Congestion Control for 6LoWPAN Networks: A Game Theoretic Framework

Hayder A. A. Al-Kashoash, *Student Member, IEEE,* Maryam Hafeez, *Member, IEEE,* and Andrew H. Kemp, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) has been considered as an emerging research area where the 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Network) protocol stack is considered as one of the most important protocol suite for the IoT. Recently, the Internet Engineering Task Force has developed a set of IPv6 based protocols to alleviate the challenges of connecting resource limited sensor nodes to the Internet. In 6LoWPAN networks, heavy network traffic causes congestion which significantly degrades network performance and effects the quality of service (QoS) aspects e.g. throughput, end-to-end delay and energy consumption. In this paper, we formulate the congestion problem as a non-cooperative game framework where the nodes (players) behave uncooperatively and demand high data rate in a selfish way. Then, the existence and uniqueness of Nash equilibrium is proved and the optimal game solution is computed by using Lagrange multipliers and KKT conditions. Based on this framework, we propose a novel and simple congestion control mechanism called game theory based congestion control framework (GTCCF) specially tailored for IEEE 802.15.4, 6LoWPAN networks. GTCCF is aware of node priorities and application priorities to support the IoT application requirements. The proposed framework has been tested and evaluated through two different scenarios by using Contiki OS and compared with comparative algorithms. Simulation results show that GTCCF improves performance in the presence of congestion by an overall average of 30.45%, 39.77%, 26.37%, 91.37% and 13.42% in terms of throughput, end-to-end delay, energy consumption, number of lost packets and weighted fairness index respectively as compared to DCCC6 algorithm.**

***Index Terms*—Congestion control, rate adaptation, non-cooperative game theory, 6LoWPAN networks, IoT applications.**

I. Introduction

THE IoT is considered to be the next big opportunity and challenge for the Internet research community [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 where they can collaborate and provide services such as smart environments, health care, etc. [2].

H. A. A. Al-Kashoash is with the Electronic and Electrical Engineering School, University of Leeds, Leeds LS2 9JT, U.K., and also with Technical Institute/Qurna, Southern Technical University, Basra, Iraq (e-mail: ml14haak@leeds.ac.uk; hayderaam@gmail.com).

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

Wireless sensor networks (WSNs) are considered as one of the most important elements in the IoT [3]. 6LoWPANs [4] are used for full integration of WSN 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 WSN and 6LoWPAN networks has many issues and problems due to the limitation of bandwidth, energy and buffer resources. TCP requires extra resources for connection setup and termination before and after the data transmission whilst UDP does not provide a congestion control mechanism. Thus, TCP and UDP are not efficient for WSN and 6LoWPAN networks [1], [2]. Therefore, one of the main issues in WSN and 6LoWPAN networks is congestion that causes packet loss, increased energy consumption and degraded throughput.

WSNs connected to the Internet through 6LoWPAN have wide applications in industrial, automation, healthcare, military, environment, logistics, etc. An estimate by Bell Labs suggests that from 50 to 100 billion things are expected to be connected to the Internet by 2020 [5], and the number of the wireless sensor devices will account for a majority of these. Generally, the applications can be categorized into four types: event-based, continuous, query-based and hybrid applications based on the data delivery method [6]. In the hybrid application type, the first three categories are combined into hybrid application i.e. sensor nodes send packets in response to an event (event-based) and at the same time send packets periodically (continuous) as well as send a reply to a sink query (query-based). This type of application will be common in the future as WSNs are integrated with the Internet to form the IoT [2]. In the IoT applications, the sensor nodes host many different application types simultaneously (event-based, continuous and query-based) with varied requirements. Some of them are real-time applications where the application data is time critical and delay constrained, while others are non-real time applications. Some applications send very important data and losing this data is not permitted e.g. medical applications and fire detection applications. This brings new challenges to the congestion control algorithms and mechanisms designed to be aware of application priorities as well as node priorities. However, according to our best knowledge; none of the existing congestion control literature in WSNs and 6LoWPAN networks supports awareness of both node priorities and application priorities. To address this, later we define a 'priority cost function' to support node priority awareness and distinguish between high priority nodes and low priority nodes.

In 6LoWPAN networks, every node selects its parent based on RPL (IPv6 routing protocol for low-power and lossy networks) [7] where there are three types of nodes: sink node, intermediate node and leaf node. When congestion occurs, the leaf nodes start to send high data rate packets to their parent node where each leaf node wants to send packets as high as it can in a selfish way without considering the remaining channel capacity, the available parent's buffer space and the other leaf nodes' sending rate. This problem can be formulated as a non-cooperative game where each selfish leaf node is modelled as a player in the game. To the best of our knowledge, none of the existing work in the congestion control literature of WSNs and 6LoWPAN networks uses game theory to solve the congestion problem through traffic control (rate adaptation). However, the non-cooperative game theory gives a natural and suitable framework to study and formulate the congestion control problem in 6LoWPAN networks where the nodes (players) are non-cooperative in their behaviors and each node demands high data rate in a selfish way. Also, the non-cooperative game theory provides an optimal solution concept, which is Nash equilibrium, where each player (node) plays a strategy (sending rate) to maximize its payoff given the strategies of other players.

This paper is motivated by these considerations to propose a new congestion control algorithm called "**G**ame **T**heory based **C**ongestion **C**ontrol **F**ramework" (GTCCF) which uses the non-cooperative game theory framework to solve the congestion problem and is aware of both node priorities and application priorities to support the IoT application requirements. Our main contributions in this paper include:

- Design a congestion control game for mitigating congestion in 6LoWPAN networks. The node's payoff function is formulated to achieve the node demand (preference) for sending high data rate (utility function) and the desirable fairness among leaf nodes according to their priorities (priority cost function), while alleviating and mitigating congestion in the network (congestion cost function).
- Prove the existence and uniqueness of Nash equilibrium in the formulated congestion control game. Also, the node's payoff function is modelled as a constrained nonlinear optimization problem which is solved by using Lagrange multipliers and KKT (Karush-Kuhn-Tucker) conditions such that each node obtains its optimal solution (sending rate) that satisfies the congestion alleviation.
- By using the formulated game, we propose a novel and simple congestion control algorithm called GTCCF which is aware of node priorities and application priorities to support the IoT application requirements. Also, the proposed framework is designed and built on the unique characteristics of the IEEE 802.15.4 standard, IPv6 and 6LoWPAN protocol stack.
- Implement and evaluate the performance of the proposed framework in the real IoT operating system, Contiki OS [8], through Cooja simulator [9].

The remainder of the paper is organized as follows: in section II, we provide a review of related work on congestion control in 6LoWPAN networks. Section III introduces a non-cooperative game framework for congestion control, proves the existence of a unique Nash equilibrium and computes the optimal solution for the designed game. The implementation of the congestion control game in 6LoWPAN networks is provided in section IV. In section V, simulation scenarios and results are given. Finally, section VI draws conclusions.

II. Related Work

Numerous algorithms have been proposed in the congestion control literature for mitigating congestion in WSNs (see [6] and references therein). However, the most of the existing literature do not take into account the unique characteristics of the IEEE 802.15.4 standard, IPv6 and 6LoWPAN protocol stack (i.e. RPL routing protocol, the adaptation layer and IEEE 802.15.4 MAC and PHY layers). Recently, a number of papers suggest new congestion control algorithms for 6LoWPAN networks. A short review of these mechanisms is given below. However, according to our best knowledge, none of the proposed algorithms in congestion control literature for WSNs and 6LoWPAN networks: (i) uses game theory for traffic control (rate adaptation) [10], [11] to solve the congestion problem (the work in [12] and [13] (the content of these two papers overlaps) use game theory for parent selection (routing)) and (ii) supports and is aware of both node priorities and application priorities. However, the non-cooperative game theory provides an analytical framework suited for characterizing the interactions and decision making process among several players with conflicting interests [14]. Therefore, in this work, we use the non-cooperative game theory framework to solve and mitigate the congestion problem. Moreover, this is the first work that is aware of both node priorities and application priorities to support the IoT application requirements where each node is assigned a priority based on its importance and hosted application types as well as each application is given a priority according to its type (i.e. real-time application or not, time-critical application or not, etc.).

In [15], Michopoulos et al. proposed a new congestion control algorithm called Duty Cycle-Aware Congestion Control (DCCC6) for 6LoWPAN networks. The proposed algorithm detects the presence of radio duty cycle and adjusts its operation accordingly. The proposed protocol uses a dynamic buffer occupancy as a congestion detection method as well as a modified AIMD (Additive-Increase Multiplicative-Decrease) to reduce the congestion in the network. In [16], Castellani et al. proposed three different congestion control schemes called Griping, Deaf and Fuse for controlling unidirectional and bidirectional data flows in (Constrained Application Protocol) CoAP/6LoWPAN networks. The proposed algorithms are based on distributed back pressure concept. The proposed algorithms use a buffer occupancy strategy (in Griping) and missing acknowledgement packet (in Deaf and Fuse) to detect the congestion as well as AIMD scheme to mitigate the congestion by adjusting the transmission rate to reduce the injected packets into the network.

In [17], Hellaoui and Koudil proposed a congestion control solution for CoAP/6LoWPAN networks. The proposed algorithm is based on a bird flocking concept to pass packets through uncongested areas and avoid congested ones. 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 packets to the destination (sink node). In [18], [19], Kim et al. proposed an effective queue utilization based RPL algorithm called (QU-RPL). QU-RPL uses the queue utilization 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 (DODAG Information Object) 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 sink node. Otherwise, without congestion, the node chooses its best parent based on the same parent selection mechanism of the default RPL.

In [12] and [13], the authors proposed a congestion control mechanism called Game Theory Congestion Control (GTCC) for 6LoWPAN networks. 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. In [20], 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 (expected transmission count), rank and number of received packets for parent selection process. In [21], 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.

III. Game Theoretic Formulation

A. Network Setup and Problem Formulation

In 6LoWPAN networks, the RPL routing protocol [7] is responsible for constructing the network topology where three types of nodes are defined: sink (root) node which provides connectivity to other networks, intermediate node which forwards packets to the sink and leaf node. The construction of network topology is based on the DAG (Directed Acyclic Graph) concept where every node selects a neighbour as its parent based on an objective function. RPL organises nodes as Destination Oriented DAGs (DODAG) where a sink node

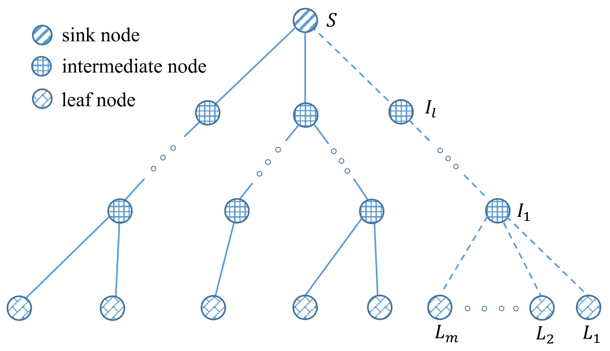

Fig. 1. RPL based network topology

works as the root of the DAG which is responsible to start forming the network topology. The DAG root broadcasts a DIO control message to other nodes in the network. When an intermediate node receives the DIO message, it replies to the sink node with DAO (Destination Advertisement Object) for joining the DODAG. Then, the intermediate node sends a DIO message to all neighbours. This process continues until the DIO message reaches the leaf nodes. When a node receives a DIO message from more than one neighbour, it selects its parent with a best rank. Also, when a node does not receive a DIO message within a specific time, it sends a DIS (DODAG Information Solicitation) message to solicit DIO message from neighbours. The formed network topology is shown in Fig. 1.

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

In 6LoWPAN networks, when congestion occurs, the leaf nodes $(L_1, L_2, \ldots, L_m)$ start to send high data rate packets to their parent (I_1) in a selfish way where each leaf node wants to send as many packets as it can without taking into account the available channel bandwidth, the buffer occupancy of the parent, the forwarding (service) rate of the parent node and sending rate of other leaf nodes. This will increase packet loss, energy consumption and end-to-end delay, decrease the network performance and throughput and impact on the QoS aspects. These selfish leaf nodes and their parent can be modelled as the following non-cooperative game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$ where:

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

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

- Utility function: we use $U_k(\lambda_k)$ to represent the utility function of player L_k where λ_k is sending rate (strategy) of player L_k. The utility function is designed such that each player gets more profit by increasing its sending rate. Many types of utility function are commonly used such as exponential, logarithmic, linear and sigmoidal [22]. In our framework, we use the logarithmic utility function as it has strict concavity property. Thus, we select the utility function of player L_k as follows:
$$U_k(\lambda_k) = \log(\lambda_k + 1) \tag{1}$$
- Congestion cost function: we use $C_k(\lambda_k, \lambda_{-k})$ to represent the congestion cost of node (player) L_k where $\lambda_{-k} = [\lambda_j]_{j \in M; j \neq k}$ is the vector of sending rates (strategies) of all players except player L_k and $s = (\lambda_k, \lambda_{-k}) \in S$ is referred to as the strategy profile. This function reflects how much the parent node is congested due to the leaf nodes. According to Queuing Theory; if the arrival rate at the parent node's buffer is higher than the service rate from the parent, the buffer starts overflowing the packets and congestion occurs. Thus, one possible method is to choose the congestion cost function as the ratio between the total receiving rate and total forwarding rate at the parent's buffer. As the receiving rate is greater than the forwarding rate, the ratio increases. Also, the number of leaf nodes has an impact on congestion. As the number of leaf nodes, m, increases, the congestion situation becomes worse at the parent. Assume that a number of sending packets from the leaf nodes are lost on the wireless channel before they arrive to the parent node with a probability of $P_k^{channel-loss}$; $\forall k \in M$. Thus, the congestion cost function can be defined as follows:
$$C_k(\lambda_k, \lambda_{-k}) = m \frac{\sum_{k=1}^{m} (1 - P_k^{channel-loss}) \lambda_k + 1}{\lambda_{out} + 1} \tag{2}$$
where λ_{out} is the outgoing rate from the parent node such that $\lambda_{out} \geq 0$.
In [23], congestion analysis for 6LoWPAN networks with different parameters and various scenarios was explored. It demonstrated that the majority of packets are lost in the nodes' buffer as compared to wireless channel loss when congestion occurs. For example, with high offered load (i.e. 8 packets/second), the percentage of packet loss due to buffer overflow is up to 99.66% compared to 0.33% due to channel loss. Therefore, to simplify the analysis, we assume that $P_k^{channel-loss}$ in equation (2) is zero; $\forall k \in M$. Thus, $C_k(\lambda_k, \lambda_{-k})$ becomes as follows:
$$C_k(\lambda_k, \lambda_{-k}) = m \frac{\lambda_{in} + 1}{\lambda_{out} + 1} \tag{3}$$
where $\lambda_{in} = \sum_{k=1}^{m} \lambda_k$, $\lambda_{out} \geq 0$ and $0 \leq \lambda_k \leq \lambda_k^{max}$ for all $k \in M$.
Remark 3.1: We add 1 to λ_k in equation (1) and to λ_{out} in the denominator of equation (2) to avoid making the values of utility function and congestion cost function equal to $-\infty$ and ∞ respectively. Since the value of λ_k ranges from zero to λ_k^{max} and the value of λ_{out} is greater than or equal to zero; therefore, without adding 1, $U_k(\lambda_k) = -\infty$ when $\lambda_k = 0$ and $C_k(\lambda_k, \lambda_{-k}) = \infty$ when $\lambda_{out} = 0$ for all $k \in M$.
- Priority cost function: we use $P_k(\lambda_k; p_k)$ to represent the priority cost function of player L_k; $\forall k \in M$. Player L_k has to pay a penalty based on its priority (p_k) and its sending rate (λ_k) to distinguish between high priority nodes and low priority nodes. A player with less p_k value has high priority (e.g. if $p_i = 1$ and $p_j = 2$, this means that player L_i has higher priority than player L_j). Therefore, the priority cost function of player L_k can be defined as follows:
$$P_k(\lambda_k; p_k) = p_k \lambda_k \tag{4}$$

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

$$\Phi_k(\lambda_k, \lambda_{-k}) = \omega_k \log(\lambda_k + 1) - \alpha_k m \frac{\lambda_{in} + 1}{\lambda_{out} + 1} - \beta_k p_k \lambda_k \tag{5}$$
where ω_k, α_k and β_k are player preference parameters of functions $U_k(\lambda_k)$, $C_k(\lambda_k, \lambda_{-k})$ and $P_k(\lambda_k; p_k)$ respectively such that ω_k, α_k, $\beta_k > 0$; $\forall k \in M$. The values of ω_k, α_k and β_k are chosen by user to satisfy the system objective and requirement. For example, as the value of β_k is greater, the difference between sending rate (λ_k) of high priority node and low priority node is higher and vice versa.

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

Lemma 3.2: In the congestion control game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$, $\forall k \in M$, every strategy set S_k is compact and convex, $\Phi_k(\lambda_k, \lambda_{-k})$ is continuous function in the profile of strategies $s \in S$ and concave in S_k; then, the game G has at least one Nash equilibrium.

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

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

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

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

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

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

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

$$H(s) = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mm} \end{bmatrix} \tag{6}$$

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

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

$$A_{k,j} = \begin{cases} -\dfrac{\omega_k}{(\lambda_k + 1)^2} < 0 & \text{if } k = j; \forall k, j \in M \\ 0 & \text{if } k \ne j; \forall k, j \in M \end{cases} \tag{7}$$

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

According to the Nikaido Isoda theorem [24], these conditions (in *Lemma* 3.2) are sufficient to satisfy the existence of at least one Nash equilibrium in the game G. ■

Lemma 3.3: The congestion control game $G = (M, (S_k)_{k\in M}, (\Phi_k)_{k\in M})$ admits unique Nash equilibrium in its pure strategy space.

Proof: Let $r = (r_1, r_2, ..., r_m)$ be an arbitrary vector of fixed positive parameters. Based on Rosen's Theorem (Theorem 2) [25], we define the weighted nonnegative sum of the payoff functions $\Phi_k(\lambda_k, \lambda_{-k})$; $\forall k \in M$ as follows:

$$\sigma(\lambda_k, \lambda_{-k}; r) = \sum_{k=1}^{m} r_k \Phi_k(\lambda_k, \lambda_{-k}), \quad r_k \ge 0 \tag{8}$$

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

$$g(\lambda_k, \lambda_{-k}; r) = \begin{bmatrix} r_1 \nabla\Phi_1(\lambda_1, \lambda_{-1}) \\ r_2 \nabla\Phi_2(\lambda_2, \lambda_{-2}) \\ \vdots \\ r_m \nabla\Phi_m(\lambda_m, \lambda_{-m}) \end{bmatrix} \tag{9}$$

where $\nabla\Phi_k(\lambda_k, \lambda_{-k}) = \dfrac{\omega_k}{\lambda_k + 1} - \alpha_k m \dfrac{1}{\lambda_{out} + 1} - \beta_k p_k$, $\forall k \in M$.

Now, we define the Jacobian matrix $(G(\lambda_k, \lambda_{-k}; r))$ of $g(\lambda_k, \lambda_{-k}; r)$ with respect to λ_k as follows:

$$G(\lambda_k, \lambda_{-k}; r) = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1m} \\ B_{21} & B_{22} & \dots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mm} \end{bmatrix} \tag{10}$$

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

Now, it is clear that the symmetric matrix $[G(\lambda_k, \lambda_{-k}; r) + G^T(\lambda_k, \lambda_{-k}; r)]$ is negative definite for all $\lambda_k, \lambda_{-k} \in S$. Then, Rosen's Theorem (Theorem 6) [25] states that the function $\sigma(\lambda_k, \lambda_{-k}; r)$ is diagonally strictly concave. Therefore, according to Rosen's Theorem (Theorem 2) [25], the game G has unique Nash equilibrium in its pure strategy space. ■

B. Game Solution Computation

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

$$\begin{aligned} \underset{\lambda_k \in S_k}{\text{maximize}} \quad & \Phi_k(\lambda_k, \lambda_{-k}) \\ \text{subject to} \quad & \lambda_k \ge 0 \\ & \lambda_k \le \lambda_k^{max}, \ \forall k \in M. \end{aligned} \tag{11}$$

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

$$\mathcal{L}_k = \Phi_k(\lambda_k, \lambda_{-k}) + u_k \lambda_k + v_k(\lambda_k^{max} - \lambda_k) \tag{12}$$

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

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

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

$$\lambda_k^* = \begin{cases} 0 & \text{if condition 1} \\ \lambda_k^{max} & \text{if condition 2} \\ \dfrac{\omega_k(\lambda_{out} + 1)}{\alpha_k m + \beta_k p_k(\lambda_{out} + 1)} - 1 & \text{otherwise} \end{cases} \tag{13}$$

where condition 1 and condition 2 respectively are:

$$\frac{\alpha_k m}{\lambda_{out} + 1} + \beta_k p_k \ge \omega_k \tag{14}$$

$$\frac{\alpha_k m}{\lambda_{out} + 1} + \beta_k p_k \le \frac{\omega_k}{\lambda_k^{max} + 1} \tag{15}$$

C. Distribution of Node's Sending Rate among Applications

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

$$\lambda_k^j = \theta_j \lambda_k^* \tag{16}$$

$$\theta_j = \begin{cases} 1 & \text{if } n = 1 \\ \dfrac{\sum\limits_{i=1; i \neq j}^{n} p_k^i}{(n-1) \sum\limits_{i=1}^{n} p_k^i} & \text{if } n > 1 \end{cases} \tag{17}$$

$$\sum_{j=1}^{n} \theta_j = 1 \tag{18}$$

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

IV. Game Theory Framework Implementation

In 6LoWPAN networks, the network topology is governed by RPL routing protocol through transmission of DIO, DAO and DIS control messages. The DIO transmission strategy is controlled by the "Trickle Algorithm" [26] where the Trickle timer is set to the minimum interval size, I_{min}, and it is doubled after the timer expires until it reaches to maximum interval size, I_{max}. Therefore, the Trickle algorithm is not aware of the occurrence of congestion. Thus, the operation of the algorithm is modified such that when congestion occurs at the parent node, the DIO packet is immediately sent and congestion information is piggybacked on it.

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

$$\lambda_k^{(initial)} = \frac{\lambda_k^{max}}{p_k}; \qquad \forall k \in M \tag{19}$$

The parent node periodically checks the congestion conditions every interval time 'I_{check}'. The value of I_{check} has to be in the right range (e.g. typically 1, 2 or 3 seconds). Below this range, the adaptation of the sending rate fluctuates wildly and also this will increase the number of overhead DIO notification packets sent. If the value of I_{check} is too large, congestion may occur and the node will not check frequently enough. According to Queuing Theory, if the arrival rate (λ_{in}) at the parent's buffer is higher than service rate (λ_{out}), the parent's buffer will be blocked and congestion does occur. As a result, the parent node broadcasts a DIO packet which contains the congestion cost function information. The forwarding rate of parent λ_{out} is not constant with time. It is increased or decreased due to the operation of the CSMA algorithm (i.e. backoff time), MAC parameters (i.e. channel check rate) and number of active nodes. Thus, to avoid sending high overhead DIO packets and fluctuating the sending rate of leaf nodes, we use Brown's simple exponential smoothing model [27] to estimate the actual maximum sending rate as follows:

$$\lambda_{out}(t+1) = \psi \lambda_{out}(t) + (1 - \psi) \lambda_{out}(t-1) \tag{20}$$

where $\lambda_{out}(t+1)$, $\lambda_{out}(t)$ and $\lambda_{out}(t-1)$ are the expected, current and historical forwarding rate of the parent respectively and ψ is smoothing factor such that $0 < \psi < 1$. A large value of ψ reduces the level of smoothing and gives high weight to current measurement of λ_{out}, while a value of ψ close to zero gives greater smoothing effect and less responsive to recent changes in λ_{out} value. In this paper, we set the value of ψ to 0.4. Also, the parent node sends DIO packet when the number of leaf nodes, m, changes because the optimal sending rate (Nash equilibrium) of each leaf node will change. When the leaf nodes receive the DIO message, they update their sending rate according to equation (13) where the parameters ω_k, α_k, β_k and p_k are already known to the player L_k; $\forall k \in M$. After that, the leaf node distributes the updated sending rate (λ_k) among the hosted applications according to their priorities as in equations (16) and (17). Algorithm 1 shows the procedures of GTCCF.

Algorithm 1 Congestion control framework

```
1: Input:
   $\omega_k$ preference parameter of $U(\lambda_k)$
   $\alpha_k$ preference parameter of $C(\lambda_k, \lambda_{-k})$
   $\beta_k$ preference parameter of $P(\lambda_k; p_k)$
   $\lambda_k^{max}$ maximum sending rate
   $I_{check}$ congestion check interval time
   $\psi$ smoothing factor
2: Output:
   An optimal sending rate to eliminate congestion
3: At each parent:
   timer_set(congestion_timer,$I_{check}$);
   If (timer_expired(congestion_timer)) then
       If ($\lambda_{out} < \lambda_{in}$ or $m$ changes) then
           DIO.send();
       End
       timer_reset(congestion_timer);
   End
4: At each leaf:
   $p_k \leftarrow$ priority of node $L_k$;
   $p_k^j \leftarrow$ priority of application $j$;
   $\lambda^{initial} \leftarrow$ equation (19);
   If (a new DIO message is received) then
       $\lambda_k^* \leftarrow$ equation (13);
       $\lambda_k^j \leftarrow$ equation (16);
   End
```

V. Performance Evaluation

The proposed congestion control framework has been tested and evaluated on different network scenarios through simulation by using the Contiki 3.0 OS and Cooja simulator. In

TABLE I
PROTOCOL STACK AND SIMULATION PARAMETERS

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

related work, four proposed algorithms exist that use traffic control strategies. These algorithms are: DCCC6 [15], Griping [16], Deaf [16] and Fuse [16]. The working principle of Deaf and Fuse algorithms is based on ACK packet loss as the congestion indicator. However, it is impractical to use ACK packet loss to detect congestion in the network because other reasons for missing ACK exist such as packet error in the wireless channel. Therefore, our proposal is compared with DCCC6 and Griping. In the simulation, we have used one sink node, a set of intermediate nodes and a group of leaf nodes which at the beginning, start sending packets at high data rate (6 packets/s) to create a congested situation. During the simulation, the leaf nodes start sending packets after 60s so the network topology construction is completed where the simulation time is set to 600s. Cooja simulates the hardware of a set of real sensor nodes such as Tmote Sky which is used in the simulation. Also, Cooja simulator implements a number of wireless channel models such as Unit Disk Graph Medium (UDGM) - Distance Loss which is used in the simulation since interference is considered [28]. We use Powertrace [29] to measure the energy consumption of each node where it is a run-time network-level power profiling system that uses state tracking to estimate the energy consumption and it is accurate up to 94%. The protocol stack and simulation parameters used in the simulation are shown in Table I. For our proposal, we have set $I_{check} = 384$ clock ticks, $\omega_k = 15$, $\alpha_k = 7$, $\beta_k = 0.9$, $\psi = 0.4$ and $\lambda_k^{max} = 8$ packet/s; $\forall k \in M$ where each 128 clock ticks = 1 second.

A. DCCC6 and Griping Implementation

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

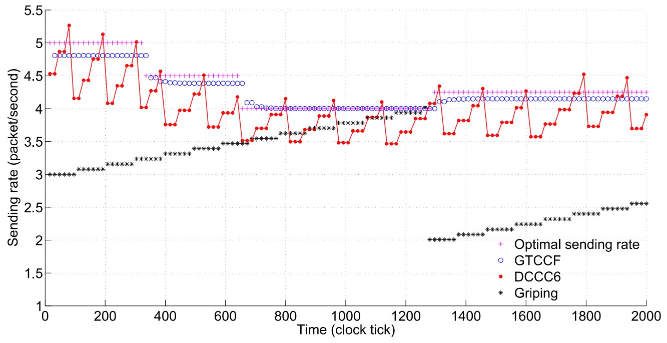

Fig. 2. Sending rate adaptation comparison

DCCC6 detects congestion by using a dynamic buffer occupancy threshold similar to the one use in [30] where the buffer is monitored per incoming packet as follows:

$$threshould(k) = threshold(k-1) + \frac{I}{2^{k-1}} \tag{21}$$

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

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

$$t_{i+1} = t_i + \alpha = t_i + \frac{\gamma \times \sqrt{t_{max}}}{\sqrt{t_i}} \tag{22}$$

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

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

$$t_{i+1} = t_i - \frac{t_i}{\delta} \tag{23}$$

$$\delta = \frac{\beta \times t_i \times \sqrt{n_1 + 1}}{(\epsilon \times \sqrt{t_{min}}) - \sqrt{t_i}} \tag{24}$$

where t_{min} is a minimum inter-packet interval, n_i is the number of active children and $\beta > 1$. In the simulation, we set $\beta = 4$ and according to Table 5.1 in [31], for channel check rate = 8, $t_{min} = 16$ and $\epsilon = 21.8$.

For Griping, when a node receives a new packet, it checks its queue length. If the queue length is greater than a threshold, Q_{thr}, the node sends back a control message. However, the receiver cannot send more than one control message to the same sender during K seconds. Whenever the sender receives the control message, it halves its transmission rate. If no control message has been received during T seconds, the sender increments its transmission rate. According to [16], we set $Q_{thr} = 6$ packets, $k = 13$ clock ticks and $T = 96$ clock ticks.

B. Sending Rate Adaptation Comparison

Fig. 2 compares the rate adaptation mechanisms used in Griping, DCCC6 and GTCCF. Firstly, Griping algorithm employs the original AIMD policy for controlling the sending rate where the rate is increased linearly by a small fixed step every T seconds. Once congestion occurs, the rate is decreased to half and then again linearly increased. Secondly, DCCC6 algorithm uses a modified AIMD mechanism where the sending rate is increased by a variable step every t_i. For example, at time 1168 clock tick, the rate is increased from 3.5 to 3.65 (increasing step = 0.15); whereas at time 1360 clock tick, the increasing step is 0.35. On the other hand, the decreasing step is variable and smaller than the step of the original AIMD. Finally, in GTCCF algorithm, game theory is applied adapting the sending rate where the rate is calculated when congestion occurs or the number of leaf nodes changes. From this figure, it is obvious that the sending rate in GTCCF is closer to the optimal sending rate than others. Also, the modified AIMD used in DCCC6 can be seen to have better rate adaptation than the original AIMD mechanism used by Griping.

C. Scenario 1

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

Fig. 3 shows the number of received packets every second from the leaf nodes at the sink. For GTCCF, it is clear that the node (L_1) with higher priority has the highest number of received packets (≈ 1.4 packet/s) as compared to other nodes, whereas the node L_3 has the lowest number of received packets (≈ 0.75 packet/s) as it has lower priority than others. For DCCC6 and Griping, the nodes do not obtain sending rates according their priorities for example, in DCCC6, the node L_2 has higher sending rates than others, while the node L_1 has the highest priority. The reason is that GTCCF is aware of node priorities where each node gets sending rate according to its priority; however, DCCC6 and Griping do not consider the node priorities in their operation. Also, from this figure, we can see that GTCCF has stable performance (number of received packets at sink) with time as GTCCF computes the optimal sending rate (Nash equilibrium) for each leaf node and this rate is still stable unless the number of leaf nodes changes or the service rate at the intermediate node is less than the incoming rate. On the other hand, DCCC6 has fluctuating sending rate. The reason is that DCCC6 uses modified AIMD where the sending rate is continuously increased every inter-packet interval (t_i) by a variable amount and decreased by α when congestion does occur and then it starts increasing every t_i. While, Griping has the lowest throughput per leaf node as it uses the original AIMD where the sending rate is incremented

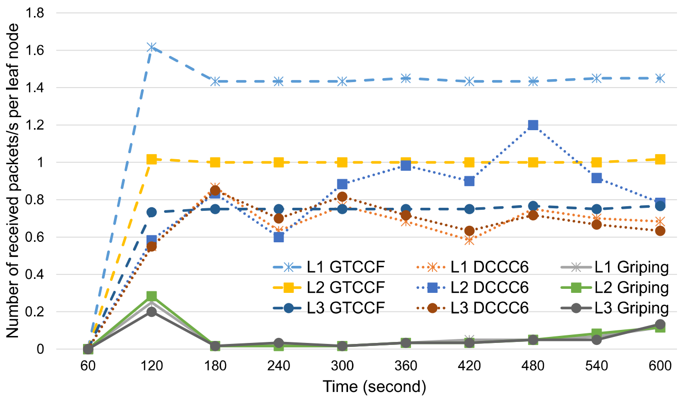

Fig. 3. Number of received packets/s from leaf nodes at sink

every interval time by a small fixed step and decreased to half when congestion occurs. Also, Fig. 3 shows that the modified AIMD used in DCCC6 has better performance in term of throughput than the original AIMD used in Griping.

Fig. 4 shows the overall throughput which is the total number of received packets every second at the sink node. It is clear that GTCCF has stable and higher throughput as compared to other algorithms as well as DCCC6 has better throughput than Griping algorithm for the same reasons stated above. Fig. 5 shows the sending rate of applications hosted in the leaf nodes for GTCCF where L_1 and L_2 host two applications each and L_3 hosts only one. It is obvious that each node distributes its sending rate among hosted applications according to their priorities. For example, in the node L_1, application 1 (App.1) obtains high sending rate ($\approx$ 1.1 packet/s) as compared to application 2 (App.2) ($\approx$ 0.35 packet/s) which has low priority. While, the node L_3 allocates all its sending rate to application 1 as it is hosted alone.

Fig. 6 shows end-to-end delay which is the time between a packet being generated at the application of the source until its successful reception at the application of the final destination. It is clear that GTCCF and Griping have lower end-to-end delay as compared to DCCC6. In GTCCF and Griping, initially; when congestion occurs, the delay is high because the buffer is full so packet waiting time in the buffer is high. After that, when each node computes its optimal sending rate (in GTCCF) or halves its sending rate (in Griping), the delay of packets will decrease. On the other hand, DCCC6 has higher delay than other algorithms because the nodes' sending rates are increased periodically every t_i and decreased when congestion occurs and then increased and this process continues. As a result, the packets wait a long time in the nodes' buffers.

Fig. 7 shows the energy consumption due to transmission and reception in the leaf and intermediate nodes per successfully delivered packet (i.e. energy consumption per packet = total energy consumption due to Tx and Rx / total number of received packets at sink). We note that with GTCCF, the energy consumption in the network is less than others as DCCC6 and Griping waste energy by transmitting and receiving packets which are then lost due to buffer overflow on the path without successful delivery. Also, the consumed

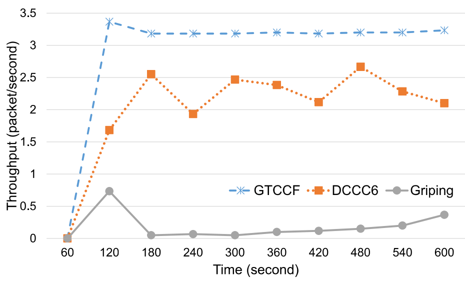

Fig. 4. Number of received packets/second at sink

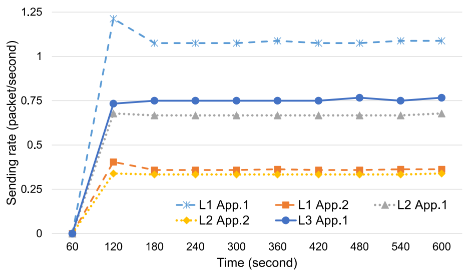

Fig. 5. Applications' sending rate for GTCCF

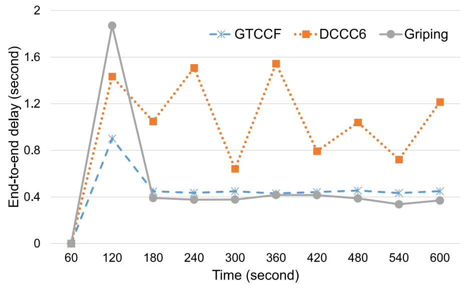

Fig. 6. End-to-end delay

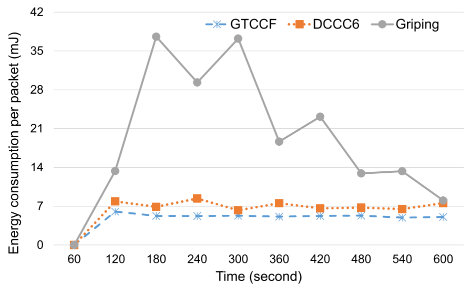

Fig. 7. Energy consumption per successful packet

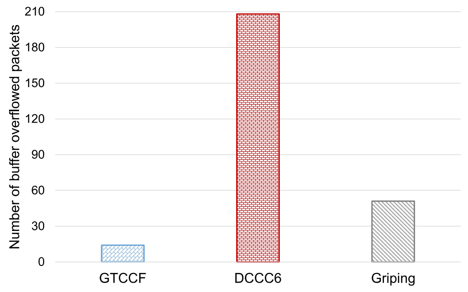

Fig. 8. Number of lost packets

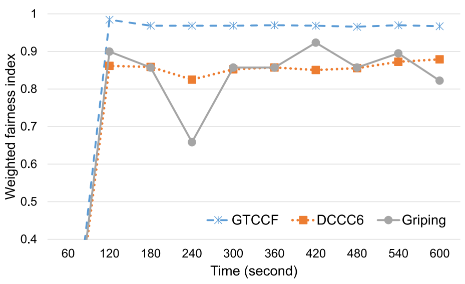

Fig. 9. Weighted fairness index in scenario 1

energy per packet in Griping is significantly higher than others as the number of delivered packets to sink in Griping is much lower than others. Fig. 8 shows the total number of lost packets in the network due to buffer overflow. It is obvious that GTCCF loses less packets at the buffer than others. GTCCF loses packets at the beginning and after the optimal sending rates (Nash equilibrium) are computed, the number of lost packets due to buffer overflow becomes zero. However, the number of lost packets in DCCC6 is higher than Griping algorithm as the sending rates are increased by a small step in Griping whereas by a large step in DCCC6.

Fig. 9 shows the weighted fairness index (WFI) which is an indication of how much the nodes associated with a parent are treated fairly according to their priorities. We measure this performance metric to show and determine whether the algorithms achieve a fair allocation of the network resources (i.e. throughput) among nodes. We have calculated this metric similar to that used in [32] as follows:

$$WFI = \frac{\left[\sum_{k=1}^{m} th_k p_k\right]^2}{m \sum_{k=1}^{m} (th_k p_k)^2} \tag{25}$$

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

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

TABLE II
ALGORITHMS PERFORMANCE SUMMARIZATION IN SCENARIO 1

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

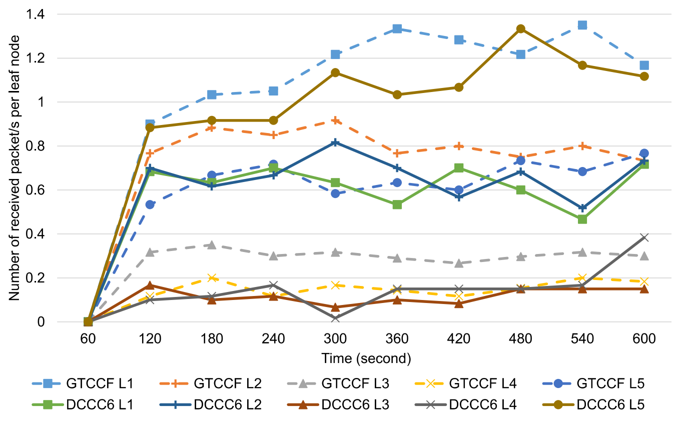

Fig. 10. Number of received packets/s from leaf nodes at sink

D. Scenario 2

In the second scenario, we use a multihop network with one sink node, 15 intermediate nodes and 5 leaf nodes distributed

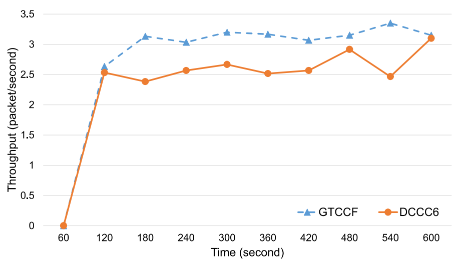

Fig. 11. Number of received packets/second at sink

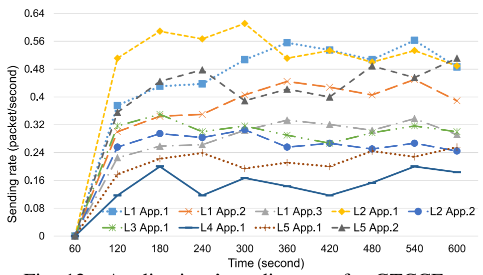

Fig. 12. Applications' sending rate for GTCCF

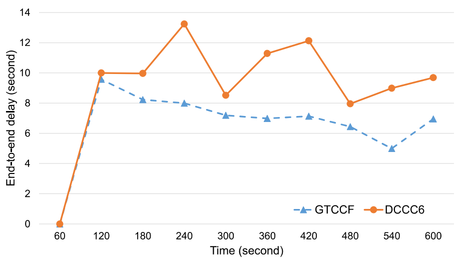

Fig. 13. End-to-end delay

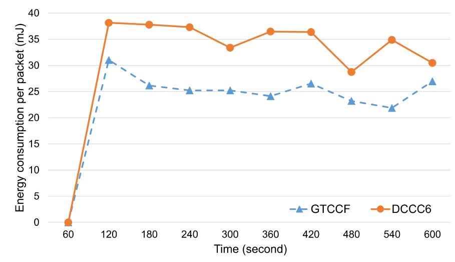

Fig. 14. Energy consumption per successful packet

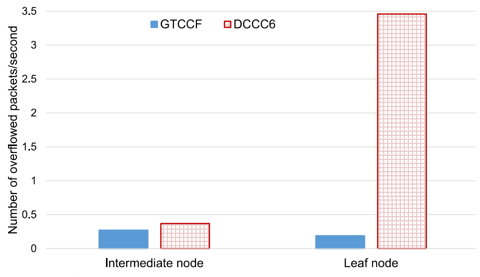

Fig. 15. Number of lost packets

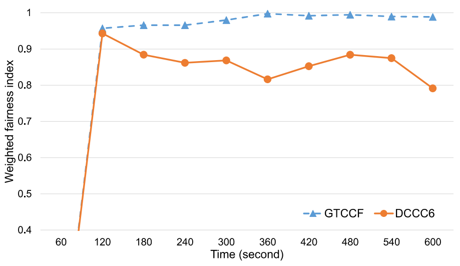

Fig. 16. Weighted fairness index in scenario 2

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

Fig. 10 shows the number of received packets from each leaf node every second at the sink node. For GTCCF, the number of received packets from L_1 (≈ 1.1 packet/s) is higher than node L_2 (≈ 0.8 packet/s) as it has higher priority. Similarly, L_3 has higher number of received packets (≈ 0.3 packet/s) at sink than L_4 (≈ 0.15 packet/s). On the other hand, for DCCC6, the number of received packets from node L_1 and L_2 is approximately the same (≈ 0.6 packet/s) and from L_3 and L_4 is also the same (≈ 0.1 packet/s). Also, from this figure, we can see that the number of received packets from nodes L_1 and L_2 is higher than nodes L_3 and L_4. The reason is that the forwarding rate of parent (P_1) is higher than parent (P_2) as P_1 is located nearer to the sink than P_2. Fig. 11 shows overall throughput which is the total number of received packets at the sink every second. It is obvious that GTCCF has better throughput than DCCC6 for the same reasons stated in scenario 1. Fig. 12 shows the sending rate (packet/second) for the applications hosted in the leaf nodes for GTCCF algorithm. It is clear that each leaf node distributes its sending rate among its applications according to their priorities. For example, the average sending rates of applications 1, 2 and 3 hosted in node L_1 are 0.488, 0.39 and 0.29 packet/s respectively.

TABLE III
ALGORITHMS PERFORMANCE SUMMARIZATION IN SCENARIO 2

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

Fig. 13 shows the end-to-end delay which is the time in second since a packet is generated at the leaf node until its arrival at the sink node. From this figure, it is obvious that GTCCF has lower end-to-end delay than DCCC6 algorithm for the same reasons stated in scenario 1. Fig. 14 shows the energy consumption per successfully received packet (in mJoule) in the leaf and intermediate nodes due to packet transmission and reception. This figure shows that GTCCF consumes less energy as compared to DCCC6. Fig. 15 shows the number of lost packets every second due to buffer overflow in each leaf node and intermediate node. It is clear that the number of lost packets in GTCCF is lower than DCCC6 algorithm in both leaf nodes and intermediate nodes. Fig. 16 shows the weighted fairness index for GTCCF and DCCC6. It is obvious that GTCCF has better fairness index that DCCC6 as it considers the priority of each leaf node in its operation. In general, table III summerizes the overall performance of GTCCF and DCCC6 in scenario 2.

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

VI. Conclusion

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

References

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

[4] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," *Internet Engineering Task Force (IETF), RFC 4919*, 2007.

[5] M. Weldon, *The Future X Network: A Bell Labs Perspective*. CRC Press, March 2016.

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

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

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

[10] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen, "Game Theory for Wireless Sensor Networks: A Survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[11] R. Machado and S. Tekinay, "A survey of game-theoretic approaches in wireless sensor networks," *Computer Networks*, vol. 52, no. 16, pp. 3047–3061, 2008.

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

[13] 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.

[14] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks:Ttheory, Models, and Applications*. Cambridge University Press, 2012.

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

[16] 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.

[17] 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.

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

[19] 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," *DOI 10.1109/TMC.2016.2585107, IEEE Transactions on Mobile Computing*, 2016.

[20] 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.

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

[22] L. Wang and G.-S. Kuo, "Mathematical Modeling for Network Selection in Heterogeneous Wireless Networks—A Tutorial," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 1, pp. 271–292, 2013.

[23] H. A. A. Al-Kashoash, Y. Al-Nidawi, and A. H. Kemp, "Congestion Analysis for Low Power and Lossy Networks," in *Wireless Telecommunications Symposium (WTS), 2016*. IEEE, 2016, pp. 1–6.

[24] H. Nikaido and K. Isoda, "Note on Noncooperative Convex Games," *Pacific Journal of Mathematics*, vol. 5, no. 5.

[25] J. B. Rosen, "Existence and Uniqueness of Equilibrium Points for Concave N-Person Games," *Econometrica: Journal of the Econometric Society*, pp. 520–534, 1965.

[26] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The trickle algorithm," *Internet Engineering Task Force, RFC6206*, 2011.

[27] R. G. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*. Courier Corporation, 2004.

[28] M. Stehlık, "Comparison of Simulators for Wireless Sensor Networks," Master's thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 2011.

[29] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes, "Powertrace: Network-level Power Profiling for Low-power Wireless Networks," Swedish Institute of Computer Science, Tech. Rep., 2011.

[30] 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.

[31] V. Michopoulos, "Congestion and Medium Access Control in 6LoWPAN WSN," Ph.D. dissertation, Computer Science, Loughborough University, 2012.

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

Hayder A. A. Al-Kashoash is a PhD student in the School of Electronic and Electrical Engineering, University of Leeds, UK. His research Interests include congestion control and resource management in WSNs, 6LoWPAN and LPWAN by utilizing game theory and optimization theory.

Maryam Hafeez is a Research Fellow with the School of Electronic and Electrical Engineering, University of Leeds, UK. Her research interests include design and analysis of protocols for next-generation green intelligent wireless networks employing tools from game theory and stochastic geometry.

Andrew H. Kemp is Senior Lecturer and Joint Programme Director with the Electronic and Electrical Engineering School, University of Leeds, UK. His research interests are in aspects of WSNs, in particular, related to localization, routing, and also multipath propagation studies to assist system development, cross-layer optimization, and the Internet of Things.