Accepted for publication in Wireless Networks Journal, Springer

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

Hayder A. A. Al-Kashoash, Harith Kharrufa, Yaarob Al-Nidawi, and Andrew H. Kemp

***Abstract*—The Internet of Things (IoT) is the next big challenge for the research community where the IPv6 over low power wireless personal area network (6LoWPAN) protocol stack is a key part of the IoT. Recently, the IETF ROLL and 6LoWPAN working groups have developed new IP based protocols for 6LoWPAN networks to alleviate the challenges of connecting low memory, limited processing capability, and constrained power supply sensor nodes to the Internet. In 6LoWPAN networks, heavy network traffic causes congestion which significantly degrades network performance and impacts on quality of service (QoS) aspects such as throughput, latency, energy consumption, reliability, and packet delivery. In this paper, we overview the protocol stack of 6LoWPAN networks and summarize a set of its protocols and standards. Also, we review and compare a number of popular congestion control mechanisms in wireless sensor networks (WSNs) and classify them into traffic control, resource control, and hybrid algorithms based on the congestion control strategy used. We present a comparative review of all existing congestion control approaches in 6LoWPAN networks. This paper highlights and discusses the differences between congestion control mechanisms for WSNs and 6LoWPAN networks as well as explaining the suitability and validity of WSN congestion control schemes for 6LoWPAN networks. Finally, this paper gives some potential directions for designing a novel congestion control protocol, which supports the IoT application requirements, in future work.**

***Index Terms*—Wireless sensor networks, 6LoWPAN networks, Internet of Things, congestion control, resource control, traffic control, hybrid schemes.**

I. Introduction

THE Internet of Things (IoT) is considered to be the next big opportunity and challenge for the Internet research community, technology users and companies [1]. The IoT is an emerging paradigm in which a variety of things or objects such as wireless sensor nodes, radio frequency identification (RFID) tags, and near field communication (NFC) devices are able to interact with each other and cooperate to achieve a common goal [2]. These things are connected to the Internet with the ability to sense status and condition and use real-time data whilst also accessing historical data and developed algorithms leading to very powerful 'smart' environments (home, office, and building), health care, etc. [2], [3].

The IoT is a huge umbrella under which are grouped a collection of technologies and different networks such as IPv6 over Low Power Wireless Personal Area Network (6LoWPAN). 6LoWPAN network is considered to be a crucial network and an important part in IoT world where 6LoWPAN motes will account for the majority of the IoT things [4], [5]. 6LoWPAN is used for full integration of WSNs with the Internet where sensor nodes implement the Internet Protocol (IP) stack, though it was originally designed for wired networks. However, the implementation of the TCP/IP model in WSNs and 6LoWPAN networks has many issues and problems due to the limited energy and buffer resources. TCP (transmission control protocol) requires connection setup and termination before and after the data transmission and UDP (user datagram protocol) does not provide a congestion control mechanism. Thus, TCP and UDP are not efficient for WSNs and 6LoWPAN networks [1], [2]. Therefore, one of the main issues in WSNs and 6LoWPAN networks is congestion that causes packet loss, increased 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, healthcare, 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 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 above three categories are combined into hybrid applications, i.e., sensor nodes send packets periodically and at the same time send packets in response to an event as well as sending a reply to a sink query. This type of application will be common in the future as WSNs are integrated with the Internet to form the IoT [2].

The major contributions of this paper are: **(i)** 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. **(ii)** The paper 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. **(iii)** To the best

H. A. A. Al-Kashoash is with Technical Institute/Qurna, Southern Technical University, Basra, Iraq (e-mail: hayderaam@stu.edu.iq).

Y. Al-Nidawi is with the Faculty of Engineering, Al-Mustansiriya University, Baghdad 10047, Iraq (e-mail: yaarob.al-_nidawi@uomustansiriyah.edu.iq).

H. Kharrufa and A. H. Kemp is with the Electronic and Electrical Engineering School, University of Leeds, Leeds LS2 9JT, U.K. (e-mail: elhdy@leeds.ac.uk; a.h.kemp@leeds.ac.uk).

of our knowledge, this is the first paper that reviews congestion control algorithms for 6LoWPAN networks and mechanisms which are built based on the unique characteristics of IEEE 802.15.4 standard, IPv6, and 6LoWPAN. **(iv)** This paper 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. **(v)** Furthermore, this paper 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.

The remainder of the paper is organized as follows: in Section II, we provide a review of related work on congestion control in WSNs and 6LoWPAN networks. Section III gives an overview of the 6LoWPAN protocol stack. Section IV provides an overview on why, how and where congestion occurs and also explains how to solve congestion. Section V provides information about performance metrics used to evaluate the proposed congestion control schemes. Section VI gives a short review of operating systems and simulators used to test and evaluate the proposed algorithms in WSNs and 6LoWPAN networks. In Sections VII and VIII, we review numerous congestion control algorithms and mechanisms for WSNs and 6LoWPAN networks respectively. Section IX discusses key issues addressed in the paper and gives directions for future work. Finally, Section X draws conclusions.

II. Related Work

Recently, a number of survey papers have focused on congestion control approaches for WSNs. Some of them cover less than 20 of existing congestion control mechanisms in WSNs [8], [9], [10], [11], [12], [13], [14] while, others review a large set of congestion control algorithms for WSNs [6], [7], [15]. However, none of those papers reviews congestion control mechanisms for 6LoWPAN networks. This paper provides a comprehensive review of a large number of congestion approaches in WSNs and gives a detailed review of the existing congestion control algorithms in 6LoWPAN networks. Moreover, this paper shows the importance of congestion control for 6LoWPAN networks and explains whether the existing congestion control mechanisms in WSNs are suitable and valid for 6LoWPAN networks.

In [8], Flora et al. reviewed a few existing congestion control algorithms (10 papers) in WSNs. Also, they gave an overview of performance metrics that are used for evaluating congestion control mechanisms. A simple comparison among the reviewed papers in terms of congestion identification, action initiator, and control mechanism was given. In [9], Yuan et al. reviewed a small number of congestion control approaches in WSNs (18 papers). They classified congestion control algorithms into four categories: rate regulation and allocation, routing optimization, data processing, and priority discrimination. Also, they highlighted some hot issues and difficulties of congestion control in WSNs.

In [10], Pant et al. reviewed a limited number of congestion control mechanisms in WSNs (eight papers). They classified algorithms into traffic control and resource control. Also, they presented the significance and limitations of the mechanisms and gave a comparison among them based on congestion detection, congestion mitigation, congestion notification, and fairness. In [11], Zhao et al. reviewed a number of congestion control algorithms for WSNs (13 papers) and they classified them into MAC layer schemes and cross layer schemes. Also, they discussed the reviewed congestion control schemes and highlighted some common features that may direct future research.

In [12] and [13], the authors reviewed a limited number of congestion detection and control techniques in WSNs (eight papers). They gave a comparison among various congestion control protocols in terms of direction, data flow, congestion detection, congestion control, and energy conservation. Also, they highlighted some improvements and future considerations in congestion control for WSNs. In [14], Budhwar surveyed a number of transport layer protocols for both congestion control and reliability in WSNs (11 papers). It described aspects of congestion control and reliability and presented a comparison among the reviewed papers in many aspects and parameters, e.g., congestion detection, congestion avoidance, reliability level, etc. Also, they presented some research issues for congestion control protocols in WSNs.

In [6], [7], [15], Ghaffari, Kafi et al., and Sergiou et al. reviewed, compared, and classified a large set of congestion control approaches in WSNs. The authors discussed the characteristics, advantages, and disadvantages of each approach. Also, they derived some potential directions for improvements of congestion control mechanisms in future work.

III. 6LoWPAN Protocol Stack Overview

6LoWPAN enables transmission of IPv6 packets over low power, memory, bandwidth, processing capability, and cost devices which are compatible with the IEEE 802.15.4 standard. 6LoWPAN provides a 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 Figure 1. A review of the 6LoWPAN model layers is given in the next subsections.

A. Application Layer

The IoT makes use of most of the Internet application protocols which are equally important for 6LoWPAN [1]. However, 6LoWPAN is challenging in this aspect due to small frame size, limited data rate, limited memory, limited processing capabilities, and power supply. Recently, the Constrained RESTfull Environments (CoRE) working group has developed an important application protocol called Constrained Application Protocol (CoAP) which is a REST based web transfer protocol [16]. CoAP includes a subset of HTTP functionalities which have been re-designed to meet the 6LoWPAN constraints. The

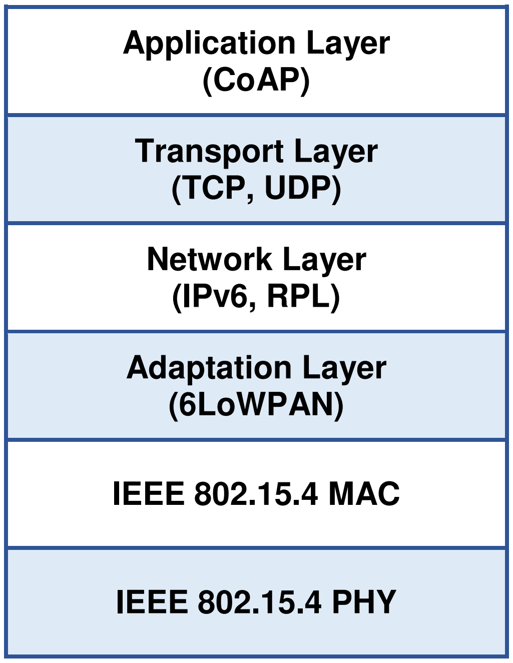
Application Layer (CoAP)
Transport Layer (TCP, UDP)
Network Layer (IPv6, RPL)
Adaptation Layer (6LoWPAN)
IEEE 802.15.4 MAC
IEEE 802.15.4 PHY

Fig. 1. 6LoWPAN protocol stack

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 a Method Code. 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.

B. 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 ACK and retransmission. Also, TCP provides end-to-end flow control and congestion control by using a sliding window algorithm. Figure 2 shows the difference between flow control and congestion control [17]. Figure 2 (a) shows the flow control problem where a small capacity and slower receiver is overwhelmed by a fast-transmitting sender. While, in the congestion control problem, a limited resources network is congested due to high offered-load packets into the network as shown in Figure 2 (b). On the other hand, UDP is the simplest protocol on the TCP/IP suite. It does not support reliability and congestion control. Due to the 6LoWPAN limitations, UDP is the most common transport protocol used in 6LoWPAN networks.

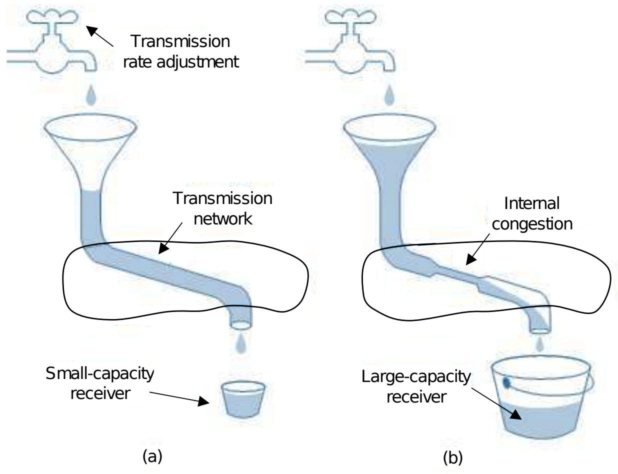

Fig. 2. (a) Flow control problem. (b) Congestion control problem. [17]

C. 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 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 [18]:

- 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 take 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' [19]. With the mesh-under scheme, 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.

Recently, a number of routing protocols have been developed for 6LoWPAN such as HiLow, LOAD, DYMO-low, and RPL. Hierarchical routing over 6LoWPAN (HiLow) [20] uses dynamically assigned 16-bit unique short address for a 6LoWPAN device during an association operation with a neighboring 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) [21] is proposed based on 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 that 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 the 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) [22] 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. Finally, IPv6 routing protocol for low power and lossy networks (RPL) [23] was developed by the RoLL working group to meet the requirements and challenges of low power and lossy networks (LLNs). RPL is a distance vector routing protocol which organises the network as a Directed Acyclic Graph (DAG) routed at the sink. It constructs the network topology by using an objective function which defines how routing metrics are computed to obtain a Rank value. The Rank value represents a nodes' position in the graph and the node selects its parent based on the Rank. RPL is expected to be the standard routing protocol for LLNs and 6LoWPAN networks.

D. 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, memory, bandwidth, etc.. 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: IPv6 header compression, IPv6 fragmentation and reassembly and 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 only a maximum 54 bytes for application payload. Therefore, IPv6 header compression is very important to reduce header overhead and increase application payload space. The RFC 6282 [24] defines how to compress the IPv6 and UDP headers efficiently by using improved header compression (IPHC) and next header compression (NHC) methods.

The IEEE 802.15.4 defines the maximum transmission unit (MTU) of 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 is reassembled and delivered up to the network layer. RFC 4944 [25] 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 above two functions, the adaptation layer supports the mesh-under routing scheme to forward packets inside the 6LoWPAN network.

E. MAC and Physical Layers

IEEE 802.15.4 [26] 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 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 slot (GTS) management, frame validation, acknowledged frame delivery, association, and disassociation. 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 the 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: 868-868.6 MHz (Europe), 902-928 MHz (North America), or 2400-2483.5 MHz (worldwide).

IV. Congestion in WSNs and 6LoWPAN Networks

WSN is a network formed by a large number of sensor nodes that are spatially distributed and organised 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. [27]. Sensor nodes have limited resources with regards to memory, computation capabilities, bandwidth, and power supply. Due to these limitations and constraints, the traditional congestion control schemes used in the Internet, i.e., TCP, cannot be applied to WSNs and designing a new congestion control scheme is challenging [6]. 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 [6]. 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) [6], [7], [10], [15]. 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 [28]. 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 [29]. As wireless links are unreliable, it is impractical to support an end-to-end connection to transmit packets in wireless links [30]. Also, the major benefits of the hop-by-hop approach is 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 [6], [7], [15] as shown in Figure 3.

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 [31].

- **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.
- **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.
- **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 wireless 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 misunderstanding 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. By 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 by using two ways either rate adjustment

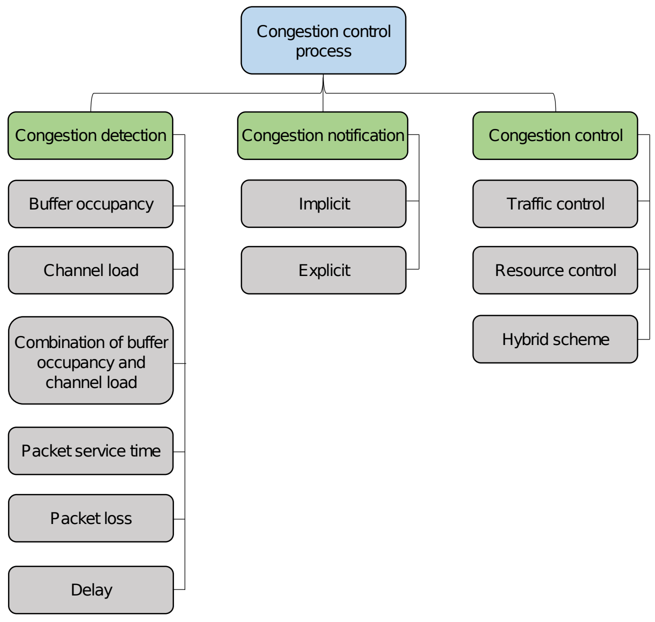

Fig. 3. Congestion control steps

(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 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 [32] as follows:

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

  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 a successful transmission at the MAC layer (in second), c_b is channel busyness ratio and th_b is channel bandwidth threshold. 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 practical.
- **Resource control**: to avoid the drawback of the traffic control scheme, 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 by 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.

V. 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, fidelity penalty, normalized reliability, energy consumption, throughput, fairness, latency, buffer drop rate, packet loss rate, queue length, packet delivery ratio, source rate, Jain's fairness index, 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 [33]. 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 [34].
- **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 [32].
- **Throughput**: is the total number of successfully received packets at sink node per unit time (typically every second) [32]. Some papers count the total number of packets received by the server (sink node) and call it **goodput** [35].
- **Fairness**: 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 [36]. Some papers use **weighted fairness** to achieve different throughput according to nodes' priority and importance [37].
- **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 [38]. Some papers call it **end-to-end delay** [39]. Some algorithms are evaluated by using **hop-by-hop delay** which is the time from a child node to its parent (one hop only) [40].
- **Buffer drop rate (queue loss ratio)**: this metric measures the probability that a packet will be dropped due to buffer overflow [38], [41]. Some papers call it **rejection rate** [42]. This metric does not take into account the wireless channel loss [43].
- **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 [36]. Some papers call it **loss probability** [42]. This metric takes into account the total number of lost packets due to buffer overflow and wireless channel loss [39], [44].
- **Queue length (queue level)**: this metric shows the average number of packets stored in the nodes' buffer over time [45], [46].
- **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 [36], [47]. Some papers call it **packet reception rate** [40], [45].
- **Source rate**: is the total number of packets generated by source nodes per second [32], [48].
- **Jain's fairness index**: is defined as a function of variability of throughput across nodes in the network as in the equation below. It is an indication of how many of the nodes are

treated fairly [35], [38].

$$f(th_1, th_2, ..., th_n) = \frac{(\sum_{i=1}^{n} th_i)^2}{n \sum_{i=1}^{n} th_i^2}$$

where n is the total number of nodes and th_i is throughput of node i.

- **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 [49].
- **Others** such as **control overhead packets** [41], [42], [48], **network efficiency** [38], [50], **hop count** [41], [51], [52] and **quality of data (QoD)** [50].

VI. Operating Systems and Simulators for WSN and 6LoWPAN Networks

It is very important to choose an appropriate tool for testing, analyzing, and evaluating a proposed algorithm performance. Real testbeds provide a better option for studying behavior of the proposed algorithm in realistic environments and scenarios. TinyOS, Contiki OS, and RIOS OS are an excellent choice to examine and evaluate the proposed mechanisms as they are real, widely used operating systems supporting 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. Thus, TOSSIM, Cooja, and ns-3 are good choices to evaluate algorithms at design, development, and implementation stages. Sometimes, TOSSIM and Cooja are considered emulators as they execute the same code on real motes [53].

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 Figure 4. 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** [54]: 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** [55]: 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 [56]. 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 run-time, network-level, power profiling system called Powertrace [57] which uses state tracking to estimate and measure the energy consumption of each node and it is accurate up to 94%.
- **RIOT OS** [58]: 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 micro-kernel architecture inherited from FireKernel [59] 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. RIOS OS runs on several platforms including embedded devices e.g., TelosB, Zolerita Z1, Arduino Due, etc., as well as personal computers. RIOT OS supports 6LoWPAN protocol stack, openWSN, and Arduino API.
- **TOSSIM** [60]: 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

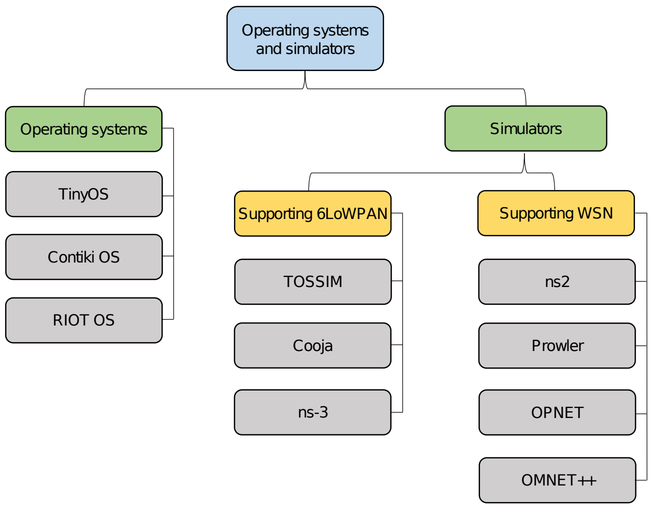

Fig. 4. Operating systems and simulators for WSN and 6LoWPAN networks

components. However, TOSSIM does not model the real environment and it provides a radio abstraction of directed independent bit errors between two nodes. Also, it does not model the energy consumption and it supports only one hardware platform model (MicaZ).

- **Cooja** [61]: is a cross-level, flexible, Java-based simulator designed for simulating a network of sensor nodes which run Contiki OS. Cooja simulates different types of real sensor motes such as Tmote Sky, Z1, WiSMote, MicaZ, and ESB (embedded sensor board). Cooja allows for simultaneous simulations 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). Also, it has a useful tool for development and debugging called TimeLine which shows a time line for each node [62]. TimeLine shows a power state of the node's radio transceiver: off, on, transmission and reception as well as radio interference. However, the limitation of Cooja is that when the number of nodes exceeds the allowable limit, the simulation time becomes very long.
- **ns2** [63]: 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** [64]: 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** [65]: 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 communication channel and application, e.g., radio channel and MAC layer. However, it does not consider and support the 6LoWPAN protocol stack.
- **OPNET** [66]: 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++** [67]: 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 by 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 [68] have developed a 6LoWPAN simulation model for OMNET++ by integrating Contiki's implementation into OMNET++.

VII. 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.

A. *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 I summarizes these algorithms.

In [33], 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 toward sources. Every node receives the backpressure message, it decides whether to broadcast the message again or not, based on its local network conditions. When a source node receives a backpressure message, it regulates its rate based on the maximum theoretical throughout of the channel S_{max}. When the source event rate r is less than a fraction η of S_{max}, the source regulates itself. Otherwise, if the value of r is higher than η, the closed-loop control is triggered where the sink node regulates the source rate.

CODA has been tested through real experiments by using a small sensor network testbed with TinyOS and through simulations by 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 [34], 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

TABLE I
TRAFFIC CONTROL ALGORITHMS IN WSNS.

Algorithm	**Congestion detection**	**Congestion notification**	**Application type**	**Implementation/ (Number of nodes)**	**Evaluation metrics**	**Compared with**
CODA [33]	Buffer occupancy and channel load	Explicit	Event-based	Simulation and real experiments (TinyOS)/ (30 – 120 nodes)	Energy tax and fidelity penalty	No CC and Open-loop CC
ESRT [34]	Buffer occupancy	Implicit	Event-based	Simulation (ns2) and analytical/ (–)	Normalized reliability and power consumption	—-
Fusion [38]	Buffer occupancy	Implicit	Event-based and continuous	Real experiments (TinyOS)/ (55 nodes)	Throughput, fairness, latency, drop rate, and efficiency	No CC, occupancy, channel sampling, rate limiting, and occupancy+Delay
IFRC [46]	Weighted moving average of queue length	Explicit	Continuous	Real experiments (TinyOS)/ (40 nodes)	Throughput and Instantaneous queue size	—-
PCCP [37]	Packet service time / packet inter-arrival time	Implicit	Event-based and continuous	Simulation/ (7 nodes)	Normalized throughput, queue length, and fairness	CCF [69]
DPCC [70]	Buffer occupancy	Implicit	—-	Simulation (ns2 and MATLAB)/ (10 nodes)	Throughput, queue level, and delay	CODA [33]
HCCP [48]	Buffer occupancy and flow rate	Explicit	Continuous	Simulation (ns2)/ (5000 nodes)	Total source rate and control overhead packets	AFA [71] and buffer-based congestion avoidance scheme [72]
Multipath CC [73]	Average packet service rate / packet scheduling rate	Implicit	Continuous	Simulation/ (200 nodes)	Queue length and throughput	—-
FACC [32]	Buffer occupancy and channel load	Explicit	Continuous	Simulation (ns2)/ (51 nodes)	Dropped packets, total source rate, throughput, and energy expenditure	No CC and CODA [33]
CL-APCC [45]	Buffer occupancy and data flow	Implicit	Continuous	Simulation (VC++)/ (100 nodes)	Packet reception rate, queue length, energy consumption	No CC
UHCC [74]	Buffer occupancy and traffic rate	Implicit	Continuous	Simulation/ (11 nodes)	Normalized throughput, fairness, and packet loss ratio	PCCP [37] and CCF [69]
ACT [50]	—-	Implicit	Continuous	Simulation (TOSSIM)/ (100 nodes)	Efficiency, fairness, quality of data, and energy	CODA [33] and CRRT [75]
Distributed CC [76]	Difference between input and output traffic rates	Implicit	Continuous	Simulation/ (100 nodes)	Goodput, fairness, and transmission rate	—-
DPCC [77]	Buffer occupancy and traffic rate	Explicit	—-	Simulation/ (10 nodes)	Normalized throughput and fairness	PCCP [37]
DRR [36]	Buffer occupancy	Explicit	Continuous	Simulation (ns2)/ (6, 26 nodes)	Packet delivery ratio, packet loss ratio, fairness, and energy consumption	—-
FBACC [39]	Buffer occupancy and traffic rate	Explicit	—-	Simulation (MATLAB)/ (–)	Congestion detection, packet loss, end-to-end delay, and energy	ESRT [34], FLCE [78], and CCSFL [79]

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, that 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, r, is lower than a specific value, the sink adjusts the reporting rate, f, 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 by 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 [38], 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 prioritised 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, α, 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 prioritised 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 [46], Rangwala et al. proposed an interference-aware fair rate control algorithm (IFRC) to allocate fair and efficient transmission rate to each node. IFRC comprises of three components: congestion level measurement, congestion information sharing and rate adaptation using the AIMD scheme. IRFC measures congestion level by using an exponentially weighted moving average of the queue length. If the average queue length, Avg_q, exceeds a certain threshold value, U, 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 [37], 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 of 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 multi-path routing scenarios.

PCCP has been evaluated through simulation within a 7 node network under single-path and multi-path routing scenarios. Also, PCCP is compared with congestion control and fairness algorithm (CCF) [69]. 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 [70], Zawodniok and Jagannathan developed a decentralized predictive congestion control algorithm (DPCC) for WSNs. The proposed algorithm comprises of two schemes (adaptive flow and adaptive CSMA back-off interval selection) that work in concert with a distributed power control (DPC). DPCC detects congestion by 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 summarised by 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 [33]. Simulation results show that DPCC increases throughput, network efficiency, and energy saving and DPCC guarantees the targeted QoS as compared to CODA.

In [48], 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 period time T. 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 x 1000 m, and compared with aggregate fairness algorithm (AFA) [71] and lightweight buffer management based congestion avoidance scheme [72]. Simulation results show that the proposed protocol has better performance in terms of throughput and packet drop rate than others.

In [73], 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 x 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 [32], 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 categorise 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 [33]. 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 [45], 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 t. CL-APCC predicts the input and output rates of node within the next period, $t + 1$, 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 [74], Wang and Liu proposed a protocol called upstream hop-by-hop congestion control (UHCC) based on cross-layer design. The proposed algorithm comprises of 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 [37] and CCF [69]. The simulation results show that the proposed algorithm achieves higher throughput, better priority-based fairness, and reduced packet loss than other algorithms.

In [50], 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 firstly transforms the data from time domain to frequency domain by 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 [33] and congestion-aware rate-controlled reliable transport algorithm (CRRT) [75]. 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 [76], 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 by 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 works as follows: each gateway node, which is one hop away node from the sink node, executes the following steps every control interval. The other nodes implement the algorithm when the transmission rate of its parent changes.

1) Calculate the average packet sending rate, r_{out}, the average aggregate input rate, r_{in}, and Q which is the minimum number of packets in the output queue during the control interval.
2) Based on the difference between r_{out} and r_{in}, and Q, the node computes the total change in aggregate traffic, Δt, as: $\Delta t = \alpha * (r_{in} - r_{out}) - \beta * (Q/t_{CI})$ where α, β, and t_{CI} are constants.
3) Divide Δt among individual flows to achieve fairness.

4) Compare the calculated bandwidth for each flow with the bandwidth advertised by its parent where the smaller upstream rate is chosen.

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

In [77], Heikalabad et al. proposed an algorithm called dynamic prediction congestion control (DPCC). The proposed algorithm comprises of 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 [37].

In [36] 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 x 1000 m. For the chain topology, there are six sensor nodes with 2 Joules each and three seconds simulation time whereas 26 sensor nodes with 2 Joules and 10 seconds 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 [39] 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 [34], fuzzy logic based congestion estimation algorithm (FLCE) [78], and congestion control scheme based on fuzzy logic (CCSFL) [79] 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.

B. *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 II summarizes these mechanisms.

In [49], 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 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-coloring 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 distributes the incoming traffic between the original path and the alternative path whereas the merger merges these 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 [80], 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.

TABLE II
RESOURCE CONTROL ALGORITHMS IN WSNs.

Algorithm	**Congestion detection**	**Congestion notification**	**Application type**	**Implementation/ (Number of nodes)**	**Evaluation metrics**	**Compared with**
TARA [49]	Buffer occupancy and channel load	Explicit	Continuous	Simulation (ns2)/ (81 nodes)	Fidelity index and energy consumption	No CC, traffic control, and resource control
TADR [80]	Buffer occupancy	—-	Event-based	Simulation (TOSSIM)/ (999 nodes)	Receiving packets rate, throughput ratio, and energy efficiency	MintRoute algorithm of TinyOS [81]
QoS Adaptive cross layer CC [82]	Packet inter-arrival time / packet service time	Implicit	Continuous	Simulation/ (50 nodes)	Average queue length and energy	No CC and CCF [69]
HTAP [40]	Buffer occupancy	Implicit	Event-based	Simulation (Prowler)/ (100 nodes)	Received packets ratio, throughput, hop-by-hop delay, and energy consumption	No CC, TARA [49], and SenTCP [83]
DAlPaS [84]	Buffer occupancy and channel load	Implicit	Continuous	Simulation (Prowler)/ (100 nodes)	Received packets ratio, throughput, hop-by-hop delay, and end-to-end delay	No CC, TARA [49], and HTAP [40]
CATree [85]	—-	—-	Continuous	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

In [82], 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 (t_a^i) and packet service time (t_s^i) 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 [69]. Simulation results show that the proposed scheme improves network throughput, average queue occupancy, and energy consumption as compared to others.

In [40], 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 a 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 recognise 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, that 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 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 x 500 m. Simulation results show that the proposed algorithm is a more efficient and simple solution for the congestion problem than TARA [49] and the hop-by-hop congestion control protocol (SenTCP) [83]. However, HTAP consumes energy by using overhead packets (hello, level_discovery, and backpressure messages).

In [84], 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 back-pressure 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 [49], and HTAP [40]. DAlPaS is tested by using the Prowler simulator with 100 nodes which are deployed uniformly in an area of 50 m x 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 back-pressure messages) during the setup phase that increase the consumed energy. Moreover, the limitation of the proposed algorithm that each node should be aware of its position and the position of the sink node.

In [85], 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 x 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.

C. Hybrid Schemes

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

In [43], 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 multi-path detouring technique is used to increase the channel capacity for congested flows. TALONet comprises of 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

TABLE III
HYBRID ALGORITHMS IN WSNS.

Algorithm	**Congestion detection**	**Congestion notification**	**Application type**	**Implementation/ (Number of nodes)**	**Evaluation metrics**	**Compared with**
TALONet [43]	Buffer occupancy	Implicit	Continuous	Simulation (ns2)/ (50 – 200 nodes)	Dropped packets and power consumption	No CC, TARA [49], and backpressure
Mutlipath routing CC [86]	Buffer occupancy	Explicit	Continuous	Simulation (ns2)/ (1000 nodes)	Throughput and packet delivery ratio	No CC, buffer-based congestion avoidance scheme [72], and PCCP [37]
CADA [47]	Buffer occupancy and channel load	Implicit	Event-based	Simulation (ns2)/ (500 – 5000 nodes)	End-to-end delivery ratio, bit energy consumption, per-hop delay, and throughput	No CC, TARA [49]
HRTC [87]	Buffer occupancy	Explicit	Continuous	Simulation (Prowler)/ (30 nodes)	Throughput	No CC, traffic control, and resource control

[49] and backpressure method. Simulation results show that TALONet improves packet delivery rate, increases network lifetime, and saves energy as compared to others.

In [86], 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 [72], and PCCP [37]. 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 [47], 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. Firstly, 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. Secondly, 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 [49] 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 [87], 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 x 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.

VIII. 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 (Section VIII-A) and resource control algorithms (Section VIII-B). Table IV summarizes these mechanisms and table V shows the advantages and disadvantages of the algorithms.

A. *Traffic Control Algorithms*

In [35], 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 [48], AFA [71], IFRC [46], and CSMA. In the simulation, 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 high 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 [42], 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 back pressure concept which is proposed in [88], 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 (back pressure) 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, Q_{max}, 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 9 leaf nodes, 4 routers, and 1 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, transmission overhead. The transmission overhead includes the number of transmissions for successfully received single packets 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 [100], Al-Kashoash et al. formulated the congestion problem in 6LoWPAN networks as a noncooperative 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%,

TABLE IV
TRAFFIC AND RESOURCE CONTROL ALGORITHMS IN 6LOWPAN NETWORKS.

Algorithm	**Congestion detection**	**Congestion notification**	**Congestion Control**	**Application type**	**Implementation/ (Number of nodes)**	**Evaluation metrics**	**Compared with**
DCCC6 [35]	Buffer occupancy	Implicit and explicit	Traffic control	Continuous	Simulation (Cooja) and real experiments (Contiki OS)/ (15, 25 nodes)	Goodput, end-to-end delay, energy consumption, and Jain's fairness index	HCCP [48], AFA [71], IFRC [46], and CSMA
Griping, Deaf, and Fuse [42]	Buffer occupancy	Implicit and explicit	Traffic control	Continuous	Simulation (ns-3)/ (14 nodes)	Reception rate, multihop delay, loss probability, rejection rate, and transmission overhead	backpressure [88] and UDP
Bird flocking CC [89]	Buffer occupancy	—-	Resource control	Continuous	Simulation (Cooja)/ (50 nodes)	Duplicate messages and transmission time	CoAP [16]
QU-RPL [41], [90]	Buffer Overflow	Explicit	Resource control	Continuous	Real experiments (TinyOS)/ (30 nodes)	Packet delivery, packet loss ratio, hop distance, and routing overhead packets	RPL [23]
GTCC [51], [52]	Difference between packet generation rate and packet service rate	Explicit	Resource control	Continuous	Simulation (Cooja)/ (22, 26 nodes)	Packet loss rate, throughput, and hop count	RPL with OF0 [91] and RPL with ETX-OF [92]
CA-RPL [93]	—-	Implicit	Resource control	Continuous	Simulation (Cooja)/ (21 nodes)	Throughput, packet loss rate, and average end-to-end delay	Original RPL [23]
CA-OF RPL [44]	Buffer occupancy	Implicit	Resource control	Continuous	Simulation (Cooja)/ (19, 35 nodes)	Number of lost packets, throughput, packet delivery ratio, and energy consumption	RPL with OF0 [91], RPL with ETX-OF [92], and RPL with ENERGY-OF [94]
Lodhi's M-RPL [95]	Packet delivery ratio	Implicit	Resource control	Continuous	Simulation (Cooja)/ (113 nodes)	Throughput, end-to-end latency, and energy consumption	RPL [23]
MLEq [96]	—	—	Resource control	Continuous	Simulation (ns2)/ (100 nodes)	Throughput, Jain's fairness index, and control packet overhead	RPL [23]
LB-RPL [97], [98]	—	—	Resource control	Continuous	Simulation (ns2)/ (1000 nodes)	Packet delivery ratio and end-to-end delay	RPL [23]
Tang's M-RPL [99]	—	—	Resource control	Continuous	Simulation (Cooja)/ (20 nodes)	Packet reception rate, packet loss rate, and end-to-end delay	RPL [23]
GTCCF [100]	ratio of forwarding rate to receiving rate	Explicit	Traffic control	Continuous	Simulation (Cooja)/ (5, 21 nodes)	packet loss, throughput, delay, weighted fairness index, and energy consumption	DCCC6 [35]
OHCA [101]	ratio of forwarding rate to receiving rate	Explicit	Hybrid scheme	Continuous	Simulation (Cooja)/ (10, 25 nodes)	packet loss, throughput, delay, weighted fairness index, and energy consumption	DCCC6 [35] and QU-RPL [41], [90]

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 DCCC6 algorithm.

B. Resource Control Algorithms

In [89], 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.

TABLE V
ADVANTAGES AND DISADVANTAGE OF CONGESTION CONTROL ALGORITHMS IN 6LOWPAN NETWORKS.

Algorithm	Advantages	Disadvantages
DCCC6 [35]	• 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 [42]	• 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 [89]	• 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 [41], [90]	• 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 [51], [52]	• 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 [93]	• 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
CA-OF RPL [44]	• 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
Lodhi's M-RPL [95]	• 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
MLEq [96]	• Achieves load balancing and distribution based on water flow behavior 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 [97], [98]	• 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 [99]	• 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

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 x 201 m during 300 seconds 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 transactions. However, the proposed technique is executed even when the network is not congested. Therefore, packets may not pass through a 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 [41], [90], 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 [52] and [51], 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 RPL protocols.

In [93], 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 toward 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 [44], Al-Kashoash et al. 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 [95], 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 send 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 [96], 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 behavior such that water flows downward and finds its own level. The proposed scheme models all the traffic flows to each gateway in the network as a 3-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 neighbors. Every intermediate (router) node receives the VIO message, it updates its VL value and sends multicast VIO messages to its neighbors. 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 [97] and [98], 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 top k 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 to traffic load distribution, packet delivery rate, and end-to-end delay.

In [99], Tang et al. proposed a multipath routing optimization strategy for RPL called M-RPL which relives 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 [101], Al-Kashoash et al. 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 noncongested 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, in [102], Al-kashoash et al. proposed a new analytical model of congestion for 6LoWPAN network using Markov chain and queuing theory. The derived model calculates the buffer loss probability and the channel loss probability as well as the number of received packets at the final destination in the presence of congestion. Also, some papers have modelled and analyzed TCP performance over 6LoWPAN network. In [103], 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 7 node network by using Contiki OS. In [104], 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 [105], Kim et al. presented a comprehensive experimental study on the performance of TCP over RPL in 6LoWPAN network by using TinyOS and a multihop 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 effect the operation of RPL in terms of control overhead and parent changes.

IX. 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 [40], [43], [47], [49], [80], [82], [84], [85], [86], [87], [106] 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 [107], 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 [32], [33], [34], [36], [37], [43], [45], [47], [48], [49], [70], [73], [77], [82], [86], [106]. 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 [38], Hull et al. analyzed congestion through testbed experiments in a traditional WSN protocol stack with TinyOS where B-MAC and single destination DSDV (destination sequenced distance vector) are used. They concluded that wireless channel losses dominate buffer overflow and increase quickly with increasing offered load. On the other hand, in [108], Al-Kashoash et al. analyzed congestion through simulation in 6LoWPAN protocol stack by using Contiki OS and Cooja simulator. In contrast to Hull's conclusion, the authors have concluded that the majority of packets are lost due to buffer overflow as compared to channel loss. Also, they have concluded that the number of lost packets due to buffer drops increase 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 [108], Al-Kashoash et al. did congestion analysis for 6LoWPAN networks and they 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 [109], and the number of the wireless sensor devices will account for the majority of these [110], [5]. 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.

X. 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 paper, 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 end of 2017 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 congestion 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 and **vii)** be aware of data packet priority, application priority as well as node priority to support the IoT application requirements.

REFERENCES

[1] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*. John Wiley & Sons, 2009.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] C. Alcaraz, P. Najera, J. Lopez, and R. Roman, "Wireless Sensor Networks and The Internet of Things: Do We Need a Complete Integration?" in *1st International Workshop on the Security of the Internet of Things (SecIoT'10)*, Tokyo (Japan), 29th November, 2010.

[4] N. Khalil, M. R. Abid, D. Benhaddou, and M. Gerndt, "Wireless Sensors Networks for Internet of Things," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*. IEEE, 2014, pp. 1–6.

[5] H. Al-Kashoash and A. H. Kemp, "Comparison of 6LoWPAN and LPWAN for the Internet of Things," *Australian Journal of Electrical and Electronics Engineering*, vol. 13, no. 4, pp. 268–274, 2017.

[6] A. Ghaffari, "Congestion Control Mechanisms in Wireless Sensor Networks: A Survey," *Journal of Network and Computer Applications*, vol. 52, pp. 101–115, 2015.

[7] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion Control Protocols in Wireless Sensor Networks: A Survey," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1369–1390, 2014.

[8] D. J. Flora, V. Kavitha, and M. Muthuselvi, "A Survey on Congestion Control Techniques in Wireless Sensor Networks," in *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on*. IEEE, 2011, pp. 1146–1149.

[9] H. Yuan, N. Yugang, and G. Fenghao, "Congestion Control for Wireless Sensor Networks: A Survey," in *Control and Decision Conference (2014 CCDC), The 26th Chinese*. IEEE, 2014, pp. 4853–4858.

[10] N. Pant, M. Singh, and P. Kumar, "Traffic and Resource based Methods for Congestion Control in Wireless Sensor Networks: A Comparative Analysis," in *Adaptive Science & Technology (ICAST), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 1–6.

[11] J. Zhao, L. Wang, S. Li, X. Liu, Z. Yuan, and Z. Gao, "A Survey of Congestion Control Mechanisms in Wireless Sensor Networks," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*. IEEE, 2010, pp. 719–722.

[12] P. Gowthaman and R. Chakravarthi, "Survey on Various Congestion Detection and Control Protocols in Wireless Sensor Networks," *Int J Adv Comput Eng Commun Technol*, vol. 2, no. 4, pp. 15–19, 2013.

[13] R. Chakravarthi, C. Gomathy, S. K. Sebastian, K. Pushparaj, and V. B. Mon, "A Survey on Congestion Control in Wireless Sensor Networks," *International Journal of Computer Science & Communication*, vol. 1, no. 1, pp. 161–164, 2010.

[14] P. Budhwar, "A Survey of Transport Layer Protocols for Wireless Sensor Networks," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 2, pp. 985–991, 2015.

[15] C. Sergiou, P. Antoniou, and V. Vassiliou, "A Comprehensive Survey of Congestion Control Protocols in Wireless Sensor Networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 1839–1859, 2014.

[16] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," *IETF RFC 7252*, 2014.

[17] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks: Pearson New International Edition: University of Hertfordshire*. Pearson Higher Ed, 2013.

[18] C. Gomez, E. Kim, D. Kaspar, and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing," *IETF RFC 6606*.

[19] A. H. Chowdhury, M. Ikram, H.-S. Cha, H. Redwan, S. Shams, K.-H. Kim, and S.-W. Yoo, "Route-over vs Mesh-under Routing in 6LoWPAN," in *Proceedings of the 2009 international conference on wireless communications and mobile computing: Connecting the world wirelessly*. ACM, 2009, pp. 1208–1212.

[20] S. Yoo, J. Lee, and G. Mulligan, "Hierarchical Routing over 6LoWPAN (HiLow)," draft-daniel-6lowpan-hilow-hierarchical-routing-01, IETF Network Working Group, Internet-Draft, Tech. Rep., 2007.

[21] K. Kim, S. D. Park, G. Montenegro, S. Yoo, and N. Kushalnagar, "6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)," *Internet Draft, draft-daniel-6lowpan-load-adhoc-routing-03, IETF*, 2007.

[22] K. Kim, G. Montenegro, S. Park, I. Chakeres, and C. Perkins, "Dynamic MANET On-demand for 6LoWPAN (DYMO-low) Routing," *IETF, Internet-Draft*, 2007.

[23] T. Winter, P. Thubert, A. Brandt, J. Hui, and R. Kelsey, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *IETF, RFC 6550*, 2012.

[24] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams Over IEEE 802.15. 4-Based Networks," *IETF RFC 6282*, 2011.

[25] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets Over IEEE 802.15.4 Networks," *IETF RFC 4944*, 2007.

[26] L. S. Committee *et al.*, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Aetworks (LR-WPANs)," *IEEE Computer Society*, 2003.

[27] S. Yinbiao and et al., "Internet of Things: Wireless Sensor Networks. White Paper," *International Electrotechnical Commission (IEC)*, 2014.

[28] P. Reena and L. Jacob, "Hop-by-Hop versus End-to-End Congestion Control in Wireless Multi-Hop UWB Networks," in *Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on*. IEEE, 2007, pp. 255–261.

[29] S. Heimlicher, P. Nuggehalli, and M. May, "End-to-end vs. Hop-by-hop Transport," *SIGMETRICS Performance Evaluation Review*, vol. 35, no. 3, pp. 59–60, 2007.

[30] S. Heimlicher, M. Karaliopoulos, H. Levy, and M. May, "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), 2007, p. 20.

[31] M. A. Kafi, D. Djenouri, J. B. Othman, A. Ouadjaout, and N. Badache, "Congestion Detection Strategies in Wireless Sensor Networks: A Comparative Study with Testbed Experiments," *Procedia Computer Science*, vol. 37, pp. 168–175, 2014.

[32] X. Yin, X. Zhou, R. Huang, Y. Fang, and S. Li, "A Fairness-Aware Congestion Control Scheme in Wireless Sensor Networks," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 9, pp. 5225–5234, 2009.

[33] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 266–279.

[34] Y. Sankarasubramaniam, Ö. B. Akan, and I. F. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. ACM, 2003, pp. 177–188.

[35] V. Michopoulos, L. Guan, G. Oikonomou, and I. Phillips, "DCCC6: Duty Cycle-Aware Congestion Control for 6LoWPAN Networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 278–283.

[36] V. S. Deshpande, P. P. Chavan, V. M. Wadhai, and J. B. Helonde, "Congestion Control in Wireless Sensor Networks by using Differed Reporting Rate," in *Information and Communication Technologies (WICT), 2012 World Congress on*. IEEE, 2012, pp. 209–213.

[37] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream Congestion Control in Wireless Sensor Networks through Cross-Layer Optimization," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 4, pp. 786–795, 2007.

[38] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 134–147.

[39] S. Jaiswal and A. Yadav, "Fuzzy Based Adaptive Congestion Control in Wireless Sensor Networks," in *Contemporary Computing (IC3), 2013 Sixth International Conference on*. IEEE, 2013, pp. 433–438.

[40] C. Sergiou, V. Vassiliou, and A. Paphitis, "Hierarchical Tree Alternative Path (HTAP) Algorithm for Congestion Control in Wireless Sensor Networks," *Ad Hoc Networks*, vol. 11, no. 1, pp. 257–272, 2013.

[41] H.-S. Kim, J. Paek, and S. Bahk, "QU-RPL: Queue Utilization Based RPL for Load Balancing in Large Scale Industrial Applications," in *Sensing, Communication, and Networking (SECON), 2015 12th Annual IEEE International Conference on*. IEEE, 2015, pp. 265–273.

[42] A. P. Castellani, M. Rossi, and M. Zorzi, "Back Pressure Congestion Control for CoAP/6LoWPAN Networks," *Ad Hoc Networks*, vol. 18, pp. 71–84, 2014.

[43] J.-M. Huang, C.-Y. Li, and K.-H. Chen, "TALONet: A Power-Efficient Grid-Based Congestion Avoidance Scheme Using Multi-Detouring Technique in Wireless Sensor Networks," in *Wireless Telecommunications Symposium, WTS 2009*. IEEE, 2009, pp. 1–6.

[44] H. A. A. Al-Kashoash, Y. Al-Nidawi, and A. H. Kemp, "Congestion-Aware RPL for 6LoWPAN Networks," in *Accepted at Wireless Telecommunications Symposium (WTS), 2016*. IEEE, 2016, pp. 1–6.

[45] J. Wan, X. Xu, R. Feng, and Y. Wu, "Cross-Layer Active Predictive Congestion Control Protocol for Wireless Sensor Networks," *Sensors*, vol. 9, no. 10, pp. 8278–8310, 2009.

[46] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-Aware Fair Rate Control in Wireless Sensor Networks," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 63–74, 2006.

[47] W.-w. Fang, J.-m. Chen, L. Shu, T.-s. Chu, and D.-p. Qian, "Congestion Avoidance, Detection and Alleviation in Wireless Sensor Networks," *Journal of Zhejiang University SCIENCE C*, vol. 11, no. 1, pp. 63–73, 2010.

[48] J.-P. Sheu and W.-K. Hu, "Hybrid Congestion Control Protocol in Wireless Sensor Networks," in *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*. IEEE, 2008, pp. 213–217.

[49] J. Kang, Y. Zhang, and B. Nath, "TARA: Topology-Aware Resource Adaptation to Alleviate Congestion in Sensor Networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 7, pp. 919–931, 2007.

[50] J.-H. Lee and I.-B. Jung, "Adaptive-Compression Based Congestion Control Technique for Wireless Sensor Networks," *Sensors*, vol. 10, no. 4, pp. 2919–2945, 2010.

[51] J. P. Sheu, C. X. Hsu, and C. Ma, "A Game Theory Based Congestion Control Protocol for Wireless Personal Area Networks," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 2, July 2015.

[52] C. Ma, J.-P. Sheu, and C.-X. Hsu, "A Game Theory Based Congestion Control Protocol for Wireless Personal Area Networks," *Journal of Sensors*, vol. 2016, 2015.

[53] H. M. A. Fahmy, "Simulators and Emulators for WSNs," in *Wireless Sensor Networks*. Springer, 2016, pp. 381–491.

[54] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "TinyOS: An Operating System for Sensor Networks," in *Ambient intelligence*. Springer, 2005, pp. 115–148.

[55] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE, 2004, pp. 455–462.

[56] A. Dunkels, "Contiki: Bringing IP to Sensor Networks," *ERCIM News*, no. 76, 2009.

[57] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level Power Profiling for Low-power Wireless Networks," 2011.

[58] E. Baccelli, O. Hahm, M. Gunes, M. Wahlisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 79–80.

[59] H. Will, K. Schleiser, and J. Schiller, "A Real-Time Kernel for Wireless Sensor Networks Employed in Rescue Scenarios," in *IEEE 34th Conference on Local Computer Networks (LCN 2009)*. IEEE, 2009, pp. 834–841.

[60] P. Levis and N. Lee, "TOSSIM: A Simulator for TinyOS Networks," *UC Berkeley, September*, vol. 24, 2003.

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

[62] F. Österlind, J. Eriksson, and A. Dunkels, "Cooja TimeLine: a Power Visualizer for Sensor Network Simulation," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010, pp. 385–386.

[63] I. T. Downard, "Simulating Sensor Networks in NS-2," DTIC Document, Tech. Rep., 2004.

[64] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network Simulations with the ns-3 Simulator," *SIGCOMM demonstration*, vol. 15, p. 17, 2008.

[65] G. Simon, P. Volgyesi, M. Maróti, and Á. Lédeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensor networks," in *IEEE aerospace conference*, vol. 3, 2003, pp. 1339–1346.

[66] X. Chang, "Network simulations with OPNET," in *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*. ACM, 1999, pp. 307–314.

[67] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *Proceedings of the European simulation multiconference (ESM'2001)*, 2001, pp. 185–192.

[68] M. Kirsche and J. Hartwig, "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), 2013, pp. 330–333.

[69] C. T. Ee and R. Bajcsy, "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, 2004, pp. 148–161.

[70] M. Zawodniok and S. Jagannathan, "Predictive Congestion Control Protocol for Wireless Sensor Networks," *Wireless Communications, IEEE Transactions on*, vol. 6, no. 11, pp. 3955–3963, 2007.

[71] S. Chen and Z. Zhang, "Localized Algorithm for Aggregate Fairness in Wireless Sensor Networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*. ACM, 2006, pp. 274–285.

[72] S. Chen and N. Yang, "Congestion Avoidance Based on Lightweight Buffer Management in Sensor Networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 9, pp. 934–946, 2006.

[73] M. M. Monowar, M. O. Rahman, and C. S. Hong, "Multipath Congestion Control for Heterogeneous Traffic in Wireless Sensor Network," in *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, vol. 3. IEEE, 2008, pp. 1711–1715.

[74] G. Wang and K. Liu, "Upstream Hop-by-Hop Congestion Control in Wireless Sensor Networks," in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*. IEEE, 2009, pp. 1406–1410.

[75] M. M. Alam and C. S. Hong, "CRRT: Congestion-Aware and Rate-Controlled Reliable Transport in Wireless Sensor Networks," *IEICE transactions on communications*, vol. 92, no. 1, pp. 184–199, 2009.
[76] S. Brahma, M. Chatterjee, and K. Kwiat, "Congestion Control and Fairness in Wireless Sensor Networks," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 2010, pp. 413–418.
[77] S. R. Heikalabad, A. Ghaffari, M. A. Hadian, and H. Rasouli, "DPCC: Dynamic Predictive Congestion Control in Wireless Sensor Networks," *IJCSI International Journal of Computer Science Issues*, vol. 8, no. 1, 2011.
[78] S. A. Munir, Y. W. Bin, R. Biao, and M. Jian, "Fuzzy Logic Based Congestion Estimation for QoS in Wireless Sensor Network," in *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*. IEEE, 2007, pp. 4336–4341.
[79] J. Wei, B. Fan, and Y. Sun, "A Congestion Control Scheme Based on Fuzzy Logic for Wireless Sensor Networks," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*. IEEE, 2012, pp. 501–504.
[80] T. He, F. Ren, C. Lin, and S. Das, "Alleviating Congestion Using Traffic-Aware Dynamic Routing in Wireless Sensor Networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON'08. 5th Annual IEEE Communications Society Conference on*. IEEE, 2008, pp. 233–241.
[81] A. Woo, T. Tong, and D. Culler, "Taming The Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 14–27.
[82] M. O. Rahman, M. M. Monowar, and C. S. Hong, "A QoS Adaptive Congestion Control in Wireless Sensor Network," in *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, vol. 2. IEEE, 2008, pp. 941–946.
[83] C. Wang, K. Sohraby, and B. Li, "SenTCP: A Hop-by-Hop Congestion Control Protocol for Wireless Sensor Networks," in *IEEE INFOCOM*, 2005, pp. 107–114.
[84] C. Sergiou, V. Vassiliou, and A. Paphitis, "Congestion Control in Wireless Sensor Networks through Dynamic Alternative Path Selection," *Computer Networks*, vol. 75, pp. 226–238, 2014.
[85] R. Dasgupta, R. Mukherjee, and A. Gupta, "Congestion Avoidance Topology in Wireless Sensor Network using Karnaugh Map," in *Applications and Innovations in Mobile Computing (AIMoC), 2015*. IEEE, 2015, pp. 89–96.
[86] M. A. Razzaque and C. S. Hong, "Congestion Detection and Control Algorithms for Multipath Data Forwarding in Sensor Networks," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, vol. 1. IEEE, 2009, pp. 651–653.
[87] C. Sergiou and V. Vassiliou, "HRTC: A Hybrid Algorithm for Efficient Congestion Control in Wireless Sensor Networks," in *New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on*. IEEE, 2014, pp. 1–5.
[88] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936–1948, 1992.
[89] H. Hellaoui and M. Koudil, "Bird Flocking Congestion Control for CoAP/RPL/6LoWPAN Networks," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*. ACM, 2015, pp. 25–30.
[90] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load balancing under heavy traffic in rpl routing protocol for low power and lossy networks," *IEEE Transactions on Mobile Computing*, 2016.
[91] P. Thubert, "Objective Function Zero for The Routing Protocol for Low-Power and Lossy Networks (RPL)," *RFC 6552*, 2012.
[92] O. Gnawali and P. Levis, "The ETX Objective Function for RPL," *Internet Draft: draft-gnawali-roll-etxof-00*, 2010.
[93] W. Tang, X. Ma, J. Huang, and J. Wei, "Toward improved rpl: A congestion avoidance multipath routing protocol with time factor for wireless sensor networks," *Journal of Sensors*, vol. 2016, 2015.
[94] P. O. Kamgueu, E. Nataf, T. D. Ndié, and O. Festor, "Energy-Based Routing Metric for RPL," *[Research Report] RR-8208, INRIA*, p. 14, 2013.
[95] M. A. Lodhi, A. Rehman, M. M. Khan, and F. B. Hussain, "Multiple Path RPL for Low Power Lossy Networks," in *Wireless and Mobile (APWiMob), 2015 IEEE Asia Pacific Conference on*. IEEE, 2015, pp. 279–284.
[96] M. Ha, K. Kwon, D. Kim, and P.-Y. Kong, "Dynamic and Distributed Load Balancing Scheme in Multi-Gateway based 6LoWPAN," in *Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE*. IEEE, 2014, pp. 87–94.
[97] X. Liu, J. Guo, G. Bhatti, P. Orlik, and K. Parsons, "Load Balanced Routing for Low Power and Lossy Networks," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 2238–2243.
[98] J. Guo, X. Liu, G. Bhatti, P. Orlik, and K. Parsons, "Load Balanced Routing for Low Power and Lossy Networks," Jan. 21 2013, US Patent App. 13/746,173.
[99] W. Tang, Z. Wei, Z. Zhang, and B. Zhang, "Analysis and Optimization Strategy of Multipath RPL Based on the COOJA Simulator," *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 5, pp. 27–30, 2014.
[100] H. Al-Kashoash, M. Hafeez, and A. Kemp, "Congestion Control for 6LoWPAN Networks: A Game Theoretic Framework," *IEEE Internet of Things Journal*, 2017.
[101] H. A. Al-Kashoash, H. M. Amer, L. Mihaylova, and A. H. Kemp, "Optimization Based Hybrid Congestion Alleviation for 6LoWPAN Networks," *IEEE Internet of Things Journal*, 2017.
[102] H. A. Al-Kashoash, F. Hassen, H. Kharrufa, and A. H. Kemp, "Analytical Modelling of Congestion for 6LoWPAN Networks," *ICT Express*, 2018.
[103] T. Zheng, A. Ayadi, and X. Jiang, "TCP over 6LoWPAN for Industrial Applications: An Experimental Study," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*. IEEE, 2011, pp. 1–4.
[104] A. Ayadi, P. Maillé, and D. Ros, "TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*. IEEE, 2011, pp. 1–5.
[105] H.-S. Kim, H. Im, M.-S. Lee, J. Paek, and S. Bahk, "A Measurement Study of TCP over RPL in Low-power and Lossy Networks," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 647–655, 2015.
[106] P. Antoniou, A. Pitsillides, T. Blackwell, A. Engelbrecht, and L. Michael, "Congestion Control in Wireless Sensor Networks Based on Bird Flocking Behavior," *Computer Networks*, vol. 57, no. 5, pp. 1167–1191, 2013.
[107] V. Michopoulos, L. Guan, G. Oikonomou, and I. Phillips, "A Comparative Study of Congestion Control Algorithms in IPv6 Wireless Sensor Networks," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. IEEE, 2011, pp. 1–6.
[108] H. A. A. Al-Kashoash, Y. Al-Nidawi, and A. H. Kemp, "Congestion Analysis for Low Power and Lossy Networks," in *Accepted at Wireless Telecommunications Symposium (WTS), 2016*. IEEE, 2016, pp. 1–6.
[109] M. Weldon, *The Future X Network: A Bell Labs Perspective*. CRC Press, March 2016.
[110] J. Dunlap, "From Billing and Technology Convergence to Ecosystem Convergence: Why M2M Matters to Your Business," *Pipeline: Technology for Service Providers*, vol. 8, no. 7, p. 14, 2011.