conference on embedded networked sensor systems. ACM, pp 134–147
7. Gebali F (2015) Analysis of computer networks. Springer, Berlin
8. Keshav S (2012) Mathematical foundations of computer networking. Addison-Wesley, Upper Saddle River
9. Dunkels A, Grönvall B, Voigt T (2004) Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of 29th annual IEEE international conference on local computer networks. IEEE, pp 455–462
10. Osterlind F, Dunkels A, Eriksson J, Finne N, Voigt T (2006) Cross-level sensor network simulation with COOJA. In: Proceedings of 31st IEEE conference on local computer networks. IEEE, pp 641–648
11. Winter T, Thubert P, Brandt A, Hui J, Kelsey R (2012) RPL: IPv6 routing protocol for low-power and lossy networks. IETF, RFC 6550
12. Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). IEEE Standard 802.15.4 (2003)
13. Di Marco P, Park P, Fischione C, Johansson KH (2012) Analytical modeling of multi-hop IEEE 802.15. 4 networks. IEEE Trans Veh Technol 61(7):3191–3208
14. Sun T, Chen L-J, Han C-C, Yang G, Gerla M (2006) Measuring effective capacity of IEEE 802.15. 4 beaconless mode. In: Proceedings of wireless communications and networking conference (WCNC), vol 1. IEEE, pp 493–498
15. Sun T, Chen L-J, Yang G, Sanadidi M, Gerla M (2005) SenProbe: path capacity estimation in wireless sensor networks. In: Proceedings of the wireless traffic measurements and modeling workshop (SenMetrics). Citeseer
16. Latré B, Mil PD, Moerman I, Dhoedt B, Demeester P, Dierdonck NV (2006) Throughput and delay analysis of unslotted IEEE 802.15. 4. J Netw 1(1):20–28

17. Yang S-H (2014) Wireless sensor networks: principles, design and applications. Springer, London
18. Hui J, Thubert P (2011) Compression format for IPv6 datagrams over IEEE 802.15. 4-based networks. IETF RFC 6282
19. Montenegro G, Kushalnagar N, Hui J, Culler D (2007) Transmission of IPv6 packets over IEEE 802.15.4 networks. IETF RFC 4944
20. Dunkels A, Österlind F, He Z (2007) An adaptive communication architecture for wireless sensor networks. In: Proceedings of the 5th international conference on embedded networked sensor systems. ACM, pp 335–349
21. Al-Nidawi Y, Salman N, Kemp AH (2014) Mesh-under cluster-based routing protocol for IEEE 802.15.4 sensor network. In: Proceedings of 20th European wireless conference. VDE, pp 1–7
22. Teo KH, Abdullah A, Subramaniam SK, Sinniah GR (2013) New reassembly buffer management system in 6LoWPAN. Proc Asia-Pac Adv Netw 36:57–64
23. ADVANTICSYS. MTM-CM5000-MSP. https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html

Chapter 4
Congestion-Aware Routing Protocol for 6LoWPANs

4.1 Introduction

It is known that existing protocols and the architecture of the Internet are inefficient for WSNs. Recently, the IETF has developed a set of IP-based protocols for 6LoWPAN networks through the 6LoWPAN and ROLL working groups [1]. One of the main protocols is RPL [2] which is expected to be the standard routing protocol for 6LoWPAN networks [3]. Many metrics have been proposed to be used with RPL that can be divided into link and node metrics, e.g. hop count, expected transmission count (ETX), node energy, latency, link quality and throughput [4].

In this chapter, we are addressing the congestion problem in 6LoWPAN networks and improving the network performance when congestion occurs. In Chap. 3, we reported experiments to assess and analyse network conditions with congestion. We have concluded that with high network traffic, the majority of packets are lost at node buffers due to buffer overflow. Therefore, it is important to take the buffer occupancy into account to reduce the number of dropped packets at the buffer. To the best of our knowledge, this is the first work that considers the buffer occupancy as a metric in the RPL objective function. This scheme can improve network performance by reducing the number of lost packets due to buffer overflow. Thus, in this chapter, we propose a new objective function called congestion-aware objective function (CA-OF) which combines two metrics: buffer occupancy (BO) and ETX. With the proposed objective function, packets are forwarded to a sink node through less congested nodes.

The remainder of the chapter is organized as follows: in Sect. 4.2, we provide a literature review of related work about the proposed objective functions in RPL. A new objective function with a new metric is proposed in Sect. 4.3. In Sect. 4.4, simulation scenarios and results are given. Finally, Sect. 4.5 concludes this chapter.

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_4

4.2 Objective Function Related Work

Many routing metrics and objective functions have been proposed to be used within the RPL routing protocol in LLNs and 6LoWPAN networks. In this section, a discussion and review of these routing metrics and objective functions are given.

In [5], the default objective function for RPL, called objective function zero (OF0), is developed. OF0 is designed to find the nearest path to the root node in terms of distance by using the hop count as a metric. In OF0, a node selects its preferred parent with minimum rank which is increased by a strictly positive normalized scalar called rank_increase to obtain the node's rank. OF0 does not use the link and node metrics which are defined in [4]. Thus, OF0 does not reflect the node and link conditions and characterizations such as when a high packet loss occurs in a wireless link or at a parent node.

In [6], ETX-OF is proposed where it is based on the ETX metric. ETX describes the expected number of transmissions to successfully transmit a packet on a link. ETX-OF finds a path which can deliver a packet from a node to the sink node with minimum number of transmissions. A node computes the ETX path value for a path through each candidate neighbour by adding two components: the ETX value of a link to a candidate neighbour and the ETX value of the path, which is advertised in a DIO message, from the selected neighbour to the sink. The node selects its preferred parent with minimum ETX path value. In other words, ETX-OF selects a path which has least packet channel loss. Thus, ETX-OF reflects how much the wireless link or channel is congested. However, it does not reflect how much a node is congested which is where the majority of packets are lost when congestion occurs.

In [7], a new RPL metric called averaged delay (AVG_DEL) has been proposed. This metric aims to minimize the delay from a node to the root node. AVG_DEL is computed as the cumulative sum of link-by-link delays along the path to the root node. The proposed metric has been compared with ETX in terms of delay over a 19-node network. In [8], a new objective function is developed based on remaining energy as a metric. The path cost from a node to the root node is defined as the minimum value between the preferred parent path cost and the node's energy. A node selects its parent that has maximum path cost value. The energy-based objective function is compared with ETX-OF within a 20-sensor network.

In [9], combined methods are proposed to quantify and combine one or multiple RPL routing metrics in an additive or lexical manner according to system user requirements. The routing metrics used in this work are hop count, ETX, link quality level (LQL) and nodes' available energy. The lexical combination manner provides a metrics prioritization to ensure application's requirements, while the additive method offers a flexible way to combine metrics using a metric weight pair. In [10], two RPL metrics are proposed based on the link performance at the MAC layer. The first metric called R-metric which includes ETX and packet losses due to the MAC contention. The second metric called Q-metric which provides a balanced load distribution in the network by selecting the lowest traffic loaded parent. The proposed metrics are

implemented and tested within a seven mote testbed network and compared with the back-pressure algorithm [11] which uses a weighted ETX cost.

In [12], a new RPL routing metric called PER-HOP ETX is proposed. The proposed metric distributes the ETX value to each node along a path from a node to the root node instead of using the additive ETX metric, as in [6]. The PER-HOP ETX metric works better when the network scale becomes large. The proposed metric is compared with OF0 and ETX-OF. In [4], the authors propose a set of link and node RPL routing metrics and constraints which are suitable to be used with 6LoWPAN. The proposed metrics are divided into node metrics and link metrics. The node metrics include node state and attribute (NSA), node energy (NE) and hop count (HP). The link metrics are throughput, latency, link reliability, which includes LQL and ETX, and link colour (LC). LC is a 10-bit value which indicates the link characteristics, e.g. whether the link supports encryption.

In [3], a new objective function called QoS-aware fuzzy logic (OF-FL) is developed based on the fuzzy logic concept. The proposed objective function combines a set of RPL metrics (end-to-end delay, hop count, ETX and battery level) to produce a single output metric called neighbour quality by using a fuzzy inference system which includes fuzzification, fuzzy rules and defuzzification. However, it is very difficult to implement the fuzzy logic system, which requires high computational processing capabilities, on a sensor node that has very limited processing resources.

Recently, in [13], a new RPL routing metric called DELAY_ROOT, which minimizes an average delay towards the DAG root, is proposed. The proposed metric can reduce the time delay between DODAG nodes and DODAG root based on the ContikiMAC radio duty cycle protocol. DELAY_ROOT is combined with three other metrics: ETX, rank and number of received packets to develop a new RPL-based routing protocol called congestion avoidance multipath routing protocol (CA-RPL). CA-RPL is tested and compared with RPL which uses ETX metric. Simulation results show that CA-RPL reduces the number of lost packets and the time delay from original RPL by an average value of 20 and 30%, respectively. However, the proposed routing protocol does not use the buffer occupancy as a metric where the majority of packets are lost when congestion does occur and it does not reflect how much the nodes are congested. Therefore, CA-RPL does not select less congested paths from nodes to the root but it selects a path with least time delay.

4.3 Congestion-Aware Objective Function Design

In RPL, the objective function, which is completely responsible for network topology construction, is separated from the core protocol specifications. This allows easy design and implementation of a new objective function that satisfies the application and network requirements. The objective function combines one or more RPL routing metrics to produce a rank value which is advertised by a DIO control message. The majority of packets are lost at node buffers when congestion occurs in 6LoWPAN networks. Hence, it is important to consider the node's buffer occupancy (measured

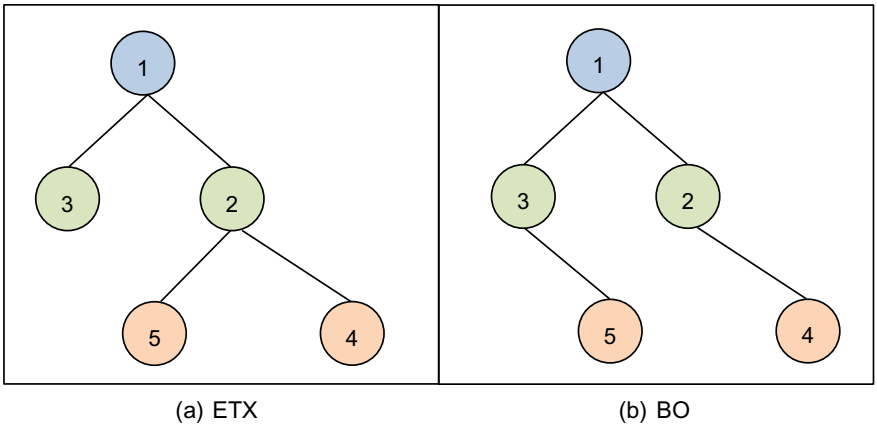

Fig. 4.1 Network topology within congestion

at MAC layer) as a metric in the objective function calculation in order to make the RPL routing protocol (network layer) aware of the dropped packets. Therefore, the objective function reflects how much the node is congested. Here, we are considering a cross-layer design where the MAC layer provides and exchanges buffer occupancy information with the network layer.

To explain the importance of the buffer occupancy metric in RPL when congestion occurs, consider the following simple scenario with a network of one sink node, two intermediate nodes and two leaf nodes. At the network topology construction stage, nodes 2 and 3 select the sink node (node 1) as parent and nodes 4 and 5 choose node 2 as parent as shown in Fig. 4.1a. First, the intermediate and leaf nodes send packets to the sink node with low data rate. In this case, with ETX metric, the packets are delivered successfully to the sink node and buffer overflow does not occur since the nodes' buffer is almost empty. After that, when an event occurs at the leaf nodes, they start to send packets at high data rate. In this situation, both nodes 4 and 5 send high data rate packets to their parent, node 2, where buffer overflow occurs. On the other hand, node 3 has no child node and its buffer is completely empty. With ETX-OF, which does not reflect the buffer overflow, nodes 4 and 5 continue to send their packets to node 2 where the majority of packets are lost at its buffer. Node 5 will not change its parent to node 3 where its buffer is empty since ETX-OF does not take the buffer occupancy into account. However, if the buffer occupancy is considered in the objective function as a metric, node 5 will change its parent to node 3, when high buffer overflow occurs at node 2, as the rank value of node 3 is smaller than the rank value of node 2 as shown in Fig. 4.1b. In this case, the network load is distributed between node 2 and 3, and hence buffer overflow is reduced significantly.

To develop a new objective function which works efficiently under low and high data conditions when congestion occurs. We must consider both the ETX metric which is important with low data rate (as the majority of packets are lost at the wireless channel) and the buffer occupancy (BO) metric (which is important to consider with

congestion occurrence where the majority of packets are lost due to buffer overflow). Thus, it is better to combine both metrics ETX and BO to develop a new objective function that works efficiently with different conditions. The modified objective function can be described as follows:

$$combined\_metric = w_{etx} * ETX + w_{bo} * BO, \tag{4.1}$$

where w_{etx} should be high with low data rate and w_{bo} should be high during periods of congestion, i.e. high traffic load. Hence, the buffer occupancy has been utilized as an indicator to realize the probability of buffer overflow. Thus, w_{etx} and w_{bo} are equal to buffer-free space and buffer occupancy, respectively. For instance, with low traffic where buffer is empty, w_{etx} becomes 100% while with high network traffic as the buffer is full, w_{bo} equals 100%. Therefore, the proposed objective function is aware of congestion and reflects how much the node and wireless channel are congested by using BO and ETX metrics, respectively.

4.4 Performance Evaluation

The proposed objective function has been tested and evaluated on three different network scenarios through simulation by using Contiki OS and Cooja simulator. The three networks were chosen to demonstrate performance on a small network of 10 nodes, a medium network of 19 nodes and a larger network of 35 nodes. CA-OF is compared with three objective functions: OF0, ETX-OF and ENERGY-OF. In all networks, we have used one sink node, a set of intermediate nodes which send packets to the sink node every minute and a group of leaf nodes which send packets at high data rate (4 packets/s) to create a congested situation. During the simulation, intermediate and leaf nodes start sending packets after 60 s as the network topology construction is completed. The protocol stack and simulation parameters used in the simulation are shown in Table 4.1.

In the first network, we used one sink node, six intermediate nodes and three leaf nodes where the sink node is located at the network edge. Figure 4.2 shows the total number of lost packets during the simulation time due to buffer overflow and channel loss for CA-OF, ENERGY-OF, ETX-OF and OF0. It is clear that CA-OF has less buffer overflow packets than others as CA-OF considers the buffer occupancy as a metric in its objective function and it tries to forward packets to the sink node through less congested nodes leading to reduced buffer overflow. Figure 4.3 shows network throughput which is measured as the number of successfully received packets at the sink node every minute. With CA-OF, the sink receives more packets than other objective functions, e.g. at simulation time of 20 min, the number of received packets at the sink is 685, 586, 398 and 355 for CA-OF, ENERGY-OF, ETX-OF and OF0, respectively.

Figure 4.4 compares packet delivery ratio (PDR) of CA-OF, ENERGY-OF, ETX-OF and OF0. It is obvious that as CA-OF forwards packets to the sink through

Table 4.1 Protocol stack

Layer	Protocol	Parameter value
Application	Every node periodically sends packet to sink node	Simulation time = 30 min for each simulation
Transport	UDP	
Network	uIPv6 + RPL	
Adaptation	SICSlowpan layer	Compression method = HC06
Data Link	CSMA (MAC layer) ContikiMAC (RDC layer) 802.15.4 (framer)	Buffer size = 8 packets CCA count = 2 MAC retransmission = 3 Channel check rate = 8 Hz
Physical	CC2420 RF transceiver	Max. packet length = 127 byte

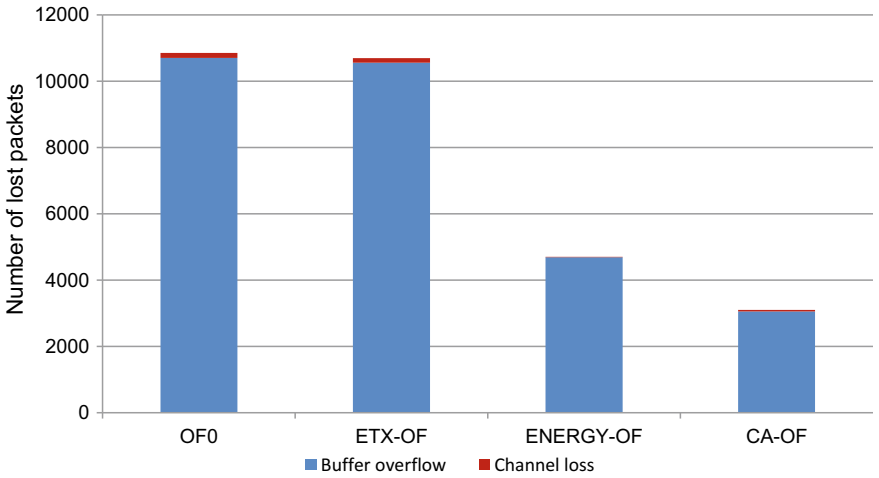

Fig. 4.2 Total number of lost packets in network 1

less congested nodes, it has better PDR than others with up to 85% compared with 77, 49 and 48% for ENERGY-OF, ETX-OF and OF0, respectively. Figure 4.5 illustrates the transmission and reception energy consumption in the intermediate and leaf nodes per successfully delivered packet to the sink node. From this figure, we can see that CA-OF consumes less energy than other objective functions. For instance, the intermediate and leaf nodes consume energy of 2.4 mJ in the reception for every successfully delivered packet to the sink with CA-OF compared with 2.77, 3.34 and 3.84 mJ in ENERGY-OF, ETX-OF and OF0, respectively. Also, from Fig. 4.5, we can see that energy consumption due to reception is higher than that due to transmission. The reason is that a node receives all signals from other nodes in the same transmission range at the physical layer. Then, the node passes the data to the MAC layer for checking. If the packet is addressed to that node, then pass it to the network layer. Otherwise, the node discards it.

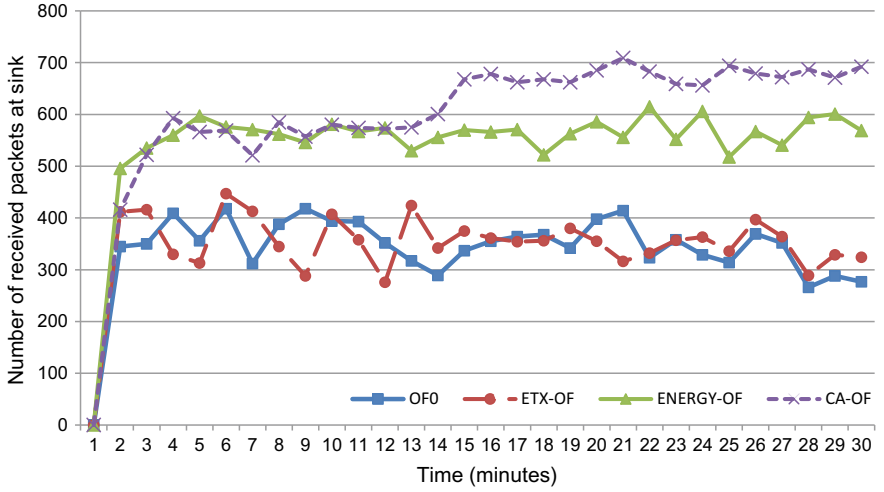

Fig. 4.3 Network 1 throughput

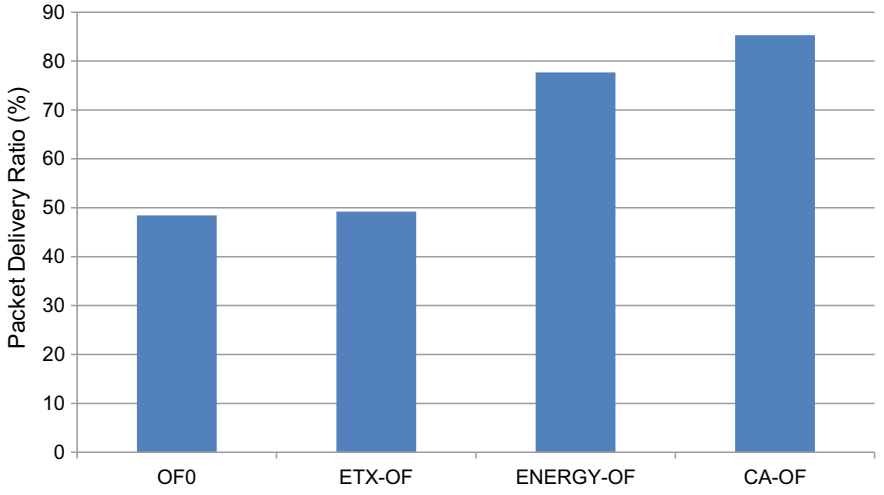

Fig. 4.4 Packet delivery Ratio in network 1

The second network consists of a sink node, 12 intermediate nodes and 6 leaf nodes. We have counted the number of buffer overflow packets and channel loss packets in the network for CA-OF and compared it with three other objective functions as shown in Fig. 4.6. This result shows that CA-OF loses less packets at the buffer than others as CA-OF considers the buffer occupancy as a metric in its objective function. Consequently, CA-OF tries to forward packets to the sink node through less congested nodes leading to reduced buffer overflow. Figure 4.7 shows network throughput which is measured as the number of successfully received packets at

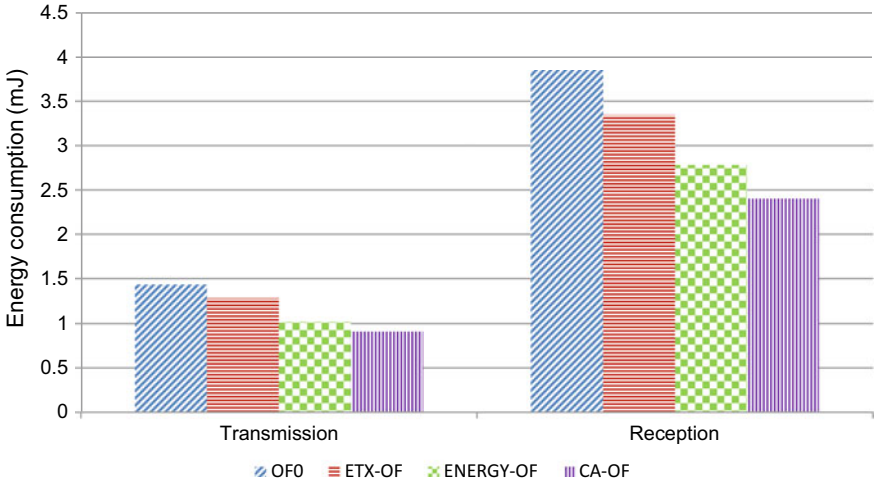

Fig. 4.5 Tx and Rx energy consumption per successful packet in network 1

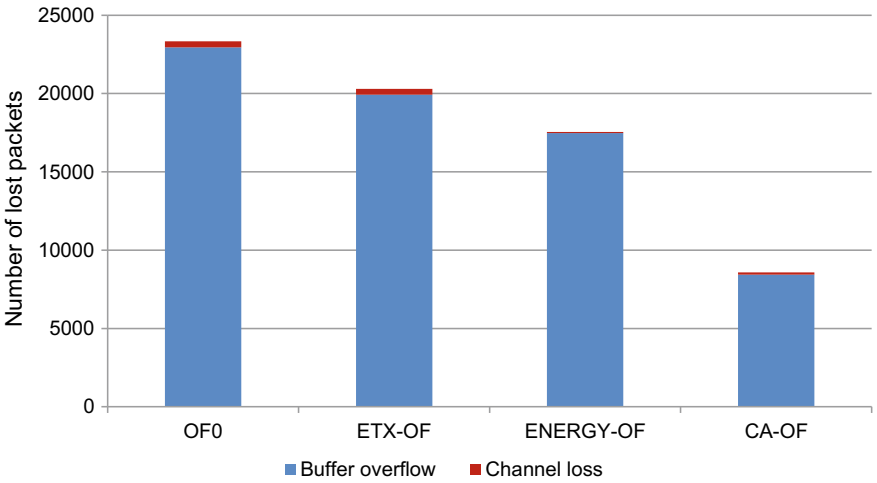

Fig. 4.6 Total number of lost packets in network 2

the sink node every minute. According to this figure, CA-OF has better network throughput than others for the reasons stated above. Figure 4.8 illustrates the ratio of the total number of received packets by the sink to the total number of sent packet in the intermediate and leaf nodes. It is clear that CA-OF has higher ratio with value of 79.57% than ENERGY-OF, ETX-OF and OF0 where they have a ratio of 58.19, 51.67 and 44.44%, respectively. During the simulation time, we have measured the total energy consumption due to transmission and reception at the intermediate and leaf nodes. Figure 4.9 shows the energy consumed in transmission and reception per

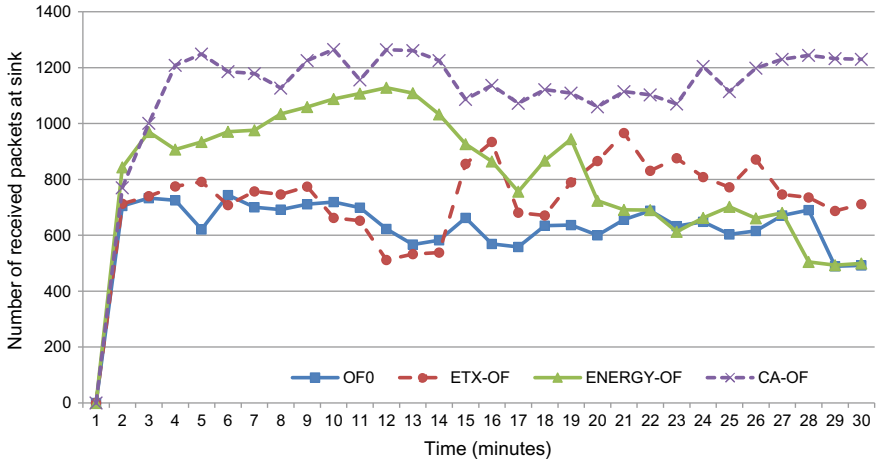

Fig. 4.7 Network 2 throughput

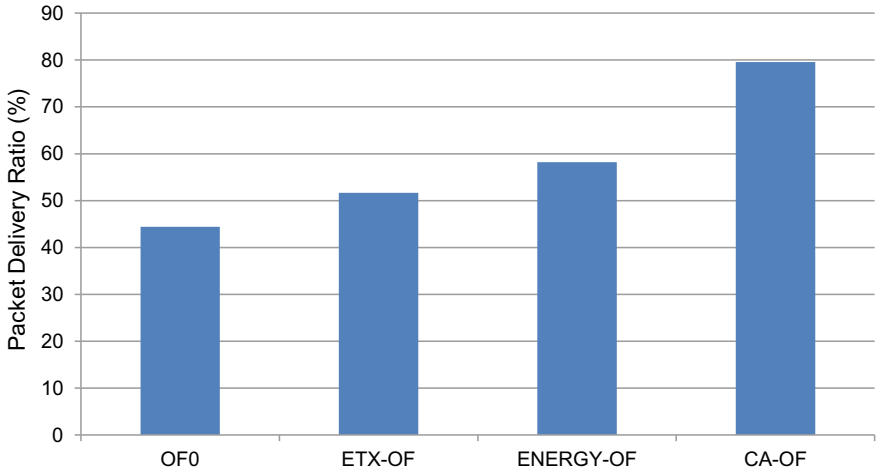

Fig. 4.8 Packet delivery ratio in network 2

successfully delivered packet. We note that with CA-OF, the energy consumption in the network is less than others as ENERGY-OF, ETX-OF and OF0 waste energy by transmitting and receiving packets which are then lost due to buffer overflow on the path without successful delivery because the buffer occupancy has not been considered.

Finally, we have tested CA-OF with a network of one sink, 24 intermediate nodes and 10 leaf nodes (this is formed network 3). Figure 4.10 shows the total number of lost packets in the network due to buffer overflow and channel loss. As the proposed objective function reflects how much the nodes are congested by using the buffer

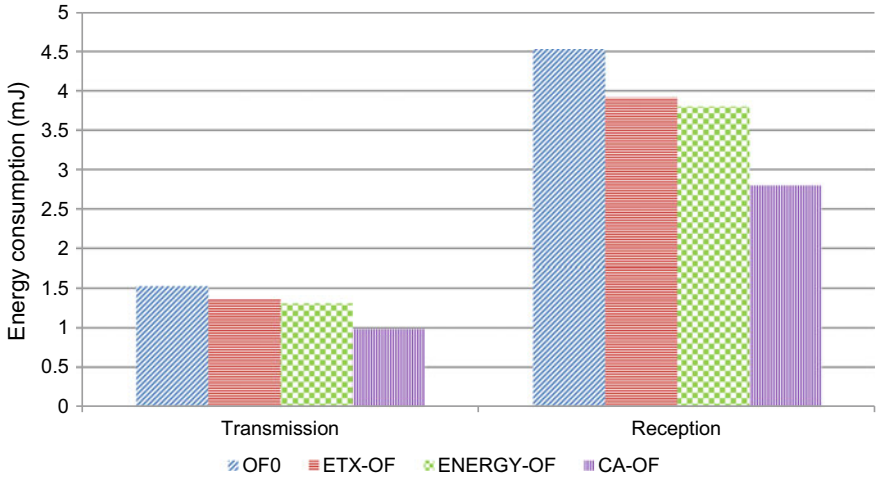

Fig. 4.9 Tx and Rx energy consumption per successful packet in network 2

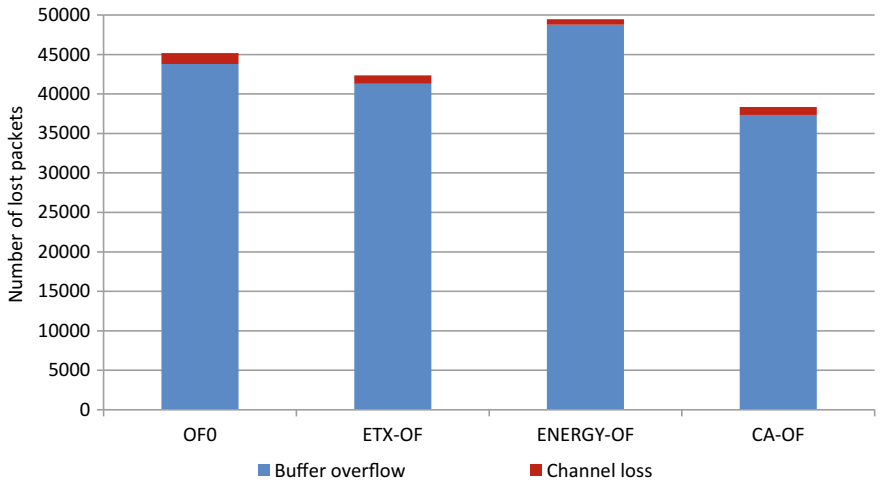

Fig. 4.10 Total number of lost packets in network 3

occupancy of these nodes as a metric, we can see that CA-OF saves more packets than others by reducing the number of dropped packets at the buffer. Figure 4.11 and Fig. 4.12 show throughput as the number of received packets at the sink per minute in the network and PDR, respectively. Generally, it is clear that CA-OF has better performance in terms of PDR and throughput than ENERGY-OF, ETX-OF and OF0. Also, in Fig. 4.11, we can see that throughput of CA-OF is fluctuated and equals to others in the first 4 minutes. The reason is that the children nodes do not receive DIO messages from their parents when congestion does occur to update

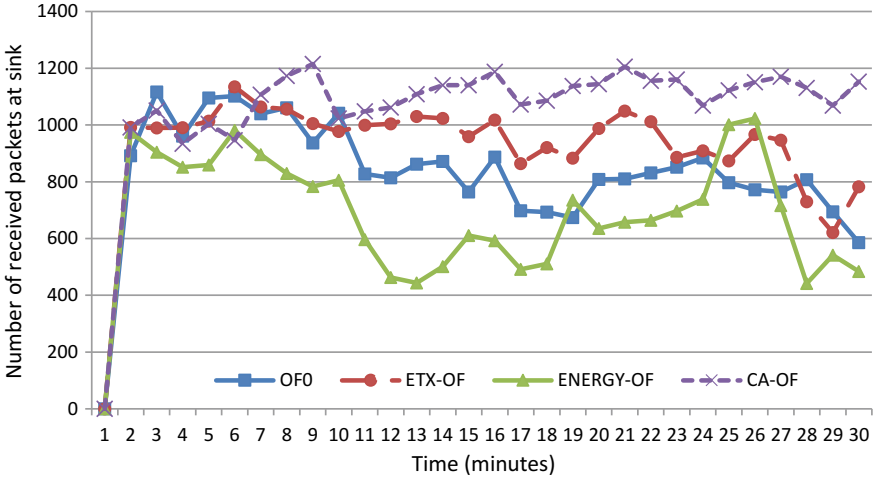

Fig. 4.11 Network 3 throughput

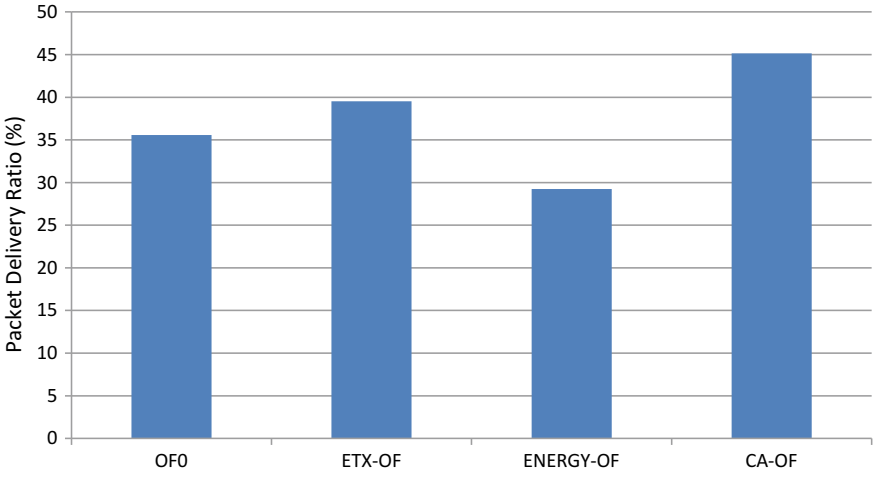

Fig. 4.12 Packet delivery ratio in network 3

Rank value. Transmission of DIO messages is controlled by an algorithm called Trickle algorithm and based on configuration parameters: the minimum interval size (I_{min}), the maximum interval size (I_{max}) and the redundancy constant (K) [14]. In the future, we are going to modify the Trickle algorithm operation to be aware and adaptive to congestion, and therefore Rank value is updated directly when congestion occurs. Lastly, Fig. 4.13 shows the energy consumed in transmission and reception per successfully received packet. This result demonstrates that CA-OF minimizes the energy consumption in the network compared to others.

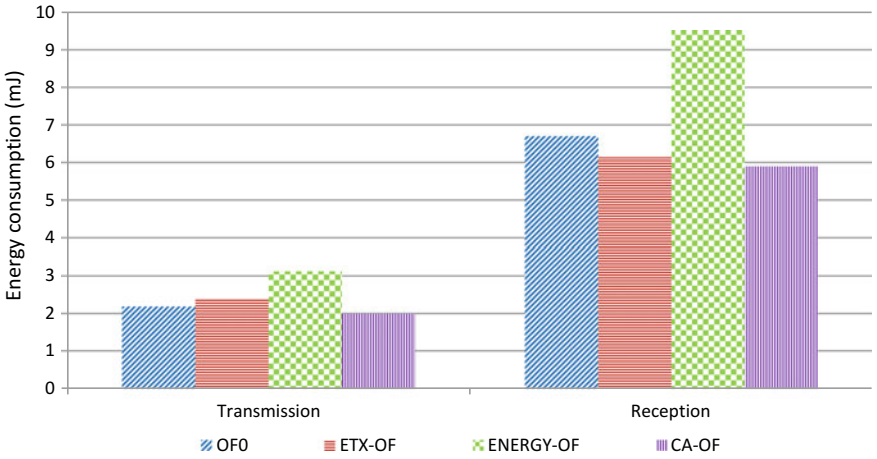

Fig. 4.13 Tx and RX energy consumption per successful packet in network 3

Table 4.2 CA-OF performance compared with others

Objective function	ENERGY-OF (%)	ETX-OF (%)	OF0 (%)
Lost packets	36.1	46	49.9
Throughput	33.7	47.1	60.6
Packet delivery ratio	33.6	47.1	60.6
Energy consumption	25.4	21.4	28.7

Overall, based on these simulation results, it is clear that CA-OF improves performance by the average values shown in Table 4.2 in terms of the number of lost packets, throughput, packet delivery ratio and total communication energy consumption as compared to other objective functions. On the other hand, the limitation of CA-OF is that as the number of intermediate nodes within the coverage area of the sink node is low, the number of possible routes to the sink is reduced. Thus, CA-OF cannot find uncongested nodes to forward high data rate packets to the sink without high packet drops at node buffers. Therefore, the performance advantage of CA-OF increases as the number of nodes close to the sink is high and vice versa.

4.5 Conclusion

In this chapter, a new RPL metric called buffer occupancy, which is important to consider when congestion does occur in 6LoWPAN network, and a new objective function called congestion-aware objective function are proposed in RPL routing protocol. The proposed objective function has been implemented and tested in Contiki with three different size networks and compared with three objective func-

tions. The simulation results show than CA-OF can choose the least congested path from a leaf node to a sink node by forwarding packets through less congested nodes. Hence, CA-OF improves network performance in terms of packet delivery ratio, throughput and energy consumption.

References

1. Zhang T, Li X (2014) Evaluating and analyzing the performance of RPL in contiki. In: Proceedings of the 1st international workshop on mobile sensing, computing and communication, ACM, pp 19–24
2. Winter T, Thubert P, Brandt A, Hui J, Kelsey R (2012) RPL: IPv6 routing protocol for low-power and lossy networks. IETF, RFC 6550
3. Gaddour O, Koubâa A, Baccour N, Abid M (2014) OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol. In: Proceedings of 12th international symposium on modeling and optimization in mobile, Ad hoc, and wireless networks (WiOpt). IEEE, pp 365–372
4. Vasseur J-P, Kim M, Pister K, Dejean N, Barthel D (2012) Routing metrics used for path calculation in low-power and lossy networks. RFC 6551
5. Thubert P (2012) Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552
6. Gnawali O, Levis P (2010) The ETX objective function for RPL. Internet draft: draft-gnawali-roll-etxof-00
7. Gonizzi P, Monica R, Ferrari G (2013) Design and evaluation of a delay-efficient RPL routing metric. In: Proceedings of 9th international wireless communications and mobile computing conference (IWCMC). IEEE, pp 1573–1577
8. Kamgueu PO, Nataf E, Ndié TD, Festor O (2013) Energy-based routing metric for RPL. [Research report] RR-8208, INRIA, 14
9. Karkazis P, Leligou HC, Sarakis L, Zahariadis T, Trakadas P, Velivassaki TH, Capsalis C (2012) Design of primary and composite routing metrics for RPL-compliant wireless sensor networks. In: Proceedings of international conference on telecommunications and multimedia (TEMU). IEEE, pp 13–18
10. Di Marco P, Fischione C, Athanasiou G, Mekikis P-V (2013) MAC-aware routing metrics for low power and lossy networks. In: Proceedings of IEEE INFOCOM. IEEE, pp 13–14
11. Moeller S, Sridharan A, Krishnamachari B, Gnawali O (2010) Routing without routes: the back-pressure collection protocol. In: Proceedings of the 9th ACM/IEEE international conference on information processing in sensor networks. ACM, pp 279–290
12. Xiao W, Liu J, Jiang N, Shi H (2014) An optimization of the object function for routing protocol of low-power and lossy networks. In: Proceedings of 2nd international conference on systems and informatics (ICSAI). IEEE, pp 515–519
13. Tang W, Ma X, Huang J, Wei J (2015) Toward improved RPL: a congestion avoidance multipath routing protocol with time factor for wireless sensor networks. J Sens 2016
14. Levis P, Clausen T, Hui J, Gnawali O, Ko J (2011) The trickle algorithm. Internet engineering task force, RFC 6206

Chapter 5
Game Theory Based Congestion Control Framework

5.1 Introduction

WSNs connected to the Internet through 6LoWPAN have wide applications in industrial, automation, health care, military, environment, logistics, etc. An estimate by Bell Labs suggests that from 50 to 100 billion things are expected to be connected to the Internet by 2020 [1], and the number of the wireless sensor devices will account for a majority of these. Generally, the applications can be categorized into four types: event-based, continuous, query-based and hybrid applications based on the data delivery method [2, 3]. In the hybrid application type, the first three categories are combined into hybrid application, i.e. sensor nodes send packets in response to an event (event based) and at the same time send packets periodically (continuous) as well as send a reply to a sink query (query based). This type of application will be common in the future as WSNs are integrated with the Internet to form the IoT [4]. In the IoT applications, the sensor nodes host many different application types simultaneously (event based, continuous and query based) with varied requirements. Some of them are real-time applications where the application data is time-critical and delay-constrained, while others are non-real-time applications. Some applications send very important data and losing this data is not permitted, e.g. medical applications and fire detection applications. This brings new challenges to the congestion control algorithms and mechanisms designed to be aware of application priorities as well as node priorities. However, according to our best knowledge, none of the existing congestion control literature in WSNs and 6LoWPAN networks supports awareness of both node priorities and application priorities. To address this, later we define a 'priority cost function' to support node priority awareness and distinguish between high-priority nodes and low-priority nodes.

In 6LoWPAN networks, every node selects its parent based on RPL [5] where there are three types of nodes: sink node, intermediate node and leaf node. When congestion occurs, the leaf nodes start to send high data rate packets to their parent node where each leaf node wants to send packets as high as it can in a selfish way

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_5

without considering the remaining channel capacity, the available parent's buffer space and the other leaf nodes' sending rate. This problem can be formulated as a non-cooperative game where each selfish leaf node is modelled as a player in the game. To the best of our knowledge, none of the existing works in the congestion control literature of WSNs and 6LoWPAN networks uses game theory to solve the congestion problem through traffic control (rate adaptation). However, the non-cooperative game theory gives a natural and suitable framework to study and formulate the congestion control problem in 6LoWPAN networks where the nodes (players) are non-cooperative in their behaviours and each node demands high data rate in a selfish way. Also, the non-cooperative game theory provides an optimal solution concept, which is Nash equilibrium, where each player (node) plays a strategy (sending rate) to maximize its payoff given the strategies of other players.

This work is motivated by these considerations to propose a new congestion control algorithm called 'Game Theory based Congestion Control Framework' (GTCCF) which uses the non-cooperative game theory framework to solve the congestion problem and is aware of both node priorities and application priorities to support the IoT application requirements. Our main contributions in this chapter include the following:

- Design a congestion control game for mitigating congestion in 6LoWPAN networks. The node's payoff function is formulated to achieve the node demand (preference) for sending high data rate (utility function) and the desirable fairness among leaf nodes according to their priorities (priority cost function), while alleviating and mitigating congestion in the network (congestion cost function).
- Prove the existence and uniqueness of Nash equilibrium in the formulated congestion control game. Also, the node's payoff function is modelled as a constrained nonlinear optimization problem which is solved by using Lagrange multipliers and KKT conditions such that each node obtains its optimal solution (sending rate) that satisfies the congestion alleviation.
- By using the formulated game, we propose a novel and simple congestion control algorithm called GTCCF which is aware of node priorities and application priorities to support the IoT application requirements. Also, the proposed framework is designed and built on the unique characteristics of the IEEE 802.15.4 standard, IPv6 and 6LoWPAN protocol stack.
- Implement and evaluate the performance of the proposed framework in the real IoT operating system, Contiki OS [6], through Cooja simulator [7].

The remainder of the chapter is organized as follows: in Sect. 5.2 introduces a non-cooperative game framework for congestion control, proves the existence of a unique Nash equilibrium and computes the optimal solution for the designed game. The implementation of the congestion control game in 6LoWPAN networks is provided in Sect. 5.3. In Sect. 5.4, simulation scenarios and results are given. Finally, Sect. 5.5 draws conclusions.

5.2 Game Theoretic Formulation

5.2.1 Network Setup and Problem Formulation

In 6LoWPAN networks, the RPL routing protocol [5] is responsible for constructing the network topology where three types of nodes are defined: sink (root) node which provides connectivity to other networks and intermediate node which forwards packets to the sink and leaf node. Consider a network of one sink node, S, a set of intermediate nodes, I, and a set of leaf nodes, L, as shown in Fig. 5.1. We consider a group of leaf nodes $(L_1, L_2, \ldots, L_m)$ are competing to send data packets to the sink node through path I_1 (parent), $I_2, \ldots, I_l$ (dash lines in Fig. 5.1). We denote by L_k to leaf node k; $\forall k \in M$ where $M = \{1, 2, \ldots, k, \ldots, m\}$. Also, we assume that (i) each node in the network has a buffer size of B packets and (ii) the leaf nodes have different priorities $P = \{p_1, \ldots, p_k, \ldots, p_m\}$ where p_k is the priority of node L_k; $\forall k \in M$. The priorities of leaf nodes are specified by user based on importance of node and importance of hosted applications, (iii) Each leaf node hosts N applications with different priorities where $N = \{1, 2, \ldots, j, \ldots, n\}$; we denote by p_k^j to the priority of application j hosted in leaf node L_k for all $k \in M$ and $j \in N$. The priorities of hosted applications are specified by user based on importance and type of application (i.e. real-time application, reliable application, etc.) and (iv) each leaf node L_k has a maximum sending packet rate of λ_k^{max}.

In 6LoWPAN networks, when congestion occurs, the leaf nodes $(L_1, L_2, \ldots, L_m)$ start to send high data rate packets to their parent (I_1) in a selfish way where each leaf node wants to send as many packets as it can without taking into account the available

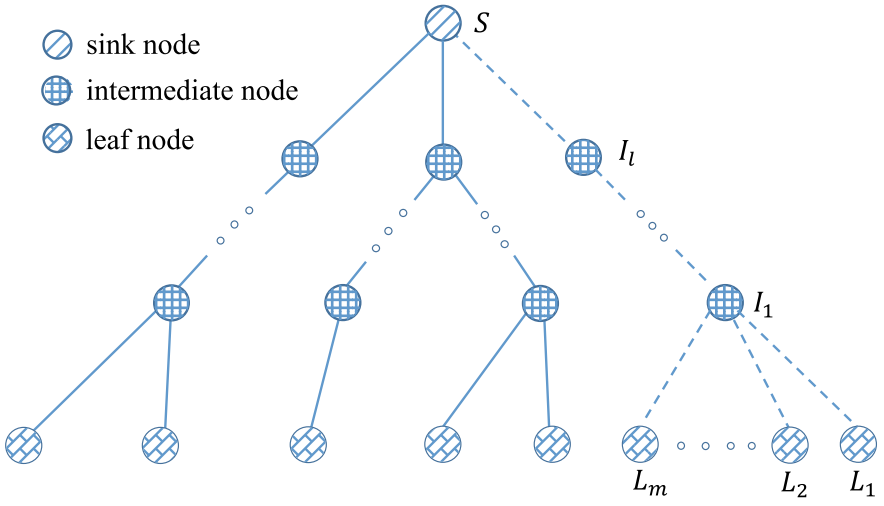

Fig. 5.1 RPL-based network topology

channel bandwidth, the buffer occupancy of the parent, the forwarding (service) rate of the parent node and sending rate of other leaf nodes. This will increase packet loss, energy consumption and end-to-end delay, decrease the network performance and throughput and impact on the QoS aspects. These selfish leaf nodes and their parent can be modelled as the following non-cooperative game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$ where

- Players: We have a group of M players (leaf nodes), $L_1, \ldots, L_k, \ldots, L_m$ where m represents number of leaf nodes which are associated with parent, I_1.
- Strategies: S_k; $\forall k \in M$ represents the feasible action space for player L_k. Each node (player) L_k can send a minimum data rate of zero and a maximum data rate of λ_k^{max}. Thus, $S_k = [0, \lambda_k^{max}]$ and the strategy space for all players is $SS = \prod_{k=1}^{m} S_k = [0, \lambda_1^{max}] \times \cdots \times [0, \lambda_k^{max}] \times \cdots \times [0, \lambda_m^{max}]$.
- Payoff function: We use $\Phi_k : SS \rightarrow \mathbb{R}$ to represent payoff function of player L_k; $\forall k \in M$. The objective function of player L_k is to optimize its profit by maximizing its payoff function Φ_k with respect to λ_k over $[0, \lambda_k^{max}]$.

In our framework, the payoff function is modelled to reflect the leaf node demand (desire) for sending high data rate (utility function), how much the parent node is congested due to the leaf nodes (congestion cost function) and the importance (priority) of the leaf node (priority cost function). Thus, the payoff function includes the following three functions:

- Utility function: We use $U_k(\lambda_k)$ to represent the utility function of player L_k where λ_k is sending rate (strategy) of player L_k. The utility function is designed such that each player gets more profit by increasing its sending rate. Many types of utility function are commonly used such as exponential, logarithmic, linear and sigmoidal [8]. In our framework, we use the logarithmic utility function as it has strict concavity property. Thus, we select the utility function of player L_k as follows:

$$U_k(\lambda_k) = \log(\lambda_k + 1). \tag{5.1}$$

- Congestion cost function: We use $C_k(\lambda_k, \lambda_{-k})$ to represent the congestion cost of node (player) L_k where $\lambda_{-k} = [\lambda_j]_{j\in M; j\neq k}$ is the vector of sending rates (strategies) of all players except player L_k and $s = (\lambda_k, \lambda_{-k}) \in SS$ is referred to as the strategy profile. This function reflects how much the parent node is congested due to the leaf nodes. One can use the buffer loss probability derived in Chap. 3 (Eq. 3.11) or Erlang B formula (blocking probability) to model the congestion cost function. However, the second partial derivative of these equations is a complex equation and implementing this equation on a limited processing capability sensor node is very difficult and impractical. According to Queuing theory, if the arrival rate at the parent node's buffer is higher than the service rate from the parent, the buffer starts overflowing the packets and congestion occurs. Thus, one possible method is to choose the congestion cost function as the ratio between the total receiving rate and total forwarding rate at the parent's buffer. As the receiving rate is greater than the forwarding rate, the ratio increases. Also, the number of leaf

nodes has an impact on congestion. As the number of leaf nodes, m, increases, the congestion situation becomes worse at the parent. Assume that a number of sending packets from the leaf nodes are lost on the wireless channel before they arrive at the parent node with a probability of $P^k_{ch-loss}; \forall k \in M$. Thus, the congestion cost function can be defined as follows:

$$C_k(\lambda_k, \lambda_{-k}) = m \frac{\sum_{k=1}^{m} (1 - P^k_{ch-loss})\lambda_k + 1}{\lambda_{out} + 1}, \tag{5.2}$$

where λ_{out} is the outgoing rate from the parent node such that $\lambda_{out} \geq 0$.
In Chap. 3, congestion analysis for 6LoWPAN networks with different parameters and various scenarios was explored. It demonstrated that the majority of packets are lost in the nodes' buffer as compared to wireless channel loss when congestion occurs. For example, with high offered load (i.e. 8 packets/second), the percentage of packet loss due to buffer overflow is up to 99.66% compared to 0.33% due to channel loss. Therefore, to simplify the analysis, we assume that $P^k_{ch-loss}$ in Eq. (5.2) is zero; $\forall k \in M$. Thus, $C_k(\lambda_k, \lambda_{-k})$ becomes as follows:

$$C_k(\lambda_k, \lambda_{-k}) = m \frac{\lambda_{in} + 1}{\lambda_{out} + 1}, \tag{5.3}$$

where $\lambda_{in} = \sum_{k=1}^{m} \lambda_k$, $\lambda_{out} \geq 0$ and $0 \leq \lambda_k \leq \lambda_k^{max}$ for all $k \in M$.

Remark 5.2.1 We add 1 to λ_k in Eq. (5.1) and to λ_{out} in the denominator of Eq. (5.2) to avoid making the values of utility function and congestion cost function equal to $-\infty$ and ∞, respectively. Since the value of λ_k ranges from zero to λ_k^{max} and the value of λ_{out} is greater than or equal to zero; therefore, without adding 1, $U_k(\lambda_k) = -\infty$ when $\lambda_k = 0$ and $C_k(\lambda_k, \lambda_{-k}) = \infty$ when $\lambda_{out} = 0$ for all $k \in M$.

- Priority cost function: We use $P_k(\lambda_k; p_k)$ to represent the priority cost function of player $L_k; \forall k \in M$. Player L_k has to pay a penalty based on its priority (p_k) and its sending rate (λ_k) to distinguish between high-priority nodes and low-priority nodes. A player with less p_k value has high priority (e.g. if $p_i = 1$ and $p_j = 2$, this means that player L_i has higher priority than player L_j). Therefore, the priority cost function of player L_k can be defined as follows:

$$P_k(\lambda_k; p_k) = p_k \lambda_k. \tag{5.4}$$

After we define the utility function $U_k(\lambda_k)$, congestion cost function $C_k(\lambda_k, \lambda_{-k})$ and priority cost function $P_k(\lambda_k; p_k)$ for player $L_k; \forall k \in M$; therefore, the payoff function of player L_k can be stated as follows:

$$\Phi_k(\lambda_k, \lambda_{-k}) = \omega_k \log(\lambda_k + 1) - \alpha_k m \frac{\lambda_{in} + 1}{\lambda_{out} + 1} - \beta_k p_k \lambda_k, \tag{5.5}$$

where ω_k, α_k and β_k are player preference parameters of functions $U_k(\lambda_k)$, $C_k(\lambda_k, \lambda_{-k})$ and $P_k(\lambda_k; p_k)$, respectively, such that $\omega_k, \alpha_k, \beta_k > 0$; $\forall k \in M$. The values of ω_k, α_k and β_k are chosen by user to satisfy the system objective and requirement. For example, as the value of β_k is greater, the difference between sending rate (λ_k) of high-priority node and low-priority node is higher and vice versa.

A non-cooperative game has a solution when Nash equilibrium exists. In the congestion control game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$, a vector of strategies (sending rates) $s^* \in SS$ is called Nash equilibrium if no player can improve its payoff by changing its strategy while other players maintain their current strategies where $s^* = [\lambda_1^*, \ldots, \lambda_k^*, \ldots, \lambda_m^*]$. Mathematically, in this game, Nash equilibrium is M-tuple $\{\lambda_k^*\}_{k\in M}$ that satisfies the following:

$$\Phi(\lambda_k^*, \lambda_{-k}^*) \geq \Phi(\lambda_k, \lambda_{-k}^*),$$

$\forall \lambda_k^*, \lambda_k \in S_k, \lambda_k^* \neq \lambda_k, \forall k \in M$.

Lemma 5.2.2 *In the congestion control game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$, $\forall k \in M$, every strategy set S_k is compact and convex, $\Phi_k(\lambda_k, \lambda_{-k})$ is continuous function in the profile of strategies $s \in SS$ and concave in S_k; then, the game G has at least one Nash equilibrium.*

Proof The strategy set for all players $\{L_k\}_{k\in M}$ is $SS = \prod_{k=1}^{m} S_k$ where $0 \leq S_k \leq \lambda_k^{max}$; $\forall k \in M$. As $S_k = [0, \lambda_k^{max}]$, the strategy set of player L_k (S_k) is closed and bounded. Thus, the set S_k is compact for all $k \in M$.

Assume two points $x, y \in S_k$ and $\gamma = [0, 1]$. Thus, we have

$$0 \leq \gamma x + (1 - \gamma)y \leq \lambda_k^{max},$$

this means that the point $\gamma x + (1 - \gamma)y \in S_k$. Therefore, we can say that the set S_k is convex; $\forall k \in M$.

Consider the following twice-differentiable payoff function of player L_k:

$$\Phi_k(\lambda_k, \lambda_{-k}) = \omega_k \log(\lambda_k + 1) - \alpha_k m \frac{\lambda_{in} + 1}{\lambda_{out} + 1} - \beta_k p_k \lambda_k.$$

In order to determine the concavity of the payoff function, we define Hessian of $\Phi_k(s)$, where $s = \{\lambda_k\}_{k\in M}$, as follows:

$$H(s) = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1m} \\ A_{21} & A_{22} & \ldots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \ldots & A_{mm} \end{bmatrix}, \tag{5.6}$$

where $A_{kj} = \dfrac{\partial^2 \Phi_k}{\partial \lambda_k \partial \lambda_j}$ $\forall k, j \in M$.

For all λ_k such that $\omega_k, \alpha_k, \beta_k > 0$ and $\lambda_{out} > 0; \forall k \in M$,

$$A_{k,j} = \begin{cases} -\frac{\omega_k}{(\lambda_k+1)^2} < 0 & \text{if } k = j; \forall k, j \in M \\ 0 & \text{if } k \neq j; \forall k, j \in M \end{cases}. \tag{5.7}$$

According to the leading principal minor of $H(s)$, it is clear that $H(s)$ is negative definite for all $s \in SS$, thus $\Phi_k(\lambda_k, \lambda_{-k})$ is strictly concave in $S_k; \forall k \in M$.

According to the Nikaido Isoda theorem [9], these conditions (in *Lemma* 5.2.2) are sufficient to satisfy the existence of at least one Nash equilibrium in the game G. ■

Lemma 5.2.3 *The congestion control game* $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$ *admits unique Nash equilibrium in its pure strategy space.*

Proof Let $r = (r_1, r_2, \ldots, r_m)$ be an arbitrary vector of fixed positive parameters. Based on Rosen's Theorem (Theorem 2) [10], we define the weighted nonnegative sum of the payoff functions $\Phi_k(\lambda_k, \lambda_{-k}); \forall k \in M$ as follows:

$$\sigma(\lambda_k, \lambda_{-k}; r) = \sum_{k=1}^{m} r_k \Phi_k(\lambda_k, \lambda_{-k}), \quad r_k \geq 0. \tag{5.8}$$

The pseudogradient of $\sigma(\lambda_k, \lambda_{-k}; r)$ is given by

$$g(\lambda_k, \lambda_{-k}; r) = \begin{bmatrix} r_1 \nabla \Phi_1(\lambda_1, \lambda_{-1}) \\ r_2 \nabla \Phi_2(\lambda_2, \lambda_{-2}) \\ \vdots \\ r_m \nabla \Phi_m(\lambda_m, \lambda_{-m}) \end{bmatrix}, \tag{5.9}$$

where $\nabla \Phi_k(\lambda_k, \lambda_{-k}) = \dfrac{\omega_k}{\lambda_k + 1} - \alpha_k m \dfrac{1}{\lambda_{out} + 1} - \beta_k p_k, \forall k \in M$.
Now, we define the Jacobian matrix $(G(\lambda_k, \lambda_{-k}; r))$ of $g(\lambda_k, \lambda_{-k}; r)$ with respect to λ_k as follows:

$$G(\lambda_k, \lambda_{-k}; r) = \begin{bmatrix} B_{11} & B_{12} & \ldots & B_{1m} \\ B_{21} & B_{22} & \ldots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \ldots & B_{mm} \end{bmatrix}, \tag{5.10}$$

where $B_{i,j} = r_i A_{i,j}; \forall i, j \in M$.

Now, it is clear that the symmetric matrix $[G(\lambda_k, \lambda_{-k}; r) + G^T(\lambda_k, \lambda_{-k}; r)]$ is negative definite for all $\lambda_k, \lambda_{-k} \in SS$. Then, Rosen's Theorem (Theorem 6) [10] states that the function $\sigma(\lambda_k, \lambda_{-k}; r)$ is diagonally strictly concave. Therefore, according to Rosen's Theorem (Theorem 2) [10], the game G has unique Nash equilibrium in its pure strategy space. ■

5.2.2 Game Solution Computation

After we design the congestion control game and prove the uniqueness of Nash equilibrium in the strategy space of each player, we need to find and compute the optimal game solution (λ_k^*) where each player chooses a strategy that maximize its payoff function. Consider the following constrained nonlinear optimization problem (***P***):

$$\begin{aligned} \underset{\lambda_k \in S_k}{\text{maximize}} \quad & \Phi_k(\lambda_k, \lambda_{-k}), \\ \text{subject to} \quad & \lambda_k \geq 0, \\ & \lambda_k \leq \lambda_k^{max}, \ \forall k \in M. \end{aligned} \tag{5.11}$$

In order to solve the problem (***P***), we introduce the Lagrange multipliers u_k and v_k and define the Lagrangian function $\mathcal{L}_k(\lambda_k, u_k, v_k)$ for player L_k; $\forall k \in M$ as follows:

$$\mathcal{L}_k = \Phi_k(\lambda_k, \lambda_{-k}) + u_k\lambda_k + v_k(\lambda_k^{max} - \lambda_k), \tag{5.12}$$

where the KKT conditions of player L_k for optimality are as follows:

$$\begin{gathered} u_k, v_k \geq 0, \\ \lambda_k \geq 0, \\ \lambda_k^{max} - \lambda_k \geq 0, \\ \nabla_{\lambda_k}\Phi_k(\lambda_k, \lambda_{-k}) + u_k\nabla_{\lambda_k}(\lambda_k) + v_k\nabla_{\lambda_k}(\lambda_k^{max} - \lambda_k) = 0, \\ u_k(\lambda_k), v_k(\lambda_k^{max} - \lambda_k) = 0. \end{gathered}$$

The optimal data rate (λ_k^*) for player L_k; $\forall k \in M$ can be computed by solving the problem (***P***) and it is as follows:

$$\lambda_k^* = \begin{cases} 0 & \text{if condition 1} \\ \lambda_k^{max} & \text{if condition 2} \\ \frac{\omega_k(\lambda_{out}+1)}{\alpha_k m + \beta_k p_k(\lambda_{out}+1)} - 1 & \text{otherwise} \end{cases}, \tag{5.13}$$

where condition 1 and condition 2, respectively, are

$$\frac{\alpha_k m}{\lambda_{out}+1} + \beta_k p_k \geq \omega_k, \tag{5.14}$$

$$\frac{\alpha_k m}{\lambda_{out}+1} + \beta_k p_k \leq \frac{\omega_k}{\lambda_k^{max}+1}. \tag{5.15}$$

From Eq. 5.13, the optimal sending rate (λ_k^*) of leaf node L_k depends on the node preference parameters (ω_k, α_k and β_k), the forwarding rate of L_k's parent (λ_{out}) and the number of leaf nodes (m) which forward their packets through L_k's parent node.

The parameters ω_k, α_k and β_k are already known by the node L_k and the parameters λ_{out} and m are sent back to the node L_k by its parent when congestion occurs through broadcasting a DIO message piggybacked with congestion information.

5.2.3 Distribution of Node's Sending Rate Among Applications

In the IoT application, it is important for each node to be aware of the priorities of the hosted applications. We assume that a leaf node, L_k, hosts N applications with different priorities where $N = \{1, 2, \ldots, n\}$. We denote by p_k^j the priority of application j hosted in leaf node L_k where an application with less value of p_k^j has higher priority. After the leaf node calculates its sending rate (λ_k^*) based on the game theory framework, the value of λ_k^* is distributed among applications according to their priorities as follows:

$$\lambda_k^j = \theta_j \lambda_k^*, \tag{5.16}$$

$$\theta_j = \begin{cases} 1 & \text{if } n = 1 \\ \dfrac{\sum\limits_{i=1;i\neq j}^{n} p_k^i}{(n-1)\sum\limits_{i=1}^{n} p_k^i} & \text{if } n > 1 \end{cases}, \tag{5.17}$$

$$\sum_{j=1}^{n} \theta_j = 1, \tag{5.18}$$

where λ_k^j is the sending rate of application j hosted in leaf node L_k, θ_j is weight of application j and n is the number of applications such that $p_k^j > 0$ for all $k \in M$ and $j \in N$.

5.3 Game Theory Framework Implementation

In 6LoWPAN networks, the network topology is governed by RPL routing protocol through transmission of DIO, DAO and DIS control messages. The DIO transmission strategy is controlled by the Trickle algorithm. However, the Trickle algorithm is not aware of the occurrence of congestion. Thus, the operation of the algorithm is modified such that when congestion occurs at the parent node, the DIO packet is immediately sent and congestion information is piggybacked on it.

Initially, a leaf node (L_k) selects its initial sending rate based on its priority (p_k) and its maximum sending rate (λ_k^{max}) as follows:

$$\lambda_k^{(initial)} = \frac{\lambda_k^{max}}{p_k}; \qquad \forall k \in M. \tag{5.19}$$

The parent node periodically checks the congestion conditions every interval time 'I_{check}'. The value of I_{check} has to be in the right range (e.g. typically 1, 2 or 3 s). Below this range, the adaptation of the sending rate fluctuates wildly and also this will increase the number of overhead DIO notification packets sent. If the value of I_{check} is too large, congestion may occur and the node will not check frequently enough. If the arrival rate (λ_{in}) at the parent's buffer is higher than service rate (λ_{out}), the parent's buffer will be blocked and congestion does occur. As a result, the parent node broadcasts a DIO packet which contains the congestion cost function information. The forwarding rate of parent λ_{out} is not constant with time. It is increased or decreased due to the operation of the CSMA algorithm (i.e. backoff time), MAC parameters (i.e. channel check rate) and number of active nodes. Thus, to avoid sending high overhead DIO packets and fluctuating the sending rate of leaf nodes, we use Brown's simple exponential smoothing model [11] to estimate the actual maximum sending rate as follows:

$$\lambda_{out}(t+1) = \psi\lambda_{out}(t) + (1-\psi)\lambda_{out}(t-1), \tag{5.20}$$

where $\lambda_{out}(t+1)$, $\lambda_{out}(t)$ and $\lambda_{out}(t-1)$ are the expected, current and historical forwarding rate of the parent, respectively, and ψ is smoothing factor such that $0 < \psi < 1$. A large value of ψ reduces the level of smoothing and gives high weight to current measurement of λ_{out}, while a value of ψ close to zero gives greater smoothing effect and less responsive to recent changes in λ_{out} value. In this chapter, we set the value of ψ to 0.4. Also, the parent node sends DIO packet when the number of leaf nodes, m, changes because the optimal sending rate (Nash equilibrium) of each leaf node will change. When the leaf nodes receive the DIO message, they update their sending rate according to Eq. (5.13), where the parameters ω_k, α_k, β_k and p_k are already known to the player L_k; $\forall k \in M$. After that, the leaf node distributes the updated sending rate (λ_k) among the hosted applications according to their priorities as in Eqs. (5.16) and (5.17). Algorithm 1 shows the procedures of GTCCF.

5.4 Performance Evaluation

The proposed congestion control framework has been tested and evaluated on different network scenarios through simulation by using the Contiki 3.0 OS and Cooja simulator. In related work, four proposed algorithms exist that use traffic control strategies. These algorithms are DCCC6 [12], Griping [13], Deaf [13] and Fuse [13]. The working principle of Deaf and Fuse algorithms is based on ACK packet loss as the congestion indicator. However, it is impractical to use ACK packet loss to detect congestion in the network because other reasons for missing ACK exist such as packet error in the wireless channel. Therefore, our proposal is compared

Algorithm 1 Congestion control framework

1: ***Input***:
ω_k preference parameter of $U(\lambda_k)$
α_k preference parameter of $C(\lambda_k, \lambda_{-k})$
β_k preference parameter of $P(\lambda_k; p_k)$
λ_k^{max} maximum sending rate
I_{check} congestion check interval time
ψ smoothing factor
2: ***Output***:
An optimal sending rate to eliminate congestion
3: ***At each parent***:
timer_set(congestion_timer,I_{check});
If (timer_expired(congestion_timer)) then
If ($\lambda_{out} < \lambda_{in}$ or m changes) then
DIO.send();
End
timer_reset(congestion_timer);
End
4: ***At each leaf***:
$p_k \leftarrow$ priority of node L_k;
$p_k^j \leftarrow$ priority of application j;
$\lambda^{initial} \leftarrow$ Eq. (5.19);
If (a new DIO message is received) then
$\lambda_k^* \leftarrow$ Eq. (5.13);
$\lambda_k^j \leftarrow$ Eq. (5.16);
End

with DCCC6 and Griping. In the simulation, we have used one sink node, a set of intermediate nodes and a group of leaf nodes which at the beginning, start sending packets at high data rate (6 packets/s) to create a congested situation. During the simulation, the leaf nodes start sending packets after 60 s so the network topology construction is completed where the simulation time is set to 600 s. Cooja simulates the hardware of a set of real sensor nodes such as Tmote Sky which is used in the simulation. Also, Cooja simulator implements a number of wireless channel models such as unit disk graph medium (UDGM)–distance loss which is used in the simulation since interference is considered [14]. We use Powertrace [15] to measure the energy consumption of each node where it is a runtime network-level power profiling system that uses state tracking to estimate the energy consumption and it is accurate up to 94%. The protocol stack and simulation parameters used in the simulation are shown in Table 5.1. For our proposal, we have set $I_{check} = 384$ clock ticks, $\omega_k = 15$, $\alpha_k = 7$, $\beta_k = 0.9$, $\psi = 0.4$ and $\lambda_k^{max} = 8$ packet/s; $\forall k \in M$ where each 128 clock ticks = 1 s.

Table 5.1 Protocol stack and simulation parameters

Layer	Protocol	Parameter value
Application	Every leaf node sends high data rate packets to sink	Application payload = 30 bytes
Transport	UDP	
Network	uIPv6 + RPL	Objective function = OF0
Adaptation	SICSlowpan layer	Compression method = HC06
Data link	CSMA (MAC layer) Contikimac (RDC layer) 802.15.4 (framer)	Buffer size = 8 packets MAC reliability (ACK) = enabled MAC max. retransmission = 3 Channel check rate = 8 Hz max. frame size = 127 bytes
Physical	CC2420 RF transceiver	

5.4.1 DCCC6 and Griping Implementation

In Contiki 3.0 OS, when the outgoing packet is unicast, the MAC layer stores the packet in its buffer to check whether the channel is free before transmission. In DCCC6 and Griping, the congested node sends a unicast notification packet to the source node when congestion occurs since the buffer is full most of the time. Therefore, the probability of loss of the notification packet due to buffer overflow is high. In this case, the congestion situation gets worse as the source node does not know about the congestion and it increases its sending rate. To avoid this, the sending of a notification packet is modified from unicast to broadcast where the packet is sent directly without storing it at the node's buffer.

DCCC6 detects congestion by using a dynamic buffer occupancy threshold similar to the one used in [16] where the buffer is monitored per incoming packet as follows:

$$threshould(k) = threshold(k-1) + \frac{I}{2^{k-1}}, \tag{5.21}$$

where k is a small integer and I is a constant increment of the queue length. In the simulation, we set $threshold(0) = 3$ and $I = 2$.

When the buffer occupancy is above $threshold(k)$, the congested node sends notification to source nodes. Each time, the congestion notification is received, the sending rate is decreased by increasing the inter-packet interval t_i by α as follows:

$$t_{i+1} = t_i + \alpha = t_i + \frac{\gamma \times \sqrt{t_{max}}}{\sqrt{t_i}}, \tag{5.22}$$

where t_{max} is a maximum inter-packet interval and γ is a slop factor ($\gamma > 1$). In the simulation, we set $\gamma = 2$ and $t_{max} = 7680$ clock ticks (1 min).

Periodically every t_i, the sending rate is increased by reducing t_i by t_i/δ as follows:

$$t_{i+1} = t_i - \frac{t_i}{\delta}, \tag{5.23}$$

$$\delta = \frac{\beta \times t_i \times \sqrt{n_1 + 1}}{(\epsilon \times \sqrt{t_{min}}) - \sqrt{t_i}}, \tag{5.24}$$

where t_{min} is a minimum inter-packet interval, n_i is the number of active children and $\beta > 1$. In the simulation, we set $\beta = 4$ and according to Table 5.1 in [17], for channel check rate = 8, $t_{min} = 16$ and $\epsilon = 21.8$.

For Griping, when a node receives a new packet, it checks its queue length. If the queue length is greater than a threshold, Q_{thr}, the node sends back a control message. However, the receiver cannot send more than one control message to the same sender during K seconds. Whenever the sender receives the control message, it halves its transmission rate. If no control message has been received during T seconds, the sender increments its transmission rate. According to [13], we set $Q_{thr} = 6$ packets, $k = 13$ clock ticks and $T = 96$ clock ticks.

5.4.2 *Sending Rate Adaptation Comparison*

Figure 5.2 compares the rate adaptation mechanisms used in Griping, DCCC6 and GTCCF. First, Griping algorithm employs the original AIMD policy for controlling the sending rate where the rate is increased linearly by a small fixed step every T seconds. Once congestion occurs, the rate is decreased to half and then again linearly increased. Second, DCCC6 algorithm uses a modified AIMD mechanism where the

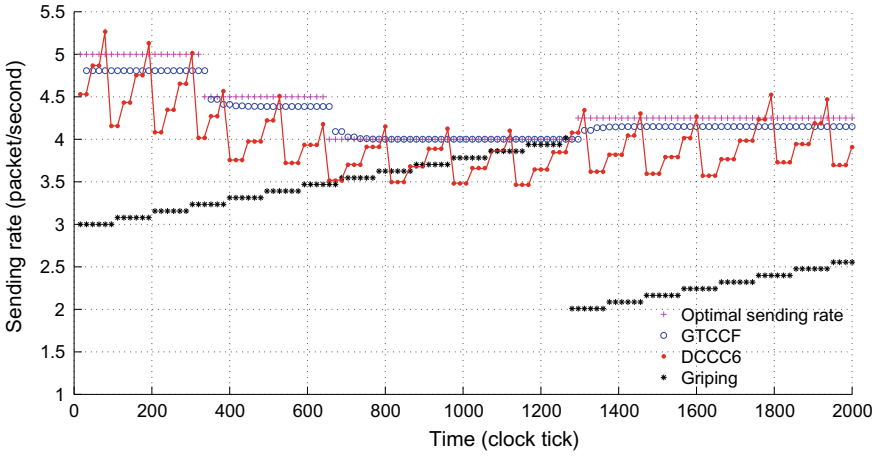

Fig. 5.2 Sending rate adaptation comparison

sending rate is increased by a variable step every t_i. For example, at time 1168 clock tick, the rate is increased from 3.5 to 3.65 (increasing step = 0.15), whereas at time 1360 clock tick, the increasing step is 0.2. On the other hand, the decreasing step is variable and smaller than the step of the original AIMD. Finally, in GTCCF algorithm, game theory is applied adapting the sending rate where the rate is calculated when congestion occurs or the number of leaf nodes changes. From Fig. 5.2, it is obvious that the sending rate in GTCCF is closer to the optimal sending rate than others. Also, the modified AIMD used in DCCC6 can be seen to have better rate adaptation than the original AIMD mechanism used by Griping.

5.4.3 Scenario 1

In the first scenario, we use a simple network with one sink node, one intermediate node and three leaf nodes (L_1, L_2 and L_3) to demonstrate the behaviour and performance of our proposal (GTCCF) compared with other algorithms (DCCC6 and Griping). We have set the priorities of leaf nodes (L_1, L_2 and L_3) to $p_1 = 1$, $p_2 = 2$ and $p_3 = 3$, respectively. Nodes L_1 and L_2 host two applications each with priorities $p_1^1 = 1$, $p_1^2 = 3$, $p_2^1 = 1$ and $p_2^2 = 2$, respectively, whereas L_3 hosts one application.

Figure 5.3 shows the number of received packets every second from the leaf nodes at the sink. In GTCCF, initially, the leaf nodes start to send with initial sending rate as in Eq. 5.19. After that, when congestion occurs, the leaf nodes adapt their sending rate according to the derived solution as in Eq. 5.13. For GTCCF, it is clear that the node (L_1) with higher priority has the highest number of received packets (≈ 1.4 packet/s) as compared to other nodes, whereas the node L_3 has the lowest number of

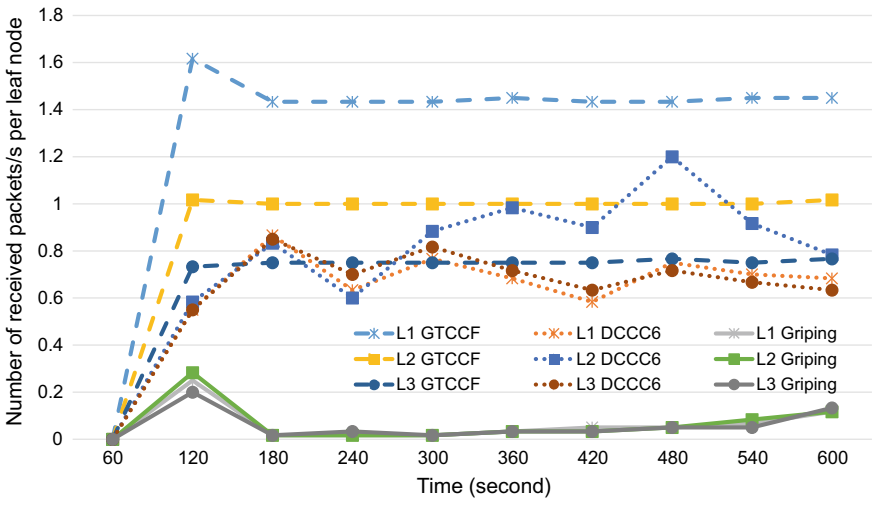

Fig. 5.3 Number of received packets/s from leaf nodes at sink

received packets (≈ 0.75 packet/s) as it has lower priority than others. For DCCC6 and Griping, the nodes do not obtain sending rates according to their priorities. For example, in DCCC6, the node L_2 has higher sending rates than others, while the node L_1 has the highest priority. The reason is that GTCCF is aware of node priorities where each node gets sending rate according to its priority; however, DCCC6 and Griping do not consider the node priorities in their operation. Also, from this figure, we can see that GTCCF has stable performance (number of received packets at sink) with time as GTCCF computes the optimal sending rate (Nash equilibrium) for each leaf node and this rate is still stable unless the number of leaf nodes changes or the service rate at the intermediate node is less than the incoming rate. On the other hand, DCCC6 has fluctuating sending rate. The reason is that DCCC6 uses modified AIMD where the sending rate is continuously increased every inter-packet interval (t_i) by a variable amount and decreased by α when congestion does occur and then it starts increasing every t_i. While Griping has the lowest throughput per leaf node as it uses the original AIMD where the sending rate is incremented every interval time by a small fixed step and decreased to half when congestion occurs. Also, Fig. 5.3 shows that the modified AIMD used in DCCC6 has better performance in term of throughput than the original AIMD used in Griping.

Figure 5.4 shows the overall throughput which is the total number of received packets every second at the sink node. It is clear that GTCCF has stable and higher throughput as compared to other algorithms as well as DCCC6 has better throughput than Griping algorithm for the same reasons stated above. Figure 5.5 shows the sending rate of applications hosted in the leaf nodes for GTCCF where L_1 and L_2 host two applications each and L_3 hosts only one. It is obvious that each node distributes its sending rate among hosted applications according to their priorities. For example, in the node L_1, application 1 (App.1) obtains high sending rate

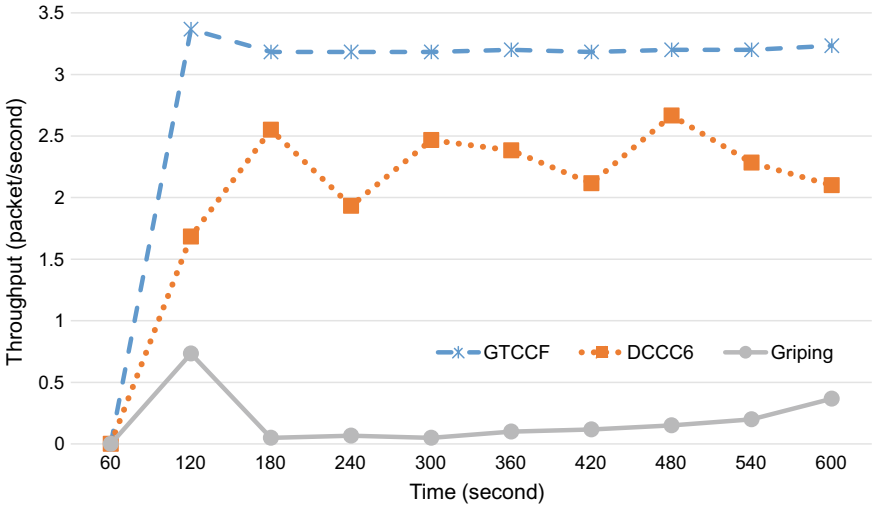

Fig. 5.4 Number of received packets/second at sink

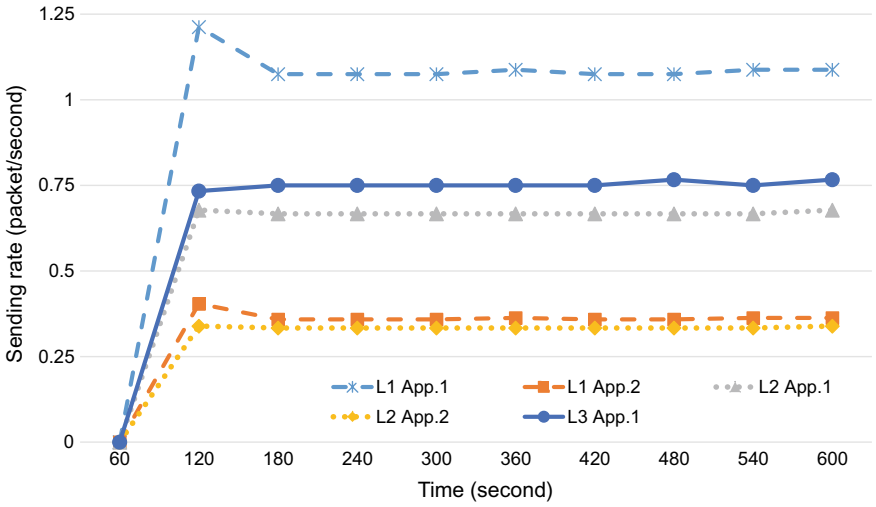

Fig. 5.5 Applications' sending rate for GTCCF

($\approx$ 1.1 packet/s) as compared to application 2 (App.2) ($\approx$ 0.35 packet/s) which has low priority. While the node L_3 allocates all its sending rate to application 1 as it is hosted alone.

Figure 5.6 shows end-to-end delay which is the time between a packet being generated at the application of the source until its successful reception at the application of the final destination. It is clear that GTCCF and Griping have lower end-to-end delay as compared to DCCC6. In GTCCF and Griping, initially, when congestion occurs, the delay is high because the buffer is full so packet waiting time in the buffer is high. After that, when each node computes its optimal sending rate (in GTCCF) or halves its sending rate (in Griping), the delay of packets will decrease. On the other hand, DCCC6 has higher delay than other algorithms because the nodes' sending rates are increased periodically every t_i and decreased when congestion occurs and then increased and this process continues. As a result, the packets wait a long time in the nodes' buffers.

Figure 5.7 shows the energy consumption due to transmission and reception in the leaf and intermediate nodes per successfully delivered packet (i.e. energy consumption per packet = total energy consumption due to Tx and Rx/total number of received packets at sink). We note that with GTCCF, the energy consumption in the network is less than others as DCCC6 and Griping waste energy by transmitting and receiving packets which are then lost due to buffer overflow on the path without successful delivery. Also, the consumed energy per packet in Griping is significantly higher than others as the number of delivered packets to sink in Griping is much lower than others. Figure 5.8 shows the total number of lost packets in the network due to buffer overflow. It is obvious that GTCCF loses less packets at the buffer than others. GTCCF loses packets at the beginning and after the optimal sending rates

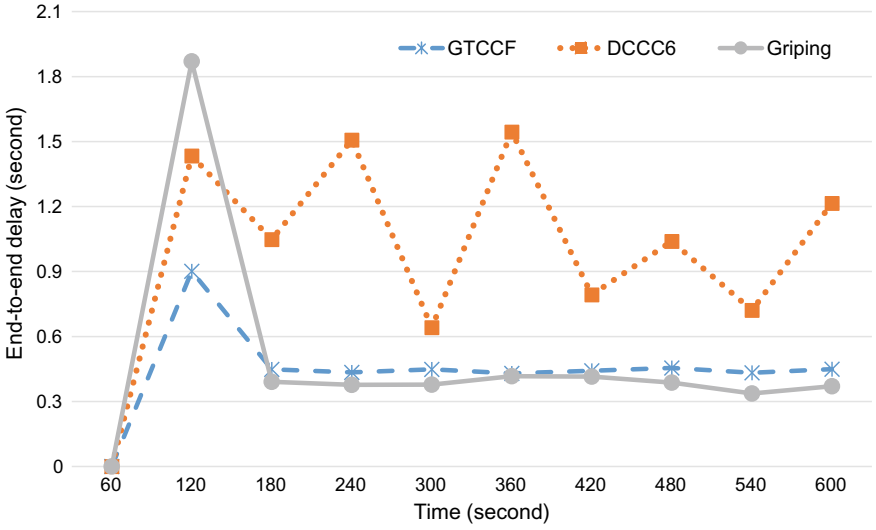

Fig. 5.6 End-to-end delay

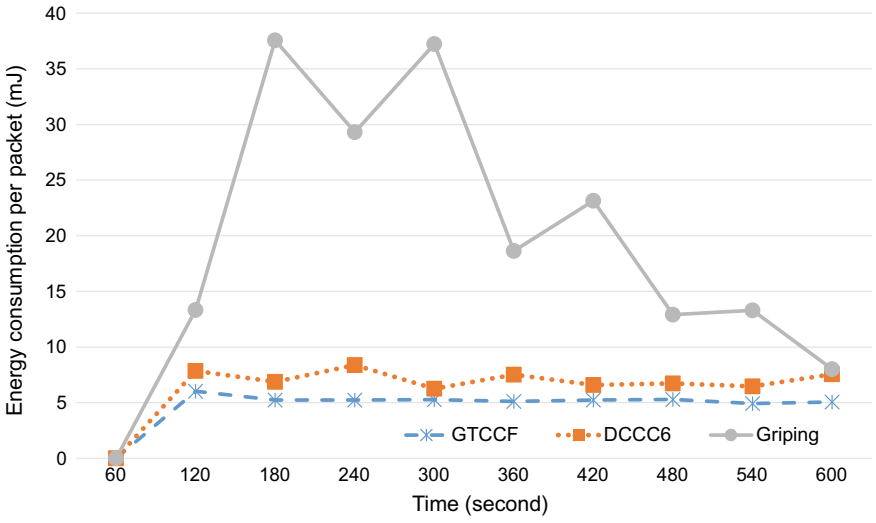

Fig. 5.7 Energy consumption per successful packet

(Nash equilibrium) are computed, the number of lost packets due to buffer overflow becomes zero. However, the number of lost packets in DCCC6 is higher than Griping algorithm as the sending rates are increased by a small step in Griping whereas by a large step in DCCC6.

Figure 5.9 shows the weighted fairness index (WFI) which is an indication of how much the nodes associated with a parent are treated fairly according to their priorities. We measure this performance metric to show and determine whether the

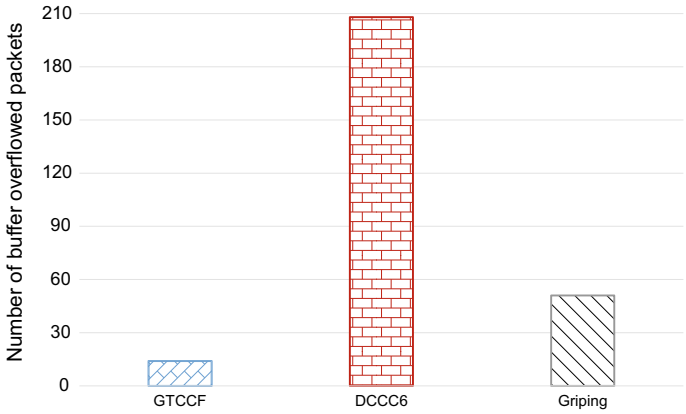

Fig. 5.8 Number of lost packets

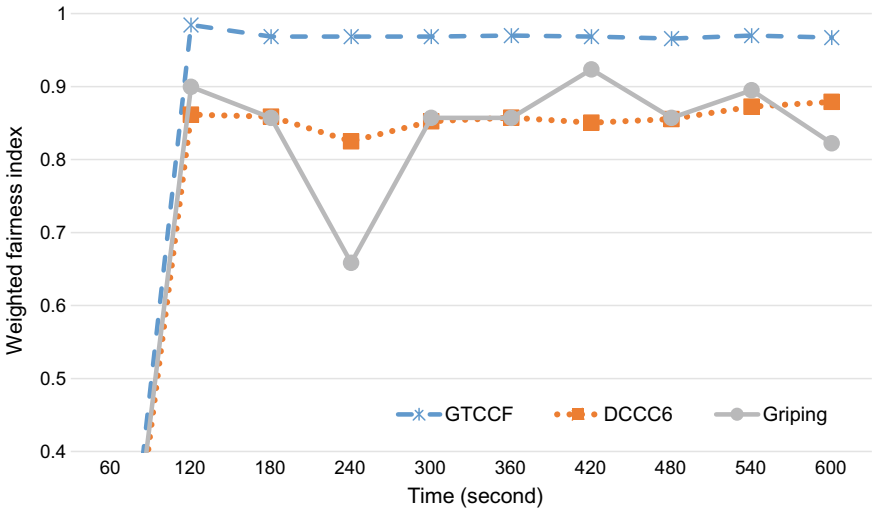

Fig. 5.9 Weighted fairness index in scenario 1

algorithms achieve a fair allocation of the network resources (i.e. throughput) among nodes. We have calculated this metric similar to that used in [18] as follows:

$$WFI = \frac{\left[\sum_{k=1}^{m} th_k p_k\right]^2}{m \sum_{k=1}^{m} (th_k p_k)^2}, \tag{5.25}$$

where th_k is throughput of leaf node L_k.

Table 5.2 Algorithms performance summarization in scenario 1

Performance metric	GTCCF	DCCC6	Griping
Throughput/L_1	1.459	0.690	0.068
Throughput/L_2	1.003	0.853	0.072
Throughput/L_3	0.751	0.698	0.062
Overall throughput	3.214	2.242	0.203
Delay/packet	0.493	1.104	0.549
Energy/packet	5.266	7.135	21.496
Lost packets/s	0.025	0.385	0.094
Average *WFI*	0.970	0.856	0.847

From this figure, it is clear that GTCCF achieves fairness index close to 1 which indicates for high fairness allocation of overall throughput among the leaf nodes based on their priorities. On the other hand, DCCC6 and Griping have lower WFI than GTCCF as they do not support awareness of node priorities.

Table 5.2 summarizes the performance of GTCCF, DCCC6 and Griping algorithms in the first scenario in terms of average number of received packets per second per leaf node (throughput/leaf), the total number of received packets per second (overall throughput), average end-to-end delay per packet in seconds (delay/packet), average energy consumption per successful delivered packet (energy/packet), average number of lost packets per second due to buffer overflow (lost packets/s) and average weighted fairness index (average WFI).

5.4.4 Scenario 2

In the second scenario, we use a multihop network with one sink node, 15 intermediate nodes and 5 leaf nodes distributed randomly (the network topology in this scenario is similar to the network topology in Fig. 5.1). L_1 and L_2 select an intermediate node (P_1) as their parent, L_2 and L_3 choose parent (P_2), whereas the node L_5 is associated alone with parent (P_3). We have set the priorities of nodes (L_1, L_2, L_3, L_4 and L_5) to $p_1 = 1$, $p_2 = 2$, $p_3 = 1$, $p_4 = 2$, and $p_5 = 2$, respectively. The node L_1 hosts three applications with priorities $p_1^1 = 1$, $p_1^2 = 2$ and $p_1^3 = 3$, the nodes L_2 and L_5 host two applications each with priorities $p_2^1 = p_5^2 = 1$ and $p_2^2 = p_5^1 = 2$, whereas L_3 and L_4 host one application each. From scenario 1, it is clear that Griping has the worst performance due to the rate adaptation mechanism used in Griping. Therefore, in this scenario, only GTCCF and DCCC6 are compared.

Figure 5.10 shows the number of received packets from each leaf node every second at the sink node. For GTCCF, the number of received packets from L_1 (≈ 1.1 packet/s) is higher than node L_2 (≈ 0.8 packet/s) as it has higher priority. Similarly, L_3 has higher number of received packets (≈ 0.3 packet/s) at sink than

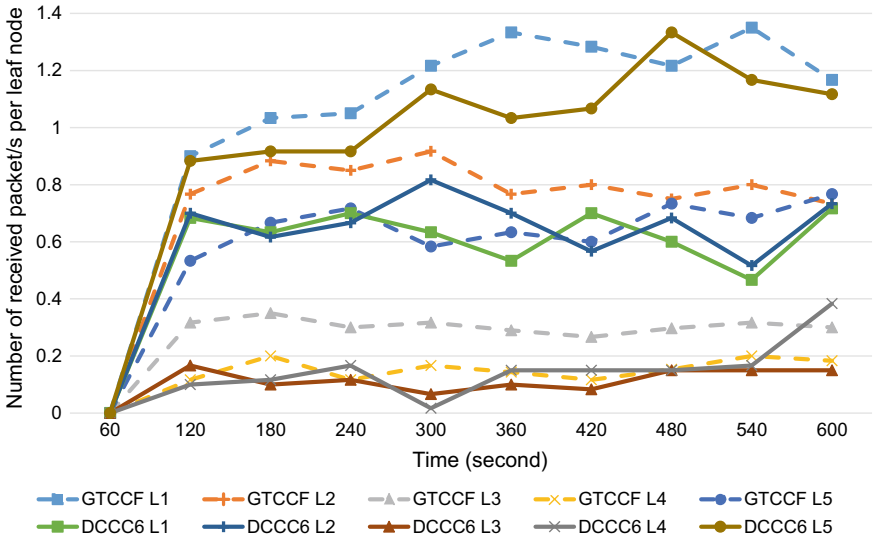

Fig. 5.10 Number of received packets/s from leaf nodes at sink

L_4 ($\approx$ 0.15 packet/s). On the other hand, for DCCC6, the number of received packets from node L_1 and L_2 is approximately the same ($\approx$ 0.6 packet/s) and from L_3 and L_4 is also the same ($\approx$ 0.1 packet/s). Also, from this figure, we can see that the number of received packets from nodes L_1 and L_2 is higher than nodes L_3 and L_4. The reason is that the forwarding rate of parent (P_1) is higher than parent (P_2) as P_1 is located nearer to the sink than P_2. Figure 5.11 shows overall throughput which is the total number of received packets at the sink every second. It is obvious that GTCCF has better throughput than DCCC6 for the same reasons stated in scenario 1. Figure 5.12 shows the sending rate (packet/second) for the applications hosted in the leaf nodes for GTCCF algorithm. It is clear that each leaf node distributes its sending rate among its applications according to their priorities. For example, the average sending rates of applications 1, 2 and 3 hosted in node L_1 are 0.488, 0.39 and 0.29 packet/s, respectively.

Figure 5.13 shows the end-to-end delay which is the time in second since a packet is generated at the leaf node until its arrival at the sink node. From this figure, it is obvious that GTCCF has lower end-to-end delay than DCCC6 algorithm for the same reasons stated in scenario 1. Figure 5.14 shows the energy consumption per successfully received packet (in mJoule) in the leaf and intermediate nodes due to packet transmission and reception. This figure shows that GTCCF consumes less energy as compared to DCCC6. Figure 5.15 shows the number of lost packets every second due to buffer overflow in each leaf node and intermediate node. It is clear

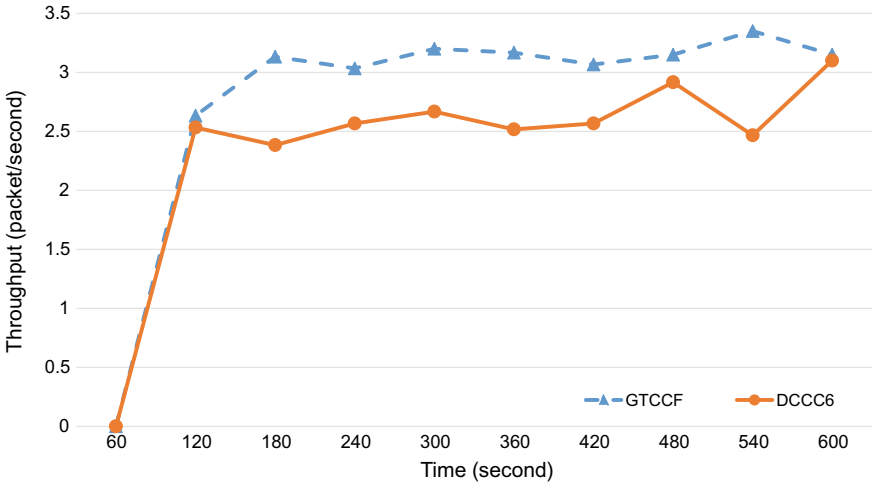

Fig. 5.11 Number of received packets/second at sink

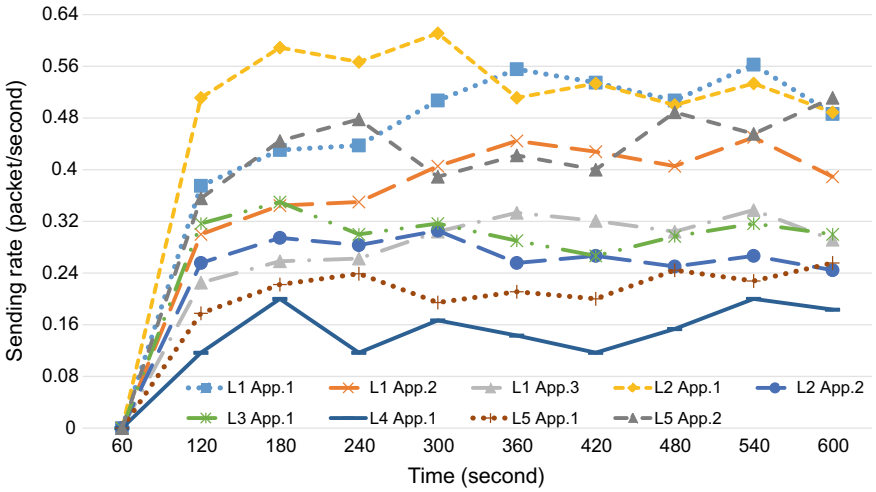

Fig. 5.12 Applications' sending rate for GTCCF

that the number of lost packets in GTCCF is lower than DCCC6 algorithm in both leaf nodes and intermediate nodes. Figure 5.16 shows the weighted fairness index for GTCCF and DCCC6. It is obvious that GTCCF has better fairness index than DCCC6 as it considers the priority of each leaf node in its operation. In general, Table 5.3 summarizes the overall performance of GTCCF and DCCC6 in scenario 2.

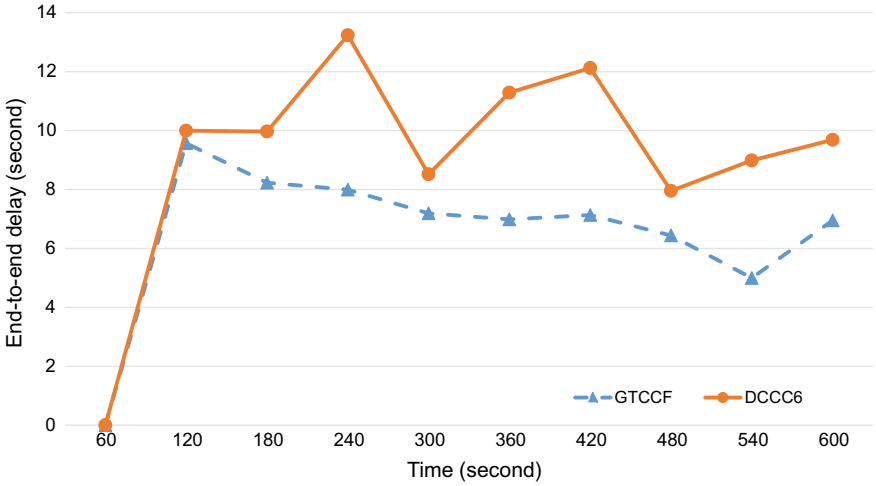

Fig. 5.13 End-to-end delay

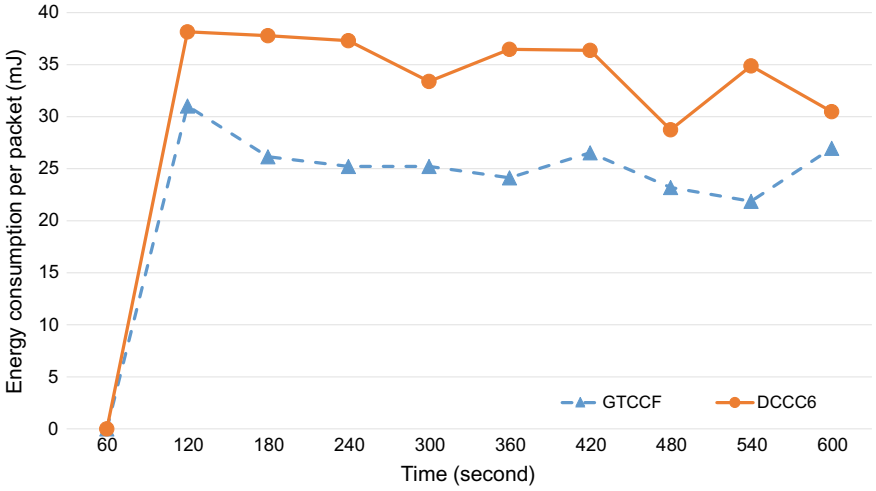

Fig. 5.14 Energy consumption per successful packet

Overall, based on the simulation results from scenario 1 and scenario 2, it is obvious that GTCCF and DCCC6 have better performance than Griping algorithm. Also, it is clear that GTCCF improves performance in terms of overall throughput, end-to-end delay, energy consumption, number of lost packets due to buffer overflow and average weighted fairness index by 30.45, 39.77, 26.37, 91.37 and 13.43%, respectively, as compared to DCCC6 algorithm.

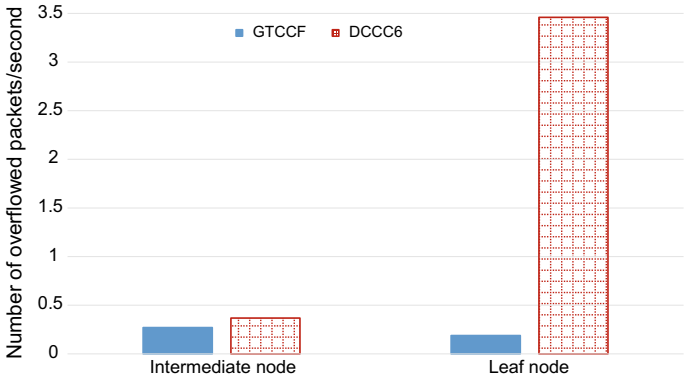

Fig. 5.15 Number of lost packets

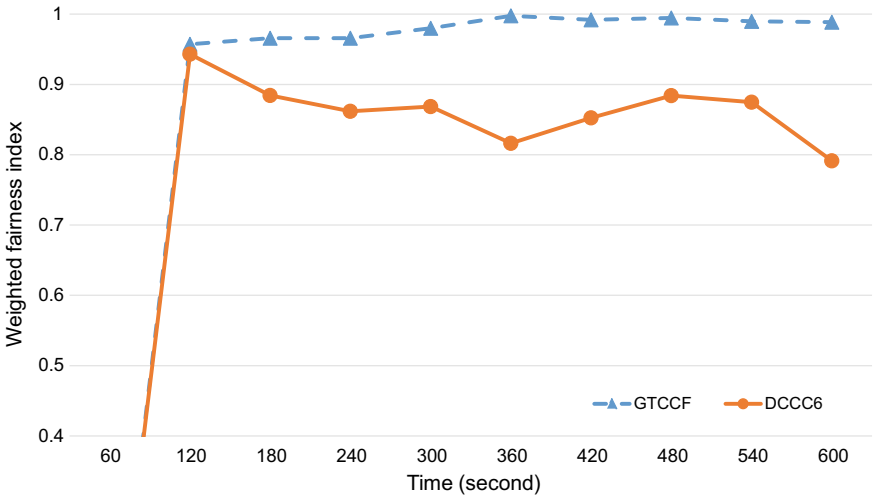

Fig. 5.16 Weighted fairness index in scenario 2

Table 5.3 Algorithms performance summarization in scenario 2

Performance metric	GTCCF	DCCC6
Throughput/L_1	1.172	0.629
Throughput/L_2	0.807	0.666
Throughput/L_3	0.305	0.120
Throughput/L_4	0.155	0.155
Throughput/L_5	0.657	1.062
Overall throughput	3.098	2.635
Delay/packet	7.276	10.195
Energy/packet	25.590	34.841
Lost packets/s	0.224	2.085
Average WFI	0.981	0.864

5.5 Conclusion

In this chapter, the congestion problem in 6LoWPAN networks is modelled as a game by using the non-cooperative game theory as well as the uniqueness of Nash equilibrium in the pure strategy space of the designed game is proved. Also, a new and simple congestion control mechanism called game theory based congestion control framework (GTCCF) is proposed. To support the IoT application requirements, the proposed framework is aware of node priorities and application priorities. Also, GTCCF is built and designed on the unique characteristics of IEEE 802.15.4, IPv6 and 6LoWPAN protocol stack. The proposed algorithm is evaluated in Contiki 3.0 OS under two scenarios and compared with other algorithms. Simulation results show that our proposal improves the QoS aspects, e.g. throughput, end-to-end delay, energy consumption, packet loss ratio and weighted fairness index as compared to existing algorithms.

References

1. Weldon M (2016) The future X network: a bell labs perspective. CRC Press
2. Ghaffari A (2015) Congestion control mechanisms in wireless sensor networks: a survey. J Netw Comput Appl 52:101–115
3. Kafi MA, Djenouri D, Ben-Othman J, Badache N (2014) Congestion control protocols in wireless sensor networks: a survey. IEEE Commun Surv Tutor 16(3):1369–1390
4. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805
5. Winter T, Thubert P, Brandt A, Hui J, Kelsey R (2012) RPL: IPv6 routing protocol for low-power and lossy networks. IETF, RFC 6550
6. Dunkels A, B. Grönvall B, Voigt T (2004) Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of 29th annual IEEE international conference on local computer networks. IEEE, pp 455–462
7. Osterlind E, Dunkels A, Eriksson J, Finne N, Voigt T (2006) Cross-Level sensor network simulation with COOJA. In: Proceedings of 31st IEEE conference on local computer networks. IEEE, pp 641–648
8. Wang L, Kuo G-S (2013) Mathematical modeling for network selection in heterogeneous wireless networks–a tutorial. IEEE Commun Surv Tutor 15(1):271–292
9. Nikaido H, Isoda K (1955) Note on noncooperative convex games. Pac J Math 5(5):807–815
10. Rosen JB (1965) Existence and uniqueness of equilibrium points for concave N-person games. Econ: J Econ Soc 520–534
11. Brown RG (2004) Smoothing, Forecasting and prediction of discrete time series. Courier Corporation
12. Michopoulos V, Guan L, Oikonomou G, Phillips I (2002) DCCC6: duty cycle-aware congestion control for 6LoWPAN networks. In: Proceedings of international conference on pervasive computing and communications workshops (PERCOM workshops). IEEE, pp 278–283
13. Castellani AP, Rossi M, Zorzi M (2014) Back pressure congestion control for CoAP/6LoWPAN networks. Ad Hoc Netw 18:71–84
14. Stehlık M (2011) Comparison of simulators for wireless sensor networks, Master's thesis, Faculty of informatics, Masaryk university, Brno, Czech Republic
15. Dunkels A, Eriksson J, Finne N, Tsiftes N (2011) Powertrace: network-level power profiling for low-power wireless networks. Technical report, Swedish Institute of Computer Science (SICS)

16. Rangwala S, Gummadi R, Govindan R, Psounis K (2006) Interference-aware fair rate control in wireless sensor networks. ACM SIGCOMM Comput Commun Rev 36(4):63–74
17. Michopoulos V (2012) Congestion and medium access control in 6LoWPAN WSN, Ph.D. dissertation, Computer science, Loughborough University
18. Zawodniok M, Jagannathan S (2007) Predictive congestion control protocol for wireless sensor networks. IEEE Trans Wirel Commun 6(11):3955–3963

Chapter 6
Optimization-Based Hybrid Congestion Alleviation

6.1 Introduction

In general, two main methods are used to solve and alleviate congestion in WSNs and 6LoWPAN networks: rate adaptation (traffic control) and traffic engineering, i.e. selection of an alternate non-congested path (resource control) to forward packets to destination nodes [1, 2]. In traffic control, the sending rate of the source node is reduced to a specific value such that the number of injected packets into the network is reduced and therefore, congestion is alleviated. However, for time-critical and delay-constrained application (e.g. medical applications and fire detection applications), reducing the data rate is not desirable and impractical. In the resource control method, packets are forwarded to destination node through alternative non-congested paths without adjusting the sending rate. However, sometimes non-congested paths are not available and therefore, congestion cannot be avoided. Thus, it is very important to combine the above two strategies into a hybrid scheme and utilizing the positive aspects of using both traffic control and resource control. In such case, the resource control strategy is firstly used for searching non-congested paths. If they are not available, then the sending rate is reduced by applying the traffic control strategy. To the best of our knowledge, no existing congestion control mechanism in 6LoWPAN networks combines both strategies to solve the congestion problem.

The RPL [3] is expected to be the standard routing protocol for 6LoWPAN networks and the IoT. In 6LoWPAN networks, RPL is responsible for constructing the network topology based on an objective function which combines one or more routing metrics into a Rank. Each node selects a neighbour as its parent with the best Rank. In case of congestion, the main challenge is that the node ranks the parents and paths from the least to the most congested and selects the best one when congestion occurs according to multiple routing metrics. Thus, the selection of a parent can be modelled as a multi-criteria decision problem which can be solved by using a multi-attribute decision-making (MADM) technique. MADM presents a suitable approach and promising solution for the parent selection problem

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_6

within congestion. However, sometimes a non-congested parent is not available and applying the traffic control strategy is important to mitigate and alleviate congestion in the network. When congestion occurs, each node starts to send high data rate packets to its parent without considering the parents forwarding rate, the available bandwidth and other nodes' sending rate. Therefore, adapting and allocating the sending rate to each node subject to congestion alleviation are important. The nodes' sending rate adaptation can be modelled as a constrained optimization problem which can be solved by using optimization theory [4]. Optimization theory provides the necessary tools and techniques that can adjust node sending rate optimally and satisfactorily. However, none of the existing congestion control algorithms in WSNs and 6LoWPAN networks utilizes and uses MADM and optimization theory to mitigate congestion in the network.

This work is motivated by these considerations to propose a novel congestion control algorithm called 'optimization-based hybrid congestion alleviation' (OHCA) which combines both traffic and resource control strategies into a hybrid solution to utilize the benefits of using both of them. Also, OHCA uses a multi-criteria optimization approach for selecting less congested parent and path to forward packets to the final destination as well as optimization theory for controlling and adapting nodes' sending rate when the non-congested parent is not available. Our main contributions in this chapter include:

- Proposal of a new congestion alleviation algorithm called OHCA which provides a hybrid solution to the congestion problem in 6LoWPAN networks to use and utilize the network resources effectively. The proposed algorithm first applies the resource control strategy which searches for the non-congested path by utilizing a MADM technique. If the resource control method cannot be applied, then the traffic control strategy is executed to reduce the number of injected packets into the network by using optimization theory. Thus, OHCA utilizes the advantages of both strategies by bridging these two methods for congestion control and providing the optimal solution.
- Model the selection of parents within congestion as a multi-criteria decision problem which can be solved by using the grey relational analysis (GRA) method [5]. GRA ranks the parents from least to most congested and selects the best one by combing a set of routing metrics (attributes). In our proposal, we use three routing attributes: expected transmission count (ETX), buffer occupancy (BO) and queue delay (QD). Thus, the GRA approach is integrated with the RPL objective function to make our proposal compatible with the 6LoWPAN protocol stack. The weights of routing metrics are calculated by using the standard deviation method.
- Model the nodes' sending rate adaptation as a constrained optimization problem which can be solved using network utility maximization (NUM) framework. Here, we utilize the NUM framework in 6LoWPAN networks to allocate data rate to each node when congestion occurs where each node has a utility function. The node's utility function is modelled as a constrained nonlinear optimization problem which is solved by using Lagrange multipliers and KKT conditions such that each node obtains its optimal solution (i.e. sending rate) that satisfies the congestion alleviation.

- In the IoT applications, sensor nodes host many application types simultaneously with different requirements. Some of them are real-time applications where data is important and time critical, while others are non-real-time applications. Therefore, it is important that a new proposed algorithm supports awareness of both node priorities and application priorities. Thus, our proposal (OHCA) is aware of node priorities and application priorities to support the IoT application requirements. How to allocate and adapt the applications' sending rate in an effective way based on their priorities is important. In this chapter, we model the 'applications' sending rate adaptation' problem as a constrained optimization problem by using the NUM framework where Lagrange multipliers and KKT conditions are used to compute the optimal application's sending rate. Furthermore, OHCA is designed and built on the unique characteristics of the IEEE 802.15.4 standard, IPv6 and 6LoWPAN protocol stack.
- Implement and evaluate the performance of the proposed algorithm in the real IoT operating system, Contiki OS [6], through Cooja simulator [7].

The remainder of the chapter is organized as follows: Sect. 6.2 introduces the network setup and formulates the problem. Section 6.3 introduces resource control strategy based on MADM. The traffic control strategy based on optimization theory and NUM framework is given in Sect. 6.4. The implementation of the hybrid congestion control algorithm in 6LoWPAN networks is provided in Sect. 6.5. In Sect. 6.6, simulation scenarios and results are given. Finally, Sect. 6.7 draws conclusions.

6.2 Network Setup and Problem Formulation

In 6LoWPAN networks, the RPL routing protocol [3] is responsible for constructing the network topology. Three types of nodes are defined: sink (root) nodes which provide connectivity to other networks, intermediate nodes which forward packets to the sink and leaf nodes. Consider a part of the network (dashed-line rectangle [A] in Fig. 6.1) where 5, 2 and 1 leaf nodes select node 1, node 2 and node 3, respectively, as their parents at the network topology construction stage. Under low data rate, the leaf nodes send packets to the sink through their parents successfully. However, when congestion does occur, the leaf nodes start to send heavy traffic packets to their parents. In this situation, node 1 forwards packets from 5 leaf nodes, whereas node 2 and node 3 forward packets from 2 and 1 leaf nodes, respectively. According to congestion analysis in [8], the majority of packets are lost due to buffer overflow when congestion occurs in 6LoWPAN network. Thus, a large number of packets are lost at node 1's buffer as its receiving rate from 5 leaf nodes is much higher than its forwarding rate. The default routing metrics specified in RFC 6551 [9] and de facto objective functions (ETX-OF [10] and OF0 [11]) do not reflect or are aware of congestion occurring. Hence, they do not distribute and balance the traffic load among parent nodes to reduce packet loss due to parents' buffer overflow (i.e. the leaf nodes do not change their current parent and select another less or

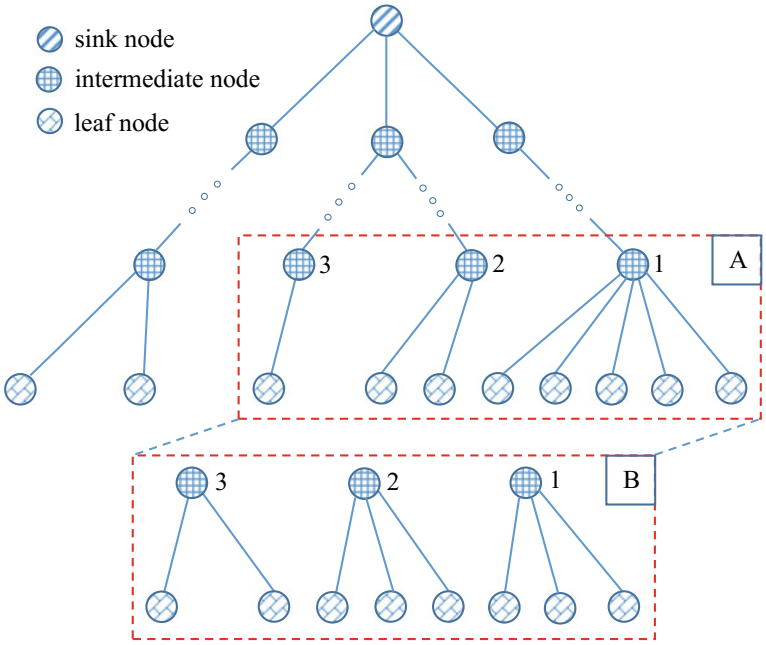

Fig. 6.1 Network topology based on RPL

non-congested one) as shown in dashed-line rectangle [B] in Fig. 6.1. The authors in [8, 12–14] also have demonstrated the problem of 'load balancing' or 'parent selection' within congestion in the RPL routing protocol. However, even with congestion-aware routing metrics and objective functions, sometimes a leaf node cannot find a less or non-congested parent and the incoming rate to the parent is higher than its outgoing rate. Therefore, according to Queuing theory [15], the parent's buffer starts overflowing the incoming packets and congestion still exists. Thus, it is very important to have a rate adaptation policy to reduce the number of sent packets and therefore congestion can be controlled in the network. In this chapter, we address both 'parent selection' and 'rate adaptation' problems and develop a hybrid solution to alleviate congestion in 6LoWPAN networks as shown in the next sections.

6.3 MADM-Based Resource Control

In RPL, the objective function, which is completely responsible for constructing the network topology, is separated from the core protocol specifications. This allows easy design and implementation of a new objective function that satisfies the application and network requirements. The objective function combines one or more routing metrics to produce a rank value which is advertised by a DIO control message. Here, we use and utilize a multi-criteria optimization approach to combine three routing

metrics and develop a new objective function called MADM-OF. The proposed objective function addresses and solves the 'parent selection' problem within congestion by selecting a less or non-congested parent node from the existing 'alternatives' or 'parents' by considering multiple 'attributes' or 'routing metrics'. In our proposal, we use GRA approach which is part of grey theory developed by Deng [16] and it has been successfully applied for solving different problems in various fields [17]. Before we describe the procedures of GRA methodology, we list and explain the routing metrics (attributes) used to find the best parent (alternative) in term of congestion. We use three routing metrics which reflect how much the nodes and network are congested: BO, ETX and queuing delay (QD).

One can use more routing metrics such as channel load (channel busyness ratio), packet loss and energy consumption [18]. But, as a sensor node has limited computation capability and to keep the calculation simple and straightforward, we use the above three metrics which are appropriate and reflect how much the node and wireless link are congested.

6.3.1 Grey Relational Analysis Procedure

Suppose a node (decision-maker) has a set of m candidate parents (alternatives) $A = \{a_i, i = 1, 2, \ldots, m\}$ with a set of 3 routing metrics (attributes) $R = \{r_j, j = 1, 2, 3\}$ for each parent and a weight vector $G = \{g_j, j = 1, 2, 3\}$ which represents the importance (weight) of the attributes. Then, the MADM parent selection problem can be represented by a decision matrix D as follows:

$$D = \begin{bmatrix} r_1(a_1) & r_2(a_1) & r_3(a_1) \\ r_1(a_2) & r_2(a_2) & r_3(a_2) \\ \vdots & \vdots & \vdots \\ r_1(a_m) & r_2(a_m) & r_3(a_m) \end{bmatrix}, \tag{6.1}$$

where $r_j(a_i)$ represents the value of jth routing metric (attribute) for the ith parent (alternative) for all $i = 1, 2, \ldots, m$ and $j = 1, 2, 3$. For our proposal, we have three routing metrics: BO, ETX and QD. Thus, $r_1 = \text{BO}$, $r_2 = \text{ETX}$ and $r_3 = \text{QD}$.

The procedure of GRA consists of four steps to generate the global comparison among the candidate parents as follows [5]:

1. *Grey Relational Generating (Normalization)*: as the unit of routing metrics are different (e.g. BO is measured in packets, while QD is measured in seconds), processing all values for every routing metric into a comparability sequence is necessary as follows:

$$x_{ij} = \frac{\max\limits_{\forall i}\{r_j(a_i)\} - r_j(a_i)}{\max\limits_{\forall i}\{r_j(a_i)\} - \min\limits_{\forall i}\{r_j(a_i)\}}, \tag{6.2}$$

where $x_{ij} \in [0, 1]$ is the normalized value of jth routing metric for the ith parent for all $i = 1, 2, \ldots, m$ and $j = 1, 2, 3$. In our MADM, all attributes (BO, ETX and QD) are cost. Equation (6.2) is used for cost attributes, while for benefit attributes, there is another equation (see Eq. (2) in [5]).

2. *Reference Sequence Definition*: the reference sequence is used to find the alternative (parent) whose comparability sequence is closet to the reference (preferred) sequence. In our MADM, if the value of x_{ij} is equal to 1 or nearer to 1, this means the performance of parent i is the best one for routing metric j. Thus, we define the reference sequence $x_{0j} = 1$ for all $j = 1, 2, 3$.
3. *Grey Relational Coefficient Calculation*: grey relational coefficient is used to determine how x_{ij} is close to x_{0j} and it can be calculated as follows:

$$\gamma(x_{ij}, x_{0j}) = \frac{\min\limits_{\forall i, \forall j}\{\Delta_{ij}\} + \zeta \max\limits_{\forall i, \forall j}\{\Delta_{ij}\}}{\Delta_{ij} + \zeta \max\limits_{\forall i, \forall j}\{\Delta_{ij}\}}, \tag{6.3}$$

where $\Delta_{ij} = |x_{0j} - x_{ij}|$ and $\zeta \in [0, 1]$ is the distinguishing coefficient for all $i = 1, 2, \ldots, m$ and $j = 1, 2, 3$.

4. *Grey Relational Grade Calculation*: after the grey relational coefficients $\gamma(x_{ij}, x_{0j})\ \forall i, \forall j$ are calculated, finally, the grey relational grade of parent (alternative) a_i for all $i = 1, 2, \ldots, m$ can be calculated as follows:

$$\Gamma(a_i) = \sum_{j=1}^{3} g_j \gamma(x_{ij}, x_{0j}), \tag{6.4}$$

where g_j is the weight of routing metric (attribute) j for all $j = 1, 2, 3$ such that $\sum_{j=1}^{3} g_j = 1$.

The grey relational grade is equivalent to the RPL objective function rank where a node selects a parent with largest grey relational grade which represents the best rank. The procedures to calculate the rank value is similar to the default RPL but with different methodology (Here we use GRA method). The advantages of GRA methodology are: (i) the results are based on the original data and (ii) the calculations are simple and straightforward where the 6LoWPAN mote has limited processing capability [19].

6.3.2 Routing Metric Weights Calculation

The weights g_1, g_2 and g_3 represent the importance of attributes (routing metrics) BO, ETX and QD, respectively. The weight of attributes plays an important role in the process of decision making where many methods have been proposed to determine the weights [20]. Here, we use the standard deviation (SD) method due to its simple

calculations as 6LoWPAN motes have constrained computational power. The SD method determines the weights in terms of their standard deviations as follows [20]:

$$g_j = \frac{\sigma_j}{\sum_{u=1}^{3} \sigma_u}, \tag{6.5}$$

$$\sigma_j = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (x_{ij} - \bar{x}_j)^2}, \tag{6.6}$$

$$\bar{x}_j = \frac{1}{m} \sum_{i=1}^{m} x_{ij}, \tag{6.7}$$

for all $j = 1, 2, 3$ where m is number of parents (alternatives).

6.4 Optimization-Based Traffic Control

The MADM-OF searches for non-congested parents to mitigate congestion by achieving traffic load balancing and distribution. On the other hand, sometimes, the non-congested parent is not available and congestion still exists. Thus, applying the traffic control strategy is important to reduce the number of injected packets and therefore congestion can be controlled and solved. Here, we utilize optimization theory to propose a new **T**raffic **C**ontrol mechanism called NUM-TC which adapts the source nodes' sending rate by using the NUM framework when the resource control strategy cannot be applied. Consider a parent node has a set of z children nodes, $L = \{N_l, l = 1, 2, \ldots, z\}$ which are competing to send data packets to sink through their parent. Also, we assume that: (i) Each node in the network has a buffer size of B packets, (ii) The children nodes have different priorities $P = \{p_1, p_2, \ldots, p_z\}$ where p_l is the priority of node N_l such that $p_l > 0$ for all $l = 1, 2, \ldots, z$. The priorities of children nodes are specified by user, based on the importance of node and the importance of the hosted applications, (iii) Each child node hosts a set of y applications $K = \{app^k; k = 1, 2, \ldots, y\}$ with different priorities, denoted by p_l^k to the priority of application app^k hosted in child node N_l such that $p_l^k > 0$ for all $l = 1, 2, \ldots, z$ and $k = 1, 2, \ldots, y$. The priorities of hosted applications are specified by user based on importance and type of application (i.e. real-time application, reliable application, etc.).

According to Queuing theory, congestion and buffer overflow occur when the incoming rate to a parent node (λ_{in}) from its children nodes is higher than its forwarding rate (λ_{out}). So, the problem is how to allocate the available parent's forwarding rate (λ_{out}) among the children nodes in an efficient manner such that congestion can be alleviated. The NUM framework can be used to model the 'sending rate

allocation' problem as a constrained optimization problem where a node N_l has a utility function $U_l(\lambda_l)$ and λ_l is the sending rate allocated to node N_l for all $l = 1, 2, \ldots, z$. Formally, the NUM problem can be expressed as follows [21]:

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{maximize}} \quad & \sum_{l=1}^{z} U_l(\lambda_l), \\ \text{subject to} \quad & \sum_{l=1}^{z} \lambda_l \leq \lambda_{out}, \\ & \lambda_l \geq 0, \qquad \forall l = 1, 2, \ldots, z, \end{aligned} \tag{6.8}$$

where $\boldsymbol{\lambda}$ is a vector consisting of $\lambda_1, \lambda_2, \ldots, \lambda_z$ and $\lambda_{out} > 0$.

Many types of utility function are commonly used such as exponential, logarithmic, linear and sigmoidal [22]. In our framework, we use the logarithmic utility function as it has strict concavity property. Also, different utility functions exist in term of fairness such as proportional fairness, weighted proportional fairness and max-min fairness [23]. We select the weighted proportional fairness to satisfy that each node obtains sending rate according to its priority. Thus, the utility function of node N_l can be expressed as follows:

$$U_l(\lambda_l) = \phi_l \log(\lambda_l), \tag{6.9}$$

where ϕ_l is the weight of node N_l's utility function such that $\phi_l > 0$ for all $l = 1, 2, \ldots, z$.

6.4.1 Optimal Sending Rate Computation

The proposed utility function $U_l(\lambda_l)$ is an increasing, strictly concave and continuously differentiable function of λ_l over $\lambda_l \geq 0$ for all $l = 1, 2, \ldots, z$. From classical optimization theory, the problem in Eq. (6.8) has a unique global maximum solution (point) [24, 25]. The problem in Eq. (6.8) can be solved using either centralized algorithms or decentralized algorithms. The centralized algorithms are very fast for finding an optimal solution in practice [26]; however, the main challenge is the overhead by exchanging congestion information among nodes in the network. The decentralized distributed algorithms decompose the original problem into subproblems (solved locally) and a master problem (e.g. primal decomposition and dual decomposition) to reduce information exchanged among nodes in the network [21, 27]. However, by using these algorithms, convergence to an optimal solution may require a long time and the solution in 6LoWPAN networks has to be fast and quick. Also, in our framework, the parent node can send congestion information in a simple way by sending a broadcast message. Now, since $\log(\lambda_l) \longrightarrow -\infty$ as $\lambda_l \longrightarrow 0$, the optimal sending rate (solution) will assign a strictly positive rate to

each node, and so the last constraint can be ignored [21]. Thus, in order to solve the problem in Eq. (6.8) without decomposing, we introduce the Lagrange multiplier v and define the Lagrangian function $\mathcal{L}(\boldsymbol{\lambda}, v)$ for all $l = 1, 2, \ldots, z$ as follows:

$$\mathcal{L}(\boldsymbol{\lambda}, v) = \sum_{l=1}^{z} U_l(\lambda_l) + v\left(\lambda_{out} - \sum_{l=1}^{z} \lambda_l\right), \tag{6.10}$$

where the KKT conditions for optimality are as follows:

$$\begin{gathered} v \geq 0, \\ \lambda_{out} - \sum_{l=1}^{z} \lambda_l \geq 0, \\ \nabla \sum_{l=1}^{z} U_l(\lambda_l) + v\nabla\left(\lambda_{out} - \sum_{l=1}^{z} \lambda_l\right) = 0, \\ v\left(\lambda_{out} - \sum_{l=1}^{z} \lambda_l\right) = 0. \end{gathered} \tag{6.11}$$

Then, the optimal sending rate of node N_l after solving the problem in Eq. (6.8) is as follows:

$$\lambda_l = \frac{\phi_l \lambda_{out}}{\sum_{c=1}^{z} \phi_c}. \tag{6.12}$$

6.4.2 *Allocation of Node's Sending Rate Among Its Applications*

In the IoT applications, a sensor node does not host a single application as in the traditional WSNs. However, it hosts many applications with different requirements. Some of them are real-time applications where data is time critical, while others are non-real time applications. Therefore, it is important for each node to be aware of the priorities of the hosted applications. Consider a node hosts a set of y applications $K = \{app^k; k = 1, 2, \ldots, y\}$ with different priorities competing to send data packets through the node as shown in Fig. 6.2. We denote by p_l^k to the priority of application app^k hosted in node N_l for all $l = 1, 2, \ldots, z$ and $k = 1, 2, \ldots, y$. To allocate the node's sending rate (λ_l) fairly among its applications according to their priorities and prevent buffer overflow to occur inside the node (i.e. internal congestion), we can model the "application sending rate allocation" problem as a constrained optimization problem by using the NUM framework. In the NUM framework, an application, app^k, has a utility function $U^k(\lambda_l^k)$ where λ_l^k is the sending rate allocated to application app^k hosted in node N_l for all $k = 1, 2, \ldots, y$ and $l = 1, 2, \ldots, z$. It can be expressed as follows:

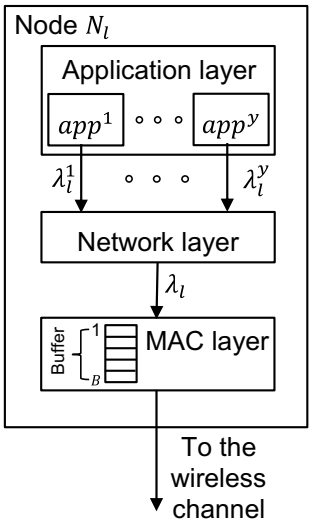

Fig. 6.2 Node model

$$\begin{aligned} \underset{\lambda_l^1,\lambda_l^2,\ldots,\lambda_l^k}{\text{maximize}} \quad & \sum_{k=1}^{y} U^k(\lambda_l^k), \\ \text{subject to} \quad & \sum_{k=1}^{y} \lambda_l^k \leq \lambda_l, \\ & \lambda_l^k \geq 0, \qquad \forall k = 1, 2, \ldots, y. \end{aligned} \tag{6.13}$$

We use the logarithmic, weighted proportional fairness utility function such that each application obtains a sending rate according to its priority as follows:

$$U^k(\lambda_l^k) = \phi^k \log(\lambda_l^k), \tag{6.14}$$

where ϕ^k is the weight of application app^k's utility function such that $\phi^k > 0$ for all $k = 1, 2, \ldots, y$.

To solve the problem in Eq. (6.13), following the same procedures used to solve the problem in Eq. (6.8) and the optimal sending rate of application, app_l^k, is as follows:

$$\lambda_l^k = \frac{\phi^k \lambda_l}{\sum_{d=1}^{y} \phi^d}. \tag{6.15}$$

We note that the solutions in Eqs. (6.12) and (6.15) associate with each node N_l's sending rate and each application app^k's sending rate a weight value (w_l and w^k) to obtain a weighted proportional fairness among nodes and applications. The optimal sending rate of each node N_l and each application app^k depends on the weight

of node N_l's utility function and application app^k's utility function respectively. As the value of weight is high, the node N_l and application app^k get higher sending rates.

With regards to the values of ϕ_l and ϕ^k, if a node (an application) with higher p_l (p_l^k) value has high priority (e.g. if $p_i = 1$ and $p_j = 2$, this means that node N_j has higher priority than node N_i), then $\phi_l = p_l$ ($\phi^k = p_l^k$). On the other hand, if a node (an application) with a lower p_l (p_l^k) value has high priority, then $\phi_l = 1/p_l$ ($\phi^k = 1/p_l^k$).

6.5 Hybrid Congestion Alleviation Algorithm Implementation

The OHCA algorithm is designed to use the network resources effectively and utilize positive aspects of using both resource and traffic control strategies. According to Queuing theory, if the arrival rate (λ_{in}) at a parent's buffer is higher than the service rate (λ_{out}), the parent's buffer will overflow and congestion will occur. Thus, the parent node periodically checks the congestion condition ($\lambda_{in} > \lambda_{out}$) every interval time '$I_{check}$'. If the parent node encounters congestion, it broadcasts a DIO message, which contains congestion information, to its children. When a child node receives the DIO message, it first applies the resource control strategy by using MADM-OF to select a non-congested parent and subsequently forwards packets through it. MADM-OF combines three metrics (BO, ETX and QD) to produce a Rank value such that a candidate parent with the best rank becomes selected as the current parent. To compute and accurately estimate the value of these metrics, we use Brown's simple exponential smoothing model [28] as follows:

$$r_j(t+1) = \psi_j r_j(t) + (1-\psi_j) r_j(t-1), \tag{6.16}$$

where $r_j(t+1)$, $r_j(t)$ and $r_j(t-1)$ are the expected, current and historic values of metric j respectively for $j = 1, 2, 3$ and ψ_j is smoothing factor of metric j such that $0 < \psi_j < 1$. A large value of ψ_j reduces the level of smoothing and gives high weight to current measurement of r_j, while a value of ψ_j close to zero gives greater smoothing effect and less responsive to recent changes in r_j value. Similarity, the forwarding rate of parent λ_{out} is not constant with time. It is increased or decreased due to the operation of the CSMA algorithm (i.e. backoff time), MAC parameters (i.e. channel check rate) and number of active nodes. Thus, to avoid sending high overhead DIO packets, we use Brown's simple exponential smoothing model to estimate the actual maximum service rate as follows:

$$\lambda_{out}(t+1) = \psi \lambda_{out}(t) + (1-\psi) \lambda_{out}(t-1), \tag{6.17}$$

where $\lambda_{out}(t+1)$, $\lambda_{out}(t)$ and $\lambda_{out}(t-1)$ are the expected, current and historic forwarding rate of the parent respectively and ψ is smoothing factor such that $0 < \psi < 1$. Equations (6.16) and (6.17) are updated on a per incoming packet basis.

On the other hand, if the child node cannot find a non-congested parent node, it applies the traffic control strategy by using the NUM-TC mechanism. First, the child node selects the less congested parent from the candidate parents. Then, it adjusts its sending rate based on Eq. (6.12) and congestion information received from the selected parent. After that, the child node allocates its updated sending rate among the hosted applications according to their priorities as in Eq. (6.15). Lastly, the network topology is governed by RPL through transmission of DIO, DAO and DIS control messages. The DIO transmission strategy is controlled by the Trickle algorithm. However, the Trickle algorithm is not aware of the occurrence of congestion. Therefore, the operation of the algorithm is modified such that when congestion occurs, the timer is reset to I_{min}.

6.6 Performance Evaluation

The proposed algorithm has been tested and evaluated on different network scenarios through simulation using Contiki 3.0 OS and Cooja simulator. In the first scenario, we use a network topology of one sink node, 5 intermediate nodes and 4 leaf (source) nodes with node ID of N_4, N_5, N_6 and N_7 (as illustrated in Fig. 6.3). In the second scenario, we use a network of one sink node, 18 intermediate nodes and 6 source nodes with node ID of N_{20}, N_{21}, N_{22}, N_{23}, N_{24} and N_{25}. Also, our proposal is compared with a traffic control based algorithm (DCCC6 [29]) and a resource control based algorithm (QU-RPL [12, 30]). In the simulation, the source nodes start sending packets at high data rate (6 packets/s) to create a congested situation. During the simulation, the source nodes start sending packets after 60 s so the network topology construction is completed, the simulation time is set to 600 s. Cooja simulates the hardware of a set of real sensor nodes, such as Tmote Sky, which is used in the

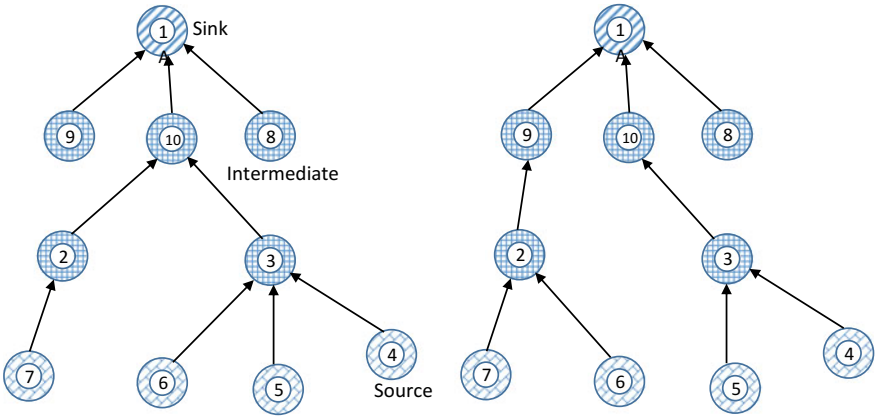

Fig. 6.3 Network topology in scenario 1 (left) DCCC6 (right) OHCA and QU-RPL

Table 6.1 Protocol stack and simulation parameters

Layer	Protocol	Parameter value
Application	Every leaf node sends high data rate packets to sink	Application payload = 30 bytes
Transport	UDP	
Network	uIPv6 + RPL	OF = MADM-OF (OHCA) OF = OF0 (DCCC6) OF = QU-OF (QU-RPL)
Adaptation	SICSlowpan layer	Compression method = HC06
Data link	CSMA (MAC layer) ContikiMAC (RDC layer) 802.15.4 (framer)	Buffer size = 8 packets MAC reliability (ACK) = enabled MAC max. retransmission = 3 Channel check rate = 8 Hz Max. frame size = 127 bytes
Physical	CC2420 RF transceiver	

simulation. Also, Cooja simulator implements a number of wireless channel models such as unit disk graph medium (UDGM)—distance loss, which is used in the simulation. We use Powertrace [31] to measure the energy consumption of each node where it is a runtime network-level power profiling system that uses state tracking to estimate the energy consumption and it is accurate up to 94%. The protocol stack and simulation parameters used in the simulation are shown in Table 6.1. We assume that a node (an application) with a higher value of priority ($p_l(p_l^k)$) has high priority. In the first scenario, we have set priorities of N_4, N_5, N_6 and N_7 to 2, 1, 1 and 2, respectively, where they host two, one, two and three applications, respectively, with priorities $p_4^1 = p_6^2 = p_7^1 = 1$, $p_4^2 = p_6^1 = p_7^2 = 2$ and $p_7^3 = 3$. In the second scenario, we have set priorities of N_{20}, N_{21}, N_{22}, N_{23}, N_{24} and N_{25} to 2, 1, 2, 2, 1, and 2, respectively where they host two, one, two, two, one and two applications respectively with priorities $p_{20}^1 = p_{22}^2 = p_{23}^1 = p_{25}^2 = 1$ and $p_{20}^2 = p_{22}^1 = p_{23}^2 = p_{25}^1 = 2$. For our proposal, we have set $I_{check} = 384$ clock ticks and $\psi = \psi_j = 0.4$; $\forall j = 1, 2, 3$ where 128 clock ticks = 1 s.

Next, we compare OHCA, DCCC6 and QU-RPL in terms of network topology layout, overall throughput, average throughput per node, applications' sending rate, weighted fairness index, end-to-end delay, energy consumption and lost packets due to buffer overflow. We have computed the average value of results obtained from scenario 1 and scenario 2 as follows.

6.6.1 Network Topology

Figure 6.3 shows the routing topology for OHCA, DCCC6 and QU-RPL algorithms in scenario 1. At the topology construction stage, nodes 2 and 3 select node 10 as their parent and nodes 4, 5 and 6 select node 3 as their parent, while node 2 is selected as

parent by node 7. When congestion occurs, many packets overflow buffers of nodes 3 and 10. As DCCC6 does not consider the load balancing problem with RPL and is not aware of buffer overflow, nodes do not change their parents and select less congested ones. In contrast, with OHCA and QU-RPL algorithms, node 2 changes its current congested parent, node 10, and selects less congested parent which is node 9. Also, node 6 changes its forwarding parent from node 3 to node 2. The reason is that OHCA and QU-RPL are aware of buffer overflow and congestion at nodes and they consider the load balancing problem in the routing protocol by using MADM-OF and QU-OF respectively. Similarly, in scenario 2, nodes forward packets through less congested parents in OHCA and QU-RPL, while DCCC6 does not consider the parent selection problem within congestion in RPL.

6.6.2 Throughput

Figure 6.4 shows the overall throughput which is the total number of received packets every second at the sink node. It is clear that OHCA has higher throughput (≈2 packet/s) than DCCC6 (≈1.5 packet/s) and QU-RPL (≈1.7 packet/s). The reason is that OHCA forwards packets through less congested nodes by using MADM-OF as well as adapting the sending rate of nodes by using NUM-CC framework when buffer drops still occur. Therefore, the number of forwarded packets to the sink node increases by exploiting the available network resources in an effective manner. Also, from this figure, QU-RPL is seen to have better performance in term of throughput as compared to DCCC6. The reason is that QU-RPL utilizes the available non-congested nodes and therefore, packets forwarded to the sink node increase. While DCCC6 does

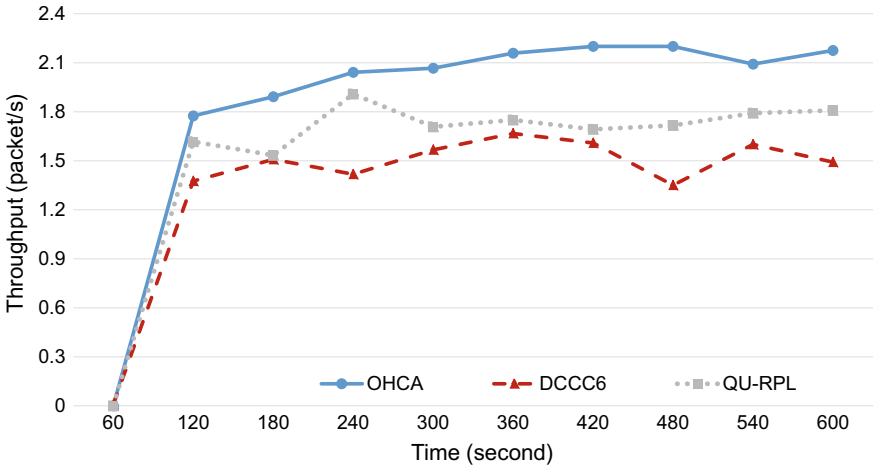

Fig. 6.4 Throughput

not utilize the available network resources (non-congested nodes) and it only adapts the nodes' sending rate by using a modified AIMD policy and therefore throughput decreases.

6.6.3 Throughput per Node

Figures 6.5 and 6.6 show the average number of received packets every second from the source nodes at sink in scenario 1 and scenario 2, respectively. From these figures, it is clear that nodes in OHCA obtain throughput according to their priorities. For instance, with OHCA in scenario 1, N_4 and N_7 have the highest number of received packets (≈0.53 and ≈0.58 packet/s), respectively. While, nodes N_5 and N_6 have the lowest throughput (≈0.34 and ≈0.36 packet/s), respectively, as they have low priorities as compared to other nodes. The reason is that OHCA is aware of node priorities where each node gets sending rate according to its priority. On the other hand, the nodes in DCCC6 and QU-RPL do not obtain a sending rate based on their priorities as these algorithms do not support awareness of node priorities. For example, with DCCC6 in scenario 1, node N_4 with higher priority has a lower number of received packets at sink (≈0.19 packet/s) as compared to node N_5 (≈0.26 packet/s) which has low priority. Similarity, in QU-RPL, node (N_7) with higher priority has lower throughput (≈0.43 packet/s) than node N_6 (≈0.54 packet/s) which has low priority.

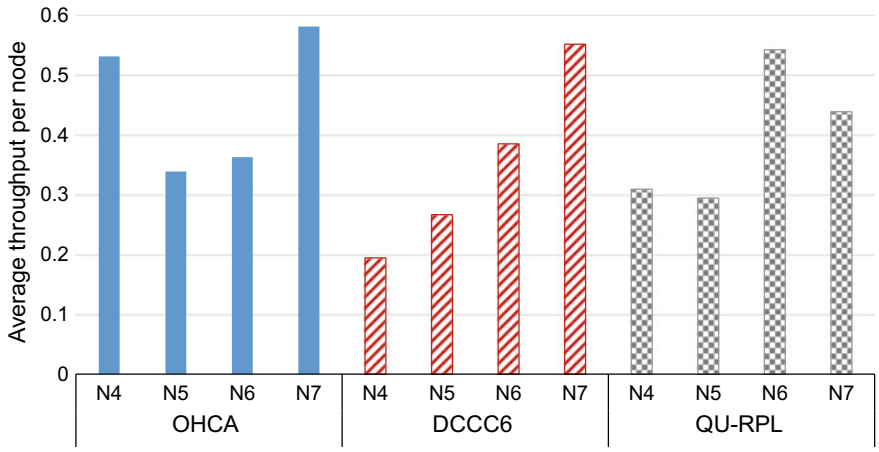

Fig. 6.5 Received packets/s from nodes in scenario 1

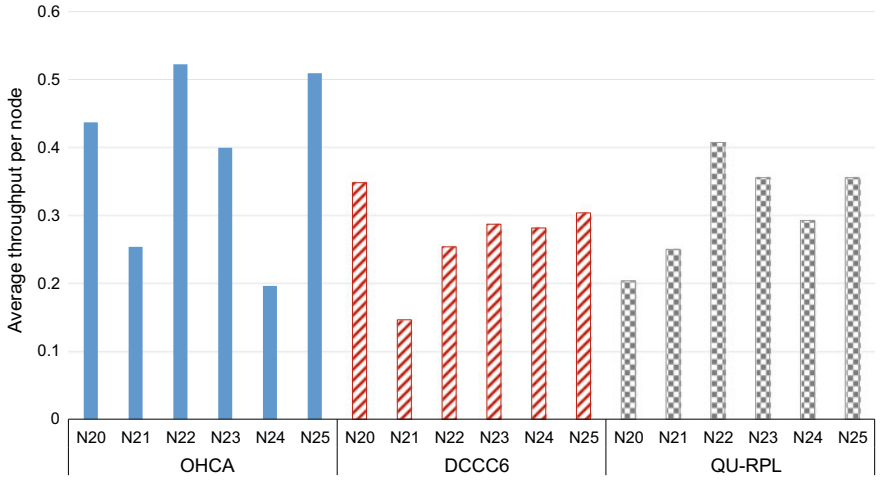

Fig. 6.6 Received packets/s from nodes in scenario 2

6.6.4 Applications' Sending Rate

Figures 6.7 and 6.8 show the average sending rate of applications (packet/s) hosted in the source nodes for OHCA in scenario 1 and scenario 2, respectively. Each application obtains the sending rate according to its priority. For example, in scenario 1, application app^1 in node N_4 obtains low sending rate ($\approx$0.17 packet/s) as compared to application app^2 ($\approx$0.35 packet/s) which has higher priority, similarity for nodes N_6 and N_7. While, in scenario 1, the application hosted in node N_5 gets sending rate equal to N_5's sending rate as it is hosted alone. In contrast, other algorithms do not support multiple applications hosted in each sensor node and they are not aware of application priorities.

6.6.5 Weighted Fairness Index

Figure 6.9 shows the weighted fairness index (WFI) which is an indication of how much the nodes associated with a parent are treated fairly according to their priorities. We have calculated this metric similar to that used in [32] as follows:

$$WFI = \frac{\left[\sum_{l=1}^{z}\left(\frac{th_l}{\phi_l}\right)\right]^2}{z\sum_{l=1}^{z}\left(\frac{th_l}{\phi_l}\right)^2}, \tag{6.18}$$

where th_l is throughput of node N_l.

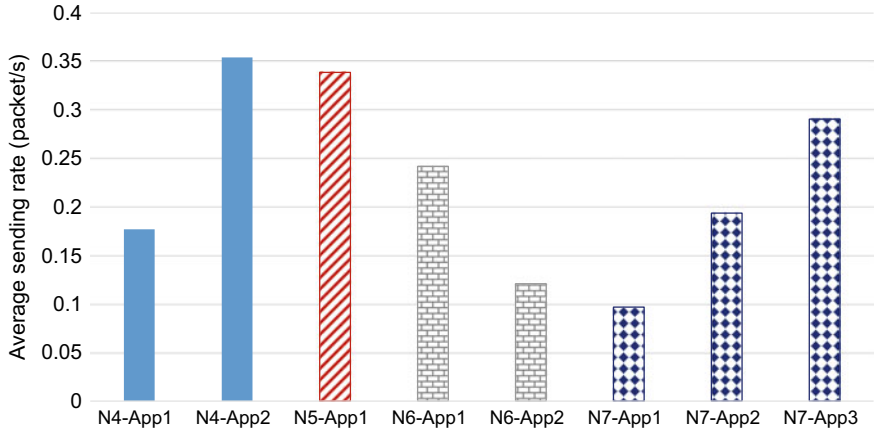

Fig. 6.7 Applications' rate of OHCA in scenario 1

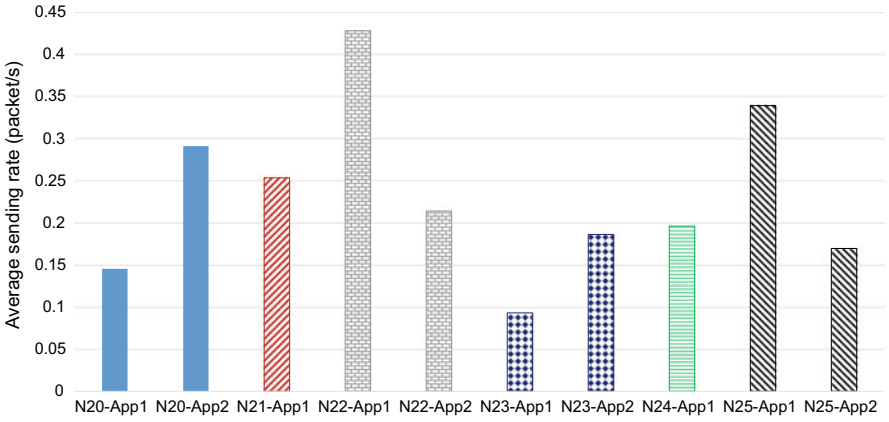

Fig. 6.8 Applications' rate of OHCA in scenario 2

From this figure, it is clear that OHCA achieves fairness index close to 1 (≈0.97) which indicates a high fairness allocation of overall throughput among the source nodes based on their priorities. On the other hand, DCCC6 and QU-RPL have lower WFI (≈0.89 and ≈0.66 respectively) than OHCA as they do not support awareness of node priorities.

6.6.6 End-to-End Delay

Figure 6.10 shows end-to-end delay which is the time between a packet being generated at the application of the source until its successful reception at the application

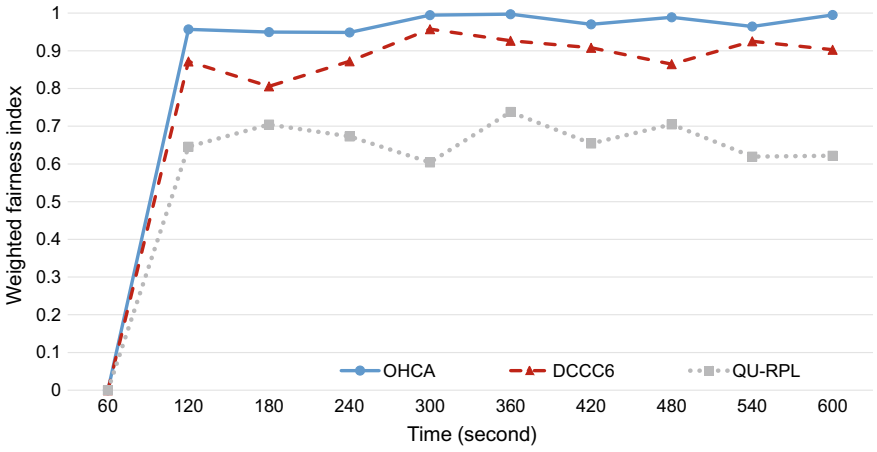

Fig. 6.9 Weighted fairness index

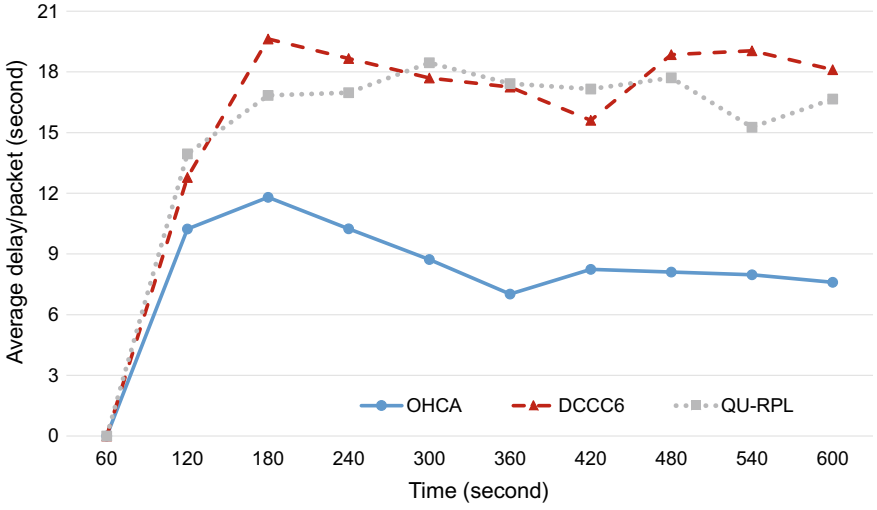

Fig. 6.10 End-to-end delay

of the final destination. OHCA has lower end-to-end delay as compared to DCCC6 and QU-RPL. The reason is that OHCA firstly searches for a non-congested parent to forward packets and if congestion still exists, then the number of injected packets into the network is reduced by reducing the nodes' sending rates. Therefore, buffer overflow is removed and packets do not a wait long time in the buffer. On the other hand, DCCC6 has high delay because of the modified AIMD mechanism used where the nodes' sending rates are increased periodically and decreased when congestion occurs and then this process continues. As a result, the packets wait a long time in the

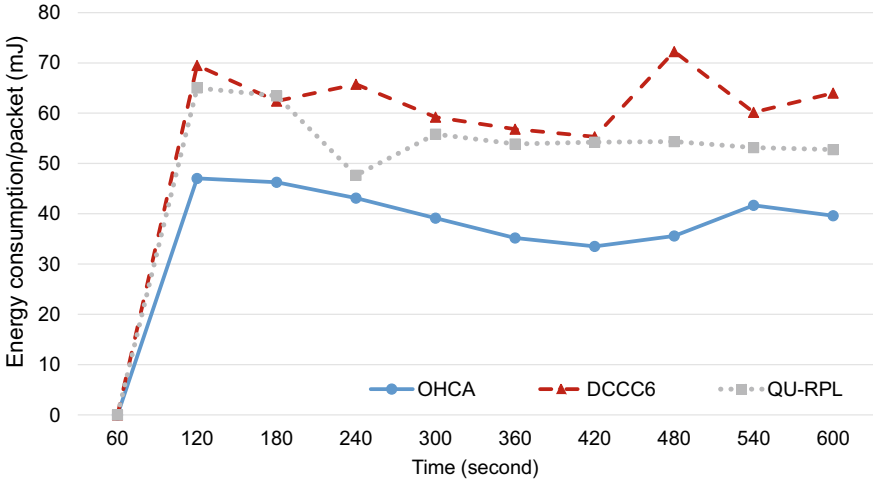

Fig. 6.11 Energy consumption per successful packet

nodes' buffers. Although QU-RPL forwards packets through less congested paths, it does not have a policy to reduce the nodes' sending rates when buffer drops still occur. Consequently, packets experience a long end-to-end delay if buffers are full most the time.

6.6.7 Energy Consumption

Figure 6.11 shows the energy consumption due to transmission and reception in the source and intermediate nodes per successfully delivered packet. We note that with OHCA, the energy consumption in the network is less than others as DCCC6 and QU-RPL waste energy by transmitting and receiving packets which are then lost due to buffer overflow on the path without successful delivery.

6.6.8 Lost Packets

Figure 6.12 shows the total number of lost packets every second in the network due to buffer overflow and due to wireless channel loss. It is obvious that OHCA loses less packets at the buffer than others for reasons stated above. However, the number of lost packets in DCCC6 and QU-RPL is higher than OHCA algorithm as DCCC6 uses the modified AIMD policy and QU-RPL does not have a sending rate adaptation mechanism. From this figure, the number of buffer overflowed packets per second for OHCA, DCCC6 and QU-RPL are 2.47, 25.41 and 25.91, respectively. Also,

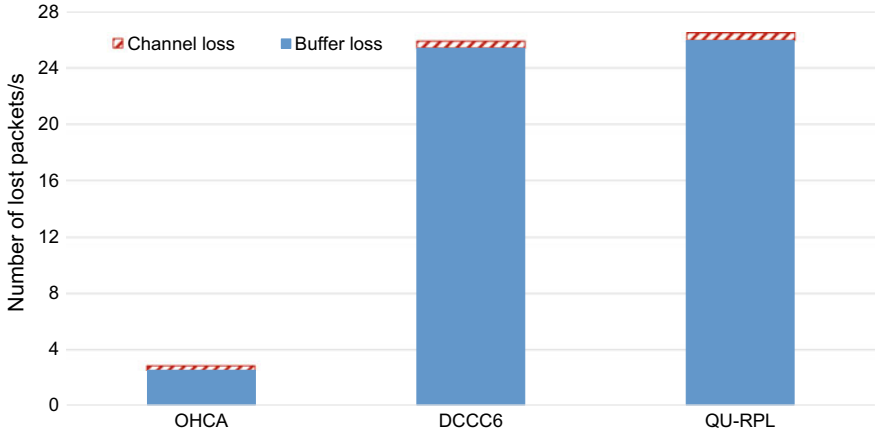

Fig. 6.12 Number of lost packets

the number of lost packets due to channel loss per second for OHCA, DCCC6 and QU-RPL are 0.37, 0.49 and 0.54, respectively.

Overall, based on the simulation results, it is clear that OHCA has superior performance than DCCC6 and QU-RPL algorithms. Also, it is clear that OHCA improves performance in terms of overall throughput, average weighted fairness index, end-to-end delay, energy consumption and number of lost packets due to buffer overflow by an overall average of more than 28.36%, 28.02%, 48.07%, 31.97% and 90.35%, respectively, compared to DCCC6 and QU-RPL schemes.

6.7 Conclusion

In this chapter, the congestion problem in 6LoWPAN networks is addressed by using a hybrid solution which combines traffic control and resource control strategies. We have modelled the "parent selection" problem as a MADM problem which is solved by using grey relational analysis methodology for achieving traffic load distribution in the presence of congestion and forwarding packets through non-congested parents. Also, we have modelled the 'nodes' sending rate adaptation' and 'applications' sending rate allocation' as constrained optimization problems by using optimization theory and the NUM framework. The optimal sending rates of nodes and applications are computed by using Lagrange multipliers and KTT conditions. Based on the MADM and NUM frameworks, we propose a new congestion control algorithm called optimization-based hybrid congestion alleviation (OHCA) which utilizes the advantages of using both traffic and resource control strategies and uses the network resources effectively. To support the IoT application requirements, OHCA is aware of node priorities and application priorities as well as being designed for the unique

characteristics of IEEE 802.15.4, IPv6 and 6LoWPAN. The proposed algorithm has been evaluated in Contiki 3.0 OS and compared with other algorithms. Simulation results show that OHCA improves the QoS parameters, i.e. throughput, weighted fairness index, end-to-end delay, energy consumption and packet loss.

References

1. Ghaffari A (2015) Congestion control mechanisms in wireless sensor networks: a survey. J Netw Comput Appl 52:101–115
2. Kafi MA, Djenouri D, Ben-Othman J, Badache N (2014) Congestion control protocols in wireless sensor networks: a survey. IEEE Commun Surv Tutor 16(3):1369–1390
3. Winter T, Thubert P, Brandt A, Hui J, Kelsey R (2012) RPL: IPv6 routing protocol for low-power and lossy networks. IETF, RFC 6550
4. Kelly FP, Maulloo AK, Tan DK (1998) Rate control for communication networks: shadow prices, proportional fairness and stability. J Oper Res Soc 49(3):237–252
5. Kuo Y, Yang T, Huang G-W (2008) The use of grey relational analysis in solving multiple attribute decision-making problems. Comput Ind Eng 55(1):80–93
6. Dunkels A, Grönvall B, Voigt T (2004) Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of 29th annual IEEE international conference on local computer networks. IEEE, pp 455–462
7. Osterlind F, Dunkels A, Eriksson J, Finne N, Voigt T (2006) Cross-level sensor network simulation with COOJA. In: Proceedings of 31st IEEE conference on local computer networks. IEEE, pp 641–648
8. Al-Kashoash HAA, Al-Nidawi Y, Kemp AH (2016) Congestion-aware RPL for 6LoWPAN networks. In: Proceedings of wireless telecommunications symposium (WTS 2016). IEEE, pp 1–6
9. Vasseur J-P, Kim M, Pister K, Dejean N, Barthel D (2012) Routing metrics used for path calculation in low-power and lossy networks. RFC 6551
10. Gnawali O, Levis P (2010) The ETX objective function for RPL. Internet draft: draft-gnawali-roll-etxof-00
11. Thubert P (2012) Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552
12. Kim H-S, Paek J, Bahk S (2015) QU-RPL: queue utilization based RPL for load balancing in large scale industrial applications. In: Proceedings of 12th annual IEEE international conference on sensing, communication, and networking (SECON). IEEE, pp 265–273
13. Liu X, Guo J, Bhatti G, Orlik P, Parsons K (2013) Load balanced routing for low power and lossy networks. In: Proceedings of wireless communications and networking conference (WCNC). IEEE, pp 2238–2243
14. Guo J, Liu X, Bhatti G, Orlik P, Parsons K (2013) Load balanced routing for low power and lossy networks, 21 January 2013, US Patent Application 13/746,173
15. Sztrik J (2012) Basic queueing theory. University of Debrecen, Faculty of Informatics
16. Ju-Long D (1982) Control problems of grey systems. Syst Control Lett 1(5):288–294
17. Liu S, Forrest JYL (2010) Grey systems: theory and applications. Springer, Berlin
18. Kafi MA, Djenouri D, Othman JB, Ouadjaout A, Badache N (2014) Congestion detection strategies in wireless sensor networks: a comparative study with testbed experiments. Procedia Comput Sci 37:168–175
19. Verma R, Singh NP (2013) GRA based network selection in heterogeneous wireless networks. Wirel Pers Commun 72(2):1437–1452
20. Wang Y-M, Luo Y (2010) Integration of correlations with standard deviations for determining attribute weights in multiple attribute decision making. Math Comput Model 51(1):1–12

21. Srikant R, Ying L (2013) Communication networks: an optimization, control, and stochastic networks perspective. Cambridge University Press, Cambridge
22. Wang L, Kuo G-S (2013) Mathematical modeling for network selection in heterogeneous wireless networks—a tutorial. IEEE Commun Surv Tutor 15(1):271–292
23. Huaizhou S, Prasad RV, Onur E, Niemegeers I (2014) Fairness in wireless networks: issues, measures and challenges. IEEE Commun Surv Tutor 16(1):5–24
24. Bertsekas DP (1999) Nonlinear programming. Athena Scientific, Belmont
25. Srikant R (2012) The mathematics of internet congestion control. Springer Science and Business Media, New York
26. Palomar DP, Chiang M (2006) A tutorial on decomposition methods for network utility maximization. IEEE J Sel Areas Commun 24(8):1439–1451
27. Tychogiorgos G, Leung KK (2014) Optimization-based resource allocation in communication networks. Comput Netw 66:32–45
28. Brown RG (2004) Smoothing, forecasting and prediction of discrete time series. Courier Corporation, Chelmsford
29. Michopoulos V, Guan L, Oikonomou G, Phillips I (2012) DCCC6: duty cycle-aware congestion control for 6LoWPAN networks. In: Proceedings of international conference on pervasive computing and communications workshops (PERCOM workshops). IEEE, pp 278–283
30. Kim H-S, Kim H, Paek J, Bahk S (2016) Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks. IEEE Trans Mob Comput
31. Dunkels A, Eriksson J, Finne N, Tsiftes N (2011) Powertrace: network-level power profiling for low-power wireless networks. Swedish Institute of Computer Science (SICS). Technical report
32. Zawodniok M, Jagannathan S (2007) Predictive congestion control protocol for wireless sensor networks. IEEE Trans Wirel Commun 6(11):3955–3963

Chapter 7
Conclusion and Future Work

7.1 Conclusion

In this section, a summary of the research findings of the thesis is presented. This thesis presents a concrete, solid and logically ordered work on congestion control for 6LoWPAN networks as a step toward successful implementation of the IoT and supporting the IoT application requirements.

A comprehensive congestion analysis and assessment for 6LoWPAN networks was presented in Chap. 3. An analytical modelling of congestion using Markov chain and Queuing theory was introduced. The proposed modelling models the average number of lost packets due to buffer overflow per second and the average number of received packets at a sink node every second. The outcomes of the modelling are: (i) the probability of packet loss due to buffer overflow depends on number of leaf nodes, buffer size, sending rate of leaf nodes and most significant the channel capacity; (ii) as buffer size is increased, packet loss due to buffer overflow at the leaf node decreases while it increases at the intermediate node. Further, an extensive congestion analysis for 6LoWPAN through simulations and testbed was carried out with different scenarios and various parameters (network size, network traffic load, buffer size, node density and application payload length). It was shown that: (i) the majority of packets are lost due to buffer overflow as compared to channel packet loss when congestion occurs; (ii) when the application payload length is increased since IPv6 packets are fragmented, the reassembly timeout parameter value has a significant effect on network performance; (iii) it is important to consider the buffer occupancy and the reassembly timeout parameter in RPL protocol design to improve network performance when congestion does occur.

In Chap. 4, a new RPL routing metric called buffer occupancy was proposed that reduces the number of lost packets due to buffer overflow when congestion does occur. Also, a new RPL objective function called congestion-aware objective function (CA-OF) was presented. The proposed objective function forwards packets through less congested intermediate nodes and therefore, packet loss is reduced significantly.

H. Al-Kashoash, *Congestion Control for 6LoWPAN Wireless Sensor Networks: Toward the Internet of Things*, Springer Theses,
https://doi.org/10.1007/978-3-030-17732-4_7

It was shown that by considering the buffer occupancy in objective function design of RPL, network performance improves in term of packet delivery ratio, throughput and energy consumption in the presence of congestion.

Further, in Chap. 5, the congestion control problem in 6LoWPAN networks was modelled as a game using the non-cooperative game theory. Also, a novel and simple congestion control algorithm called game theory based congestion control framework (GTCCF) was proposed. The proposed framework controls the sending rates of the nodes such that each node sends with its optimal sending rate (Nash equilibrium), while considering congestion alleviation in the network. To support the IoT application acquirements, the framework is aware of both node priorities and application priorities. It was shown that GTCCF improves performance in the presence of congestion in terms of throughput, end-to-end delay, energy consumption, number of lost packets and weighted fairness index as compared to DCCC6 algorithm.

A novel congestion control algorithm called optimization-based hybrid congestion alleviation (OHCA) was proposed in Chap. 6. The proposed algorithm combines traffic and resource control strategies into a hybrid solution to utilize the positive aspects of each strategy and efficiently use the network resources. A multi-attribute optimization methodology called GRA was used for forwarding packets through non-congested parents (i.e. resource control strategy). The NUM framework was utilized to achieve traffic control and compute the optimal nodes' sending rate. Also, the proposed algorithm is aware of node priorities and application priorities to support the IoT application requirements. The proposed algorithm was tested within two scenarios. The results indicated that OHCA improves performance in the presence of congestion in terms of throughput, weighted fairness index, end-to-end delay, energy consumption and buffer dropped packets as compared to DCCC6 and QU-RPL algorithms.

7.2 Future Work

This section presents some of the future work that can be summarized as follows:

1. Mobility-aware congestion Control for 6LoWPAN Networks: the IoT applications are diverse with different requirements. One important requirement for many applications (e.g. healthcare (wearable sensors)) is mobility. Future work can study the impact of node mobility on the proposed algorithms (CA-OF, GTCCF and OHCA) using Contiki OS, and then expand the proposed mechanisms to take into account node mobility in the network.
2. Investigate congestion in low power wide area networks (LPWAN) (e.g. LoRaWAN and SigFox) which are more extreme in term of constraints than 6LoWPAN networks. LPWAN is characterized by node constraints (e.g. very low cost, very limited processing capabilities, very small memory size and very low energy consumption) and link constraints (very short payload length and very low bandwidth). This will bring new challenges for congestion control

mechanisms for extreme constraint IoT nodes (e.g. LoRaWAN end nodes and SigFox end points).

3. The IoT is a huge umbrella under which are grouped a collection of technologies (e.g. 6LoWPAN, Bluetooth low energy (BLE), Wi-Fi, HaLow (IEEE 802.11ah), LPWAN, vehicular ad hoc network (VANET), wireless body area network (WBAN), etc.). Propose a global congestion control algorithm for the IoT heterogeneity that considers the huge number of heterogeneous things (e.g. 6LoWPAN motes, LPWAN nodes, BLE devices, etc.) and is aware of IoT protocol stacks diversity.