Computer Communications 000 (2016) 1–11

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

REFIACC: Reliable, efficient, fair and interference-aware congestion control protocol for wireless sensor networks

Mohamed Amine Kafi[a,b,*], Jalel Ben-Othman[c], Abdelraouf Ouadjaout[a,b], Miloud Bagaa[a,b], Nadjib Badache[a]

[a] DTISI, CERIST Research Center, Algiers, ALGERIA
[b] USTHB, University Houari Boumediane, Algiers, ALGERIA
[c] Laboratoire L2TI, Université de Paris 13, Paris, FRANCE

ARTICLE INFO

Article history:
Received 18 September 2015
Revised 8 May 2016
Accepted 28 May 2016
Available online xxx

Keywords:
Wireless sensor networks
Transport protocols
Congestion control
Fairness
Interference
Link capacity
Retransmission
Reliability

ABSTRACT

The recent wireless sensor network applications are resource greedy in terms of throughput and network reliability. However, the wireless shared medium leads to links interferences in addition to wireless losses due to the harsh environment. The effect of these two points translates on differences in links bandwidth capacities, lack of reliability and throughput degradation. In this study, we tackle the problem of throughput maximization by proposing an efficient congestion control-based schedule algorithm, dubbed REFIACC (Reliable, Efficient, Fair and Interference-Aware Congestion Control) protocol. REFIACC prevents the interferences and ensures a high fairness of bandwidth utilization among sensor nodes by scheduling the communications. The congestion and the interference in inter and intra paths hot spots are mitigated through tacking into account the dissimilarity between links' capacities at the scheduling process. Linear programming is used to reach optimum utilization efficiency of the maximum available bandwidth. REFIACC has been evaluated by simulation and compared with two pertinent works. The results show that the proposed solution outperforms the others in terms of throughput and reception ratio (more than 80%) and can scale for large networks.

1. Introduction

In a wireless sensor network (WSN), sensors monitor the environment and notify the base-station about any event. Applications may require the notifications to be continuous, periodic, or on event occurrence [1]. With the increasing number of WSN applications, many challenges appear and need to be resolved. For example, in some event-based (resp. continuous) applications, nodes may transmit significant quantity of data towards the base-station upon event occurrence (resp. permanently), e.g., video tracking, surveillance applications. In such applications, nodes take (periodically or at event happening) a snapshot of their surrounding area through camera and then communicate this data to the base-station as soon as possible. This is in order to have a global view of the monitored network and take appropriate decision. The resulted snapshot is divided into several hundreds of packets and travel in a multi-hop network towards the base-station. Bulk transfer of large amounts of sensed data is also a good motivation for maximizing throughput so that to allow nodes to transmit as quickly as possible in order to sleep and save energy.

The used network in such applications forms a tree (or a mesh topology) where the base-station constitutes its root and the farther nodes form the leaves which communicate to the base-station via intermediate nodes. According to nodes type, there is two communication engine mechanisms, namely, generating packets engine and forwarding engine. The leaves and intermediate nodes have to use the generating engine with the same rate, whereas the forwarding engine rate is used according to the sub-tree depth. As sensors share the same wireless channel, contention on the available bandwidth is inevitable. In a real environment, the packets collision caused by links interference or packets loss due to congestion on inter-paths and intra-path nodes dramatically affect the application throughput and cause high energy consumption.

Congestion control becomes necessary in such applications and consists of two parts: i) the congestion and interference detection, and ii) a rate control mechanism establishment, which adjusts the reporting rate. Different metrics are used in literature for detecting congestion, such as buffer length, packet inter arrival time, packet service time, channel load, etc [2]. The existing control approaches, in their mitigating or avoiding forms, underuse the throughput that

* Corresponding author.
E-mail address: kafiamine@gmail.com (M.A. Kafi).

http://dx.doi.org/10.1016/j.comcom.2016.05.018
0140-3664/

the network capacity can offer to the application. The mechanisms based on rate regulation between nodes use AIMD (Additive Increase Multiplicative Decrease) methods to balance the offered capacity [1], but ignore nodes interference which is the main cause of loss. On the other hand, scheduling based methods do not take into account dissimilarity between physical-links-capacities which may lead to buffer overflow and lack of fairness.

In this study, our aim is to maximize the network throughput by ensuring as much as possible the nodes rate fairness, the bandwidth utilisation efficiency and avoid any type of congestion. We mean by the fairness that any node in the network except the base-station has the same rate in the generating engine regardless being far away or near the base-station. This enables the base-station to get a homogeneous view on the whole network. Weighted fairness may also be used in the case of monitoring different data like video and temperature values. Whereas in the forwarding engine, any forwarding node must have enough rate capacity to forward its sub-tree packets. We also mean by the efficiency, the possibility to increase the generating engine of some nodes without affecting the other nodes generating engine rates. Indeed, we try to use the remaining capacities resulted from nodes capacities differences in order to maximize network throughput which is translated by the fast reception at the base-station of some snapshots to take decision as quick as possible. For this end, we propose a scheduling scheme that takes into account heterogeneity in link interference and capacity in order to provide efficient and fair rate partitioning over congested links. The proposed protocol identifies the hot spot points by measuring locally, at every node, the interfering neighbor links that cannot be simultaneously active in the whole schedule. Finally, the rate of each link and the number of given slots to each node is determined by taking into account the node depth and its link capacity. In [3], the basic features of REFIACC, named IACC (Interference Aware Congestion Control), were presented and compared with different fixed rate sending. In this study, an enhancement of the previous work is done by focusing deeply on efficiency and fairness problems parts. Linear programming optimisation is used to resolve the previously shortcomings. Furthermore, comparisons with pertinent protocols presented in Flush [4] and TARA [5] are performed.

The remainder of this paper is organized as follows. Section 2 presents the related work on congestion control and avoidance. We describe in Section 3 the overall congestion control scheme to which belongs our REFIACC mechanism as a schedule construction algorithm. In Section 4, the problem is formulated with a thorough network model. The proposed protocol *REFIACC* is presented in Section 5. After that, Section 6 presents the throughput optimization part of REFIACC, while Section 7 treats the fairness part of the optimisation problem. In Section 8, basic features of using our scheme in multichannel scenario is presented. Section 9 shows the simulation results. Finally, Section 10 summarises the main contributions of this paper and gives some perspectives.

2. Related work

The congestion control field for WSN has taken researchers' attention due to its importance for the application goal completion and avoiding energy depletion. In our previous work [1], we have reviewed congestion control protocols conceived in literature for WSNs. In the following section, recent and pertinent works are presented.

In CCF [6] (Congestion Control and Fairness), each node uses packet transmission duration to estimate the channel capacity. It then divides this capacity between its sub-tree nodes. A major drawback of CCF is that the remaining capacity from no-sending nodes goes unused. QCRA [7] (Quasi-static Centralized Rate Allocation for Sensor Networks) attempts to determine optimal and fair sources transmission rates at the base-station using information about topology, link loss rates, and communication pattern. Its heuristic results to a coarse-grained TDMA schedule between neighbours using CSMA. Loss rate is used to assign nodes' sending rates. QCRA decisions are periodic with epochs at the order of few tens of minutes. MCCP [8] (Multievent Congestion Control Protocol) uses successive data and schedule intervals. During data interval, nodes send data using a schedule received from their next hop nodes. As one packet can be sent at a slot, the schedule length represents the reporting rate. With short length slots, the rate is increased. During the schedule intervals, nodes generate the schedule for the next data interval. Packet delivery time and buffer size are used to attribute time slots. But slot attribution in this scheme does not specify any contention avoidance. TARA (Topology-Aware Resource Adaptation) [5] uses resource control to ensure application fidelity. It defines the link congestion sum as the sum of link's traffic and interfering links' traffic. TARA uses graph-coloring approach for capacity estimation. It defines the topology interference degree and constructs the spatial interference graph.

In Flush [4], large data is divided into packets and sent in a pipelined transmission. The base-station schedules transfers to avoid inter-flows interferences. It uses end-to-end ACKs to ensure reliability, and hop-by-hop rate control. Flush dynamically chooses the sending rate using bandwidth measurements and interference information to avoid intra-path interference, which depends on the interference range at each node. Flush is very restrictive as only one source at a time can transmit. It does not describe the scheduling heuristic but just the capacity sharing part. CADT [9] (Capacity Aware Data Transport) protocol is similar to Flush but considers many flows. CADT applies AIMD rate control using immediate downstream nodes buffer and the concerned node link state.

CODA [10] (Congestion Detection and Avoidance) is a mitigation rate control protocol, where each node detects congestion using both channel and buffer loads. The node controls its rate in an AIMD manner. CODA considers two strategies: i) open-loop back pressure for transient congestion, where the concerned node broadcasts the message to its neighbors that further propagate these messages to upstream nodes, depending on their buffer occupancy, and ii) an end-to-end acknowledgement-based approach (also named closed-loop for persistent congestion). However, CODA does not focus on per-source fairness.

In [11], authors propose a TDMA schedule to ensure maximum throughput and fair rate allocation by taking into account the requirement imposed on network lifetime. The authors use lexicographic Max-Min to formulate the rate allocation along with fairness, maximum throughput, and slots reuse in the purpose to reach a minimum frame length.

In TSCH (Time Synchronized Channel Hopping) [12–20], which is a part of the IEEE 802.15.4e standard defined by the IETF initiative, nodes follow a schedule in order to transmit, receive, or sleep in a slotted manner. In [12,14], authors explain the terminology used in the definition of the schedule in addition to some standardisation efforts in the domain of LLN (low power and lossy networks). Whereas in [13] the required architecture of the network for industrial IoT (internet of things) appliance is proposed. The TSCH schedule indicates which channel to use, as IEEE 802.15.4e allows the use of 16 channels, and which neighbor to communicate with in the concerned active slot to allow many couples of nodes to communicate at the same slot using different channel offsets in the purpose to increase network capacity. The standard shows how to execute a schedule as well, but it does not specify how to built it. In [15], a Traffic Aware Scheduling Algorithm that builds a centralized schedule, based on the network topology and the traffic load generated by each node is proposed, while in [17] a distributed version of scheduling algorithms is shown. The idea of

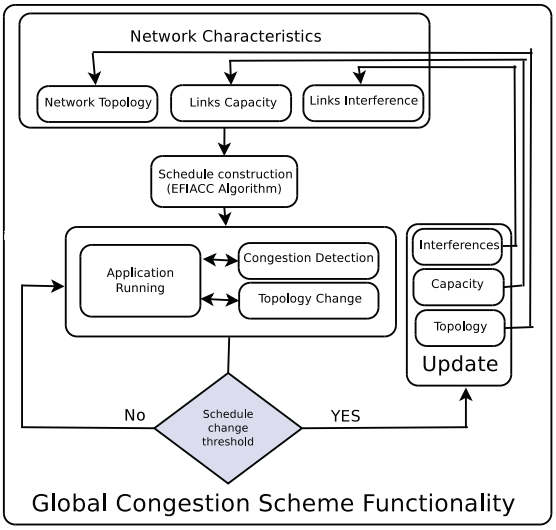


Fig. 1. Global Congestion Control during Application lifetime.

[18] is to construct a schedule on the fly according to the change in nodes demand. In [20] an energy consumption model of the TSCH schedule mechanism is presented.

The protocols aforedescribed do not address the problem of link capacities dissimilarity that may exist between nodes and lead to a lack of throughput and fairness. In the next section, we describe the global scheme for controlling congestion and highlight the scheduling construction part handled by our REFIACC mechanism.

3. Global congestion control scheme

As we have invoked in our previous introduction, the existing works for congestion handling present shortcomings either by the no consideration of interferences with rate sharing based schemes or by the no consideration of link capacity dissimilarities with schedule based schemes. The aim of our work is to consider these two parameters for the construction of the whole network schedule in order to achieve as maximum as possible of throughput while offering fairness as possible, too.

In a real application, network characteristics may change during application lifetime. In the case when these changes diverge from the initial ones, any schedule constructed previously may become inefficient. Therefore, a mechanism that allows detecting these changes during the application lifetime is mandatory for the good running of the application. In [7], authors change the network schedule periodically but this may be inefficient (too late) if the changes happen before reaching this periodicity, or may lead to free overhead if network characteristics do not change at the periodicity happening. For these reasons, we propose a threshold based schedule reconstruction. The threshold value calculation must take into account the congestion happening frequency in order to differentiate transient from permanent ones, as explained in our previous work [21]. Also, network topology change has to be taken in consideration. If the threshold value is not yet reached, the schedule is still usable, and the congestion may be considered as transient. In this case, momentary stop sending of some nodes may help the emptying of some buffers or the transient interference to disappear. If on the other hand this threshold value is reached, this is an indication to recalculate the fundamental schedule parameters, namely links capacities and interferences and probably topology change. The scheme presented in Fig. 1 highlights the global congestion control scheme functionality.

For the remaining of the paper, we will mainly focus on the schedule construction part and leave the threshold calculation and network reconstruction for future studies. The communication tree is supposed to be constructed using some routing protocol, such as [22], which serves as input to the proposed REFIACC mechanism. In the next section we introduce REFIACC basic idea through the capacity and interference problem formulation.

4. Network model and problem formulation

In order to focus on our problem, we assume that all nodes have synchronized clocks and each node has a single half-duplex radio transceiver. We suppose that our application needs to a simple fairness rather than a weighted fairness. Communication between nodes is modelled by an undirected graph $G = (V, E)$, where V is the set of nodes, with a base-station $B \in V$, and E is the set of edges that represent the links communication between nodes. Based on the observation that the links between nodes do not have the same capacity, every edge $(u, v) \in E$ is supposed having a specified capacity $C_{u,v}$ to transmit packets from node u to node v.

The data are gathered over a tree structure $G_T = (V, E_T)$ rooted at B, such that $E_T \subseteq E$ represents the tree edges. The problem that we tackle in this paper is how to gather the data, without collisions and congestions, from a set of source nodes called V_S (i.e., $V_S \subseteq V$) through the tree structure, while maximizing the network throughput and ensuring fairness as possible. All source nodes are supposed to continually generate data. Time is divided into contiguous frames, where each frame is constituted of a set of slots $\{\sigma_1, \sigma_2, \ldots \sigma_l\}$ having the same duration δ. During each slot σ_i, the data packets are generated or transferred from a set of children nodes to their parents. $\mathbb{S}(u)$ denotes the set of slots used by a node $u \in V$ to generate data, receive the data from its children, or transmit to its parent *w.* $\tau_\sigma(u)$, $\lambda_\sigma(u)$, $\mu_\sigma(u)$ denote the rates of data generation, reception, and transmission during a slot $\sigma \in \mathbb{S}(u)$, respectively. The *Maximum Throughput Congestion and Collision Prevention* (TCCP) problem is how to find the appropriate slot distribution $\mathbb{S}$ along with the appropriate functions $\tau_\sigma(u)$, $\lambda_\sigma(u)$, $\mu_\sigma(u)$ for every node $u \in V$ and every slot $\sigma \in \mathbb{S}(u)$, such that the throughput is maximized, and collisions and congestions are prevented. An instance of the *TCCP* problem must satisfy the following conditions:

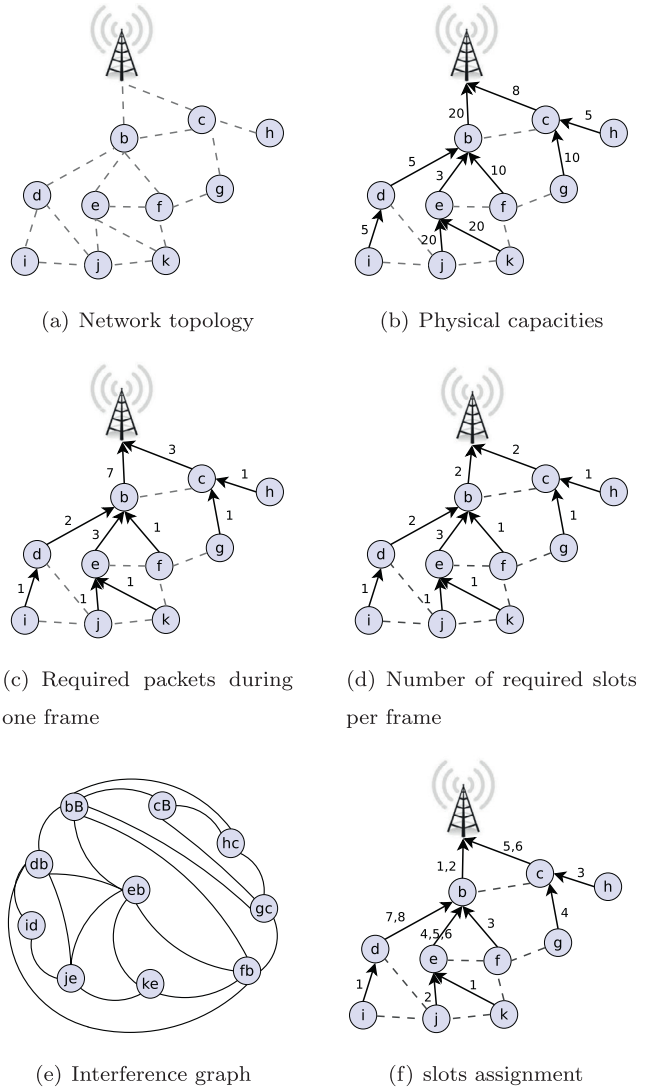
1. The data are generated only by the source nodes, i.e., $\forall u \notin V_S, \forall \sigma \in \mathbb{S}(u) : \tau_\sigma(u) = 0$.
2. The rate of data transmission from a node u to its parent w (*i.e.*, $(u, w) \in E_T$) should not exceed the capacity $C_{u,w}$. Formally, $\forall \sigma \in \mathbb{S}(u) : \mu_\sigma(u) \leq C_{u,w}$.
3. Each node has a half-duplex radio transceiver and thus it cannot transmit and receive simultaneously: $\forall u \in V, \forall \sigma \in \mathbb{S}(u) : \lambda_\sigma(u)\mu_\sigma(u) = 0$.
4. *Collision-free constraint*: Two nodes u and v cannot transmit at the same time slot σ in one of the following cases:
   - Node u is parent of v or vice versa;
   - Nodes u and v have different parents, say w and z, respectively. A collision occurs *iff* w or z is within the transmission range of v or u, respectively.

   In these two cases, to prevent the collision between nodes, u and v, the following *collision-free constraint* should be verified:

   $$\forall \sigma \in \mathbb{S}(u) \cap \mathbb{S}(v) : \mu_\sigma(u)\mu_\sigma(v) = 0$$

5. *Congestion-free constraint*: To prevent the congestion, we should verify that the amount of the transmitted data is greater than the amount of the received and generated data. Namely, the *congestion-free constraint* can be expressed as follows:

   $$\forall u \in V : \sum_{\sigma \in \mathbb{S}(u)} \mu_\sigma(u) \geq \sum_{\sigma \in \mathbb{S}(u)} (\lambda_\sigma(u) + \tau_\sigma(u))$$



(a) Network topology (b) Physical capacities

(c) Required packets during one frame (d) Number of required slots per frame

(e) Interference graph (f) slots assignment

Fig. 2. Illustrative example to show how *REFIACC* is applied.

In the next section, we will describe REFIACC protocol while trying to fulfil the aforementioned five primordial conditions so that we get the desired results.

5. Protocol description

In order to facilitate the presentation of *REFIACC*, Fig. 2 will be referenced throughout the discussion, as an illustrative example. In the figure, a solid arrow between two nodes, *a* and *b*, indicates that *b* is the parent of *a* in the communication tree. The dotted lines represent the graph connectivity, i.e., the presence of a communication link between a pair of nodes. The number besides the solid arrow (*a, b*) in Fig. 2(*b*) represents the physical link capacity, $C_{a,\,b}$, on the communication tree. The number besides the solid arrow (*a, b*) in Fig. 2(*c*) represents the number of packets required to be forwarded by node, *a*, to ensure the fairness. This number is simply the sub-tree size. In the case of using weighted fairness, this number may be different from the sub-tree size according to the needed number of packets along the sub-tree nodes. Whereas, the number besides the solid arrow (*a, b*) in Fig. 2(*d*) represents the number of slots required for node, *a*, to ensure the fairness. The number besides the solid arrow (*a, b*) in Fig. 2(*f*) represents the assigned slots for each node in the network. All the network nodes in Fig. 2 are considered as sources.

REFIACC consists of two steps aiming to establish an appropriate schedule that takes into account the real network capabilities. Firstly, the link capacity between each node and its parent is estimated. The estimated capacities are forwarded from the network nodes to the base-station. Secondly, the scheduling is established to ensure the fairness while preventing congestions and collisions in the network. The estimated capacities help *REFIACC* to establish a schedule that allocates for every node the appropriate time to transmit with the appropriate rate. The schedule construction is done at the base-station which is supposed having enough processing capacities in order to achieve complex calculation, like linear program solving software (Sections 6 and 7).

5.1. Link quality estimation

The congestion control in *REFIACC* is based on real measurements of radio links capacity. To obtain an estimation of these capacities, every node, *u*, assesses radio link relating it to its parent, *w*, by sending a burst of packets during a determined amount of time, *T*. Every packet is sent after the reception of the previous packet's acknowledgement, or after a time-out from the former submission. The link capacity, $C_{u,\,w}$, is then approximated by the total number of acknowledged packets divided by the period *T*. Upon completion of this phase, node *u* sends to the base-station the estimated link capacity, as well as the ID of its parent. This message is forwarded via the tree structure. This phase and the previous one are depicted in Fig. 2(*b*). In what follows, we use interchangeably the terms link capacity and node capacity (link between the node and its parent) for the same purpose.

5.2. Schedule construction

This step is executed by the base-station after collecting link capacity messages on the communication tree. The process of rate scheduling is then divided into three sub-phases: *Slot Size Calculation, Rate Distribution* and *Slots Assignments*.

5.2.1. Slot size calculation

After the collection of nodes' capacities at the base-station, the next step is to define the slot characteristics that are its *duration(period)* and *capacity(rate)*. The slot duration must be uniform for all nodes in order to allow their synchronisation and avoid collisions for sending and reception during the frame. To calculate the *slot duration*, the base-station chooses the slowest node capacity, let's be C_s. For this node, to send successfully one packet, it needs $1/C_s$ seconds. This value will be our *slot duration*, named δ. For the second slot characteristic (the *capacity*), each node *u* has its own *slot capacity,* C_u, towards its parent *w* which is simply $C_{u,\,w}/C_s$. For the slowest node in the network, the slot capacity will be one packet. If this difference in links capacities is due to different technologies, the slowest node may make only one transmission with the positive Ack reception. On the other hand, if this link capacity difference is due to link quality losses, our slot may be considered as a virtual mega slot composed of a fixed number of micro slots in order to allow the successful transmission.

5.2.2. Rate distribution

For each node *u*, the aim is to distribute its available bandwidth such that:

- all source nodes under the responsibility of node *u* are enabled to fairly send their data. In the case of weighted fairness, each node is attributed a rate regarding its needed packets.
- congestion is avoided by computing an appropriate sending rate for *u*. This rate does not exceed the capacity of the router nodes present in the path towards the base-station.

To guarantee the fairness condition, the base-station computes the number of source nodes in the underneath sub-tree of u, which is denoted by ST_u. This is shown in Fig. 2 (c). Therefore, node u should be allowed to send at least ST_u packets during a single frame. Since the frame is divided into slots of duration δ, node u will be assigned at least $s_u = \left\lceil \frac{ST_u}{\delta C_{u,w}} \right\rceil = \left\lceil \frac{ST_u}{C_u} \right\rceil$ slots, where w is the parent of u (as depicted in Fig. 2(d)).

5.2.3. Slots assignments

After computing the appropriate needs for each node in terms of slots per frame, *REFIACC* starts the slot assignment process. This process aims at orchestrating the starting time of each slot so that to avoid collisions and let the non interfering links to transmit simultaneously. An interference graph $G_I = (V_I, E_I)$ is first built. For the previous example, the corresponding interference graph is shown in Fig. 2(e). The set of vertices $V_I = E_T$ corresponds to the set of radio links between nodes and their parents in the tree structure. Two links, say (i, j) and (k, l) are connected in E_I if they are in interference with each other. We suppose in our example that the interfering links are picked only from the communication links, even in reality the interference may contain other links.

The algorithm of slot assignment is based on an iterative process. In every iteration n, *REFIACC* searches in, G_I, for the maximal independent set, say I_n, which contains the vertices with the higher number of edges. This is in the purpose to cut the interference graph into many sub-graphs that allow fast reuse of slots. Since the radio links in I_n are not interfering, the set of nodes $\{u|(u, v) \in I_n\}$ can transmit in slot n. When a slot is assigned to a node u, its required number of slots per frame is decreased. When this number becomes zero, the vertex of the link (u, v) is removed with its corresponding edges in, G_I. The iterations stop when the graph G_I becomes empty. The number of iterations, n, constitutes the frame size which is the minimum possible size.

The presented algorithm ensures fairness while it avoids congestion and collisions. The resulted slots assignment is shown through the example of Fig. 2(f). However, in some situations, it does not achieve an optimal efficiency and links can be underused. Indeed, the nodes that have consumed their required number of slots (removed from G_I) can take extra slots. This is provided if they do not create congestion and interference with links in I_n for the next steps. In order to ensure more throughput and at the same time as fairness as possible, we have formulated two linear programming problems where the aim of the first problem is to reach a maximum of throughput with the previous calculated frame size and with interference constraints. While the goal of the second linear programming problem is to maximize the fairness between the senders as much as possible by reducing the dissimilarity in their throughput. In the following Sections 6 and 7 the problems are formulated.

6. Throughput maximization problem

As explained in the previous section, the goal of this optimization part is to ensure that at least each node in the communication tree can transmit its sub-tree packets, but if the bandwidth allows to transmit more packets without leading to eventual congestion, this will bring more efficiency to the network capacity use. In the Table 1 below, the network notations used in the problem formulation are presented. In the set Γ, we find the groups of nodes that interfere mutually and therefore cannot send simultaneously. This is obtained by extracting all the cliques in the interference graph.

We denote by $\mathcal{T}_i^s$ a binary variable where:

$$\mathcal{T}_i^s = \begin{cases} 1 & \text{if node } i \text{ transmits in slot } s \\ 0 & \text{else} \end{cases}$$

Table 1
Throughput problem notations.

Notation	Description
N	Set of nodes, except the base-station
S	Set of slots in the frame
Γ	Set of interference groups.
I	Interference groups matrix, such that: $\forall g \in \Gamma, \forall i \neq j \in N : (I_i^g = I_j^g = 1) \Rightarrow i$ and j are interfering.
C_i	Maximum slot capacity of node i.
CH_i	Set of children of node i.
ST_i	Number of nodes in the subtree of node i.

Also, we denote by $\mathcal{P}_i^s$ the number of packets sent by node i in slot s. The *Throughput Maximization Problem* can be expressed as follows:

$$\begin{cases} \textbf{max} \sum_{i \in N, s \in S} \mathcal{P}_i^s \\ \\ \textbf{s. t.} \\ \forall s \in S, \forall g \in \Gamma : \sum_{i \in N} I_i^g \times \mathcal{T}_i^s \leq 1 \\ \text{(Nodes in the same interference group cannot send in the same slot)} \\ \\ \forall i \in N : \sum_{s \in S} \mathcal{P}_i^s \geq ST_i \\ \text{(A node should forward at least one packet from each node in its subtree)} \\ \\ \forall i \in N : \sum_{s \in S} \mathcal{P}_i^s \geq \sum_{j \in CH_i} \sum_{s \in S} \mathcal{P}_j^s \\ \text{(To avoid congestion, the output of every node should be greater than its input)} \\ \\ \forall i \in N, \forall s \in S : (\mathcal{T}_i^s = 0 \wedge \mathcal{P}_i^s = 0) \vee (\mathcal{T}_i^s = 1 \wedge \mathcal{P}_i^s \geq 1) \\ \text{(Connect the two variables } \mathcal{T} \text{and } \mathcal{P}, \text{ so that } \mathcal{T}_i^s{=}0 \Leftrightarrow \mathcal{P}_i^s{=}0) \\ \\ \textbf{with this bound} \\ \forall i \in N, \forall s \in S : \mathcal{P}_i^s \leq C_i \\ \text{(The number of packets sent in a slot should be less than the maximal capacity} \end{cases} \quad (1)$$

As the last constraint is not formulated adequately for the linear problem, we introduce another variable $\mathcal{A}_i^s$ in order to transform it. We denote by $\mathcal{A}_i^s$ a binary variable where:

$$\mathcal{A}_i^s = \begin{cases} 1 & \text{if } \mathcal{T}_i^s = 1 \wedge \mathcal{P}_i^s \geq 1 \\ 0 & \text{else } (\mathcal{T}_i^s = 0 \wedge \mathcal{P}_i^s = 0) \end{cases}$$

So, the three following constrains will replace the last previous one:

$$\begin{cases} \forall i \in N, \forall s \in S : \mathcal{T}_i^s + \mathcal{P}_i^s \leq 170 \times \mathcal{A}_i^s \\ \\ \forall i \in N, \forall s \in S : \mathcal{T}_i^s \geq \mathcal{A}_i^s \\ \\ \forall i \in N, \forall s \in S : \mathcal{P}_i^s \geq \mathcal{A}_i^s \end{cases} \quad (2)$$

Here, we use 170 to express a large number superior to the highest value of $\mathcal{P}_i^s$, (as $\mathcal{T}_i^s$ max value is one) in order to force the left hand constraint part when the concerned variable $\mathcal{A}_i^s$ is one. So, any other number higher than 170 in the right part is correct. For our case, we have done a simulation test between two nodes having a perfect transmission link. The reception rate was 155 packets/second.

The result of this step of the algorithm is the slots affectation to each node and the number of packets to send in each slot. But

as the next step's goal is to enhance fairness, the slots' capacities are not definitive.

7. Fairness through nodes throughput difference minimization problem

Noting that from the previous step we get $\mathcal{T}$ and $\mathcal{P}$ values to be used in this step of the problem. The chosen slots of each node $\mathcal{T}_i$ keep unchanged, whereas the number of packets sent in each slot $\mathcal{P}_i$ may be changed except that of the base-station's children which describe the overall throughput. In fact, the aim of maximizing throughput, which is the sum of base-station's children $\mathcal{P}_i$, is reached from the previous step. The goal of this step is to reduce the dissimilarity of generated packets between nodes in the subtrees of base-station's children.

In this problem, we denote by $\mathcal{G}_i$ the number of packets generated by node i during the frame.

Also we denote by $\mathcal{F}_i$ the number of packets forwarded by node i throughout the frame as well.

In order to ensure fairness, we minimize the difference between the nodes' generated packets. Let's denote by $\mathcal{D}_{ij} = \mid \mathcal{G}_i - \mathcal{G}_j \mid$. As the absolute value cannot be used in the PL problem, we change $\mathcal{D}_{ij}$ to be $\mathcal{D}_{ij} \geq \mathcal{G}_i - \mathcal{G}_j \wedge \mathcal{D}_{ij} \geq \mathcal{G}_j - \mathcal{G}_i$. The problem can be written as follows:

$$\begin{cases} \textbf{min} \sum\limits_{i,j \in N, i \neq j} \mathcal{D}_{ij} \\ \textbf{s. t.} \\ \forall i \in N : \mathcal{F}_i = \mathcal{G}_i + \sum\limits_{j \in \mathcal{CH}_i} \mathcal{F}_j \\ \forall i, j \in N, i \neq j : \mathcal{D}_{ij} \geq \mathcal{G}_i - \mathcal{G}_j \wedge \mathcal{D}_{ij} \geq \mathcal{G}_j - \mathcal{G}_i \\ \textbf{with these bounds} \\ \forall i \in N : \mathcal{F}_i \leq \mathcal{C}_i \times \sum\limits_{s \in \mathcal{S}} \mathcal{T}_i^s \\ \forall i \in \mathcal{CH}_{base-station} : \mathcal{F}_i = \sum\limits_{s \in \mathcal{S}} \mathcal{P}_i^s \end{cases} \quad (3)$$

So, the final result of the algorithm from this step is the number of generated packets for each node and the number of forwarded packets from each node at a single frame. The generated packets of each node are available at its queue by the beginning of the frame, while the forwarding rate at each slot is simply the division of the forwarding capacity of the node during a frame between its available sending slots.

In the Algorithm 1 a summary of the precedent steps of our scheme is presented.

Algorithm 1 REFIACC General Algorithm

Require:

N : Set of sensor nodes in the network.
E and E_T: Network Communication links and communication Tree.
1: C = CalculateCapacities(E_T); # $C[i, j]$: the capacity of the link between i and j.
2: ($CUniform, SlotPeriod$) = SlotCalculation(C); # $CUniform[i, j]$: slotCapacity of the link between i and j.
3: $SubTrees$ = CalculateSubTree(E_T); # $SubTrees[i]$: minimum number of packets sent by i during a frame.
4: $Slots$ = CalculateSlotsNeed($SubTrees$, $CUniform$); # $Slots[i]$: minimum number of slots required by i to transmit its $SubTrees[i]$ packets during a frame.
5: (G_I, Li, Γ) = CalculateInterference(E, E_T); # G_I: graph of interference links in E_T. $Li[i, j]$: the set of links that interfere on the link (i, j). Γ: the set of the sets of links that interfere mutually.
6: $FrameSize$= CalculateFrameSize($G_I, Slots$).
7: (T, P) = ThroughputMaximizationLP($E_T, FrameSize, \Gamma$, $CUniform, SubTrees$);
$T[i]$: The set of slots during which the node i transmits. $P[i, s]$: The number of packets sent by node i during the slot s.
8: (F, G) = DifferenceFairnessMinimLP($E_T, CUniform, T, P$); # $G[i]$: Number of packets generated by node i during a frame. $F[i]$: Number of packets forwarded by node i during a frame.
9: $Final_P$=SlotsPacketDistribution($T, F, G, CUniform$); # $Final_P[i, s]$: Number of packets to send by node i during slot s.
10: return $(G, T, Final_P)$;

8. Multi channel extension of REFIACC

The recent advances in radio technologies have allowed more scalability to WSN throughput by the use of multichannel paradigm. The philosophy of this method is to allow the use of many channels for transmission and reception. The previous conception of our algorithm could be used with some minor changes in order to adapt to multichannel behaviors that lead to more slots reuse and therefore more throughput. The three starting steps of our schedule keep unchanged with the multichannel use. Namely, the link quality estimation, the slot size calculation and rate distribution. The step of slot assignment is lightly changed, this is in order to allow the full use of different channels. Let L denotes the number of channels that can be used by a transceiver while the collisions inter-channels are prevented. The interfering graph exploitation is also different because the interfering nodes may be scheduled in the same slot but in different channels. The algorithm is applied in an iterative manner. In every iteration n, *REFIACC* searches in G_i for the maximal independent set, say I_n, which contains the vertices with the higher number of edges. This is in the purpose to cut the interference graph into many sub-graphs that allow simple slots reuse. Since the radio links in I_n are not interfering, the set of nodes $\{u|(u, v) \in I_n\}$ can transmit at the slot n with the channel l. When a slot is assigned to a node u with the channel l, its required number of slots per frame is decreased. When this number becomes zero, the vertex of link (u, v) is removed with its corresponding edges in G_i. This operation is repeated with the other available channels and the slot iteration is not incremented until the exploitation of all the channels is performed. For each channel l, the links used in the previous channels are not reused because they are already active in this slot, as the slot reuse in different channels happens simultaneously. The iterations stop when the graph G_i becomes empty. The number of iterations, n, constitutes the frame size which is the minimum possible size.

For ensuring more efficiency while avoiding congestion at the same time to offer suitable fairness, the optimisation problem parts are also used in this multichannel extension. Namely, the first optimisation part consists to allow as much as possible throughput maximisation through the slots reuse. While the second optimisation part consists to reduce the dissimilarities in individual throughput to get more fairness as possible.

8.1. Throughput maximization problem

The main difference of this part compared to the single channel is the use of multiple channels [1, L] in the same slot.

We denote by $\mathcal{T}_i^{sl}$ a binary variable where:

$$\mathcal{T}_i^{sl} = \begin{cases} 1 & \text{if node } i \text{ transmits at slot } s \text{ with the channel } l \\ 0 & \text{else} \end{cases}$$

Also, we denote by $\mathcal{P}_i^{sl}$ the number of packets sent by node i in slot s on the channel l. The *Throughput Maximization Problem* can

be expressed as follows:

$$\begin{cases} \textbf{max} \sum_{i\in N, s\in S, l\in L} \mathcal{P}_i^{sl} \\ \textbf{s. t.} \\ \forall i \in N, \forall s \in S : \sum_{l\in L} \mathcal{T}_i^{sl} \leq 1 \\ \text{(a node cannot send in different channels of the same slot)} \\ \forall s \in S, \forall l \in L, \forall g \in \Gamma : \sum_{i\in N} I_i^g \times \mathcal{T}_i^{sl} \leq 1 \\ \text{(Nodes in the same interference group cannot send at the same slot with the same channel)} \\ \forall i \in N : \sum_{s\in S, l\in L} \mathcal{P}_i^{sl} \geq ST_i \\ \text{(A node should forward at least one packet from each node in its subtree)} \\ \forall i \in N : \sum_{s\in S, l\in L} \mathcal{P}_i^{sl} \geq \sum_{j\in CH_i} \sum_{s\in S, l\in L} \mathcal{P}_j^{sl} \\ \text{(To avoid congestion, the output of every node should be greater than its input)} \\ \forall i \in N, \forall s \in S, \forall l \in L : (\mathcal{T}_i^{sl} = 0 \wedge \mathcal{P}_i^{sl} = 0) \vee (\mathcal{T}_i^{sl} = 1 \wedge \mathcal{P}_i^{sl} \geq 1) \\ \text{(Connect the two variables } \mathcal{T} \text{ and } \mathcal{P}, \text{ so that } \mathcal{T}_i^{sl} = 0 \Leftrightarrow \mathcal{P}_i^{sl} = 0) \\ \text{with this bound} \\ \forall i \in N, \forall s \in S, \forall l \in L : \mathcal{P}_i^{sl} \leq \mathcal{C}_i \\ \text{(The number of packets sent during a slot should be less than the maximal capacity)} \end{cases} \tag{4}$$

As the last constraint is not formulated adequately for the linear problem, we introduce another variable $\mathcal{A}_i^{sl}$ in order to transform it. We denote by $\mathcal{A}_i^{sl}$ a binary variable where:

$$\mathcal{A}_i^{sl} = \begin{cases} 1 & \text{if } \mathcal{T}_i^{sl} = 1 \wedge \mathcal{P}_i^{sl} \geq 1 \\ 0 & \text{else } (\mathcal{T}_i^{sl} = 0 \wedge \mathcal{P}_i^{sl} = 0) \end{cases}$$

8.2. Fairness through Nodes throughput difference minimization problem

For this optimisation part, it is the same reasoning as the single channel deployment apart the utilisation of channel indexing. So, the equations became as follows:

$$\begin{cases} \textbf{min} \sum_{i,j\in N, i\neq j} \mathcal{D}_{ij} \\ \textbf{s. t.} \\ \forall i \in N : \mathcal{F}_i = \mathcal{G}_i + \sum_{j\in \mathcal{CH}_i} \mathcal{F}_j \\ \forall i, j \in N, i \neq j : \mathcal{D}_{ij} \geq \mathcal{G}_i - \mathcal{G}_j \wedge \mathcal{D}_{ij} \geq \mathcal{G}_j - \mathcal{G}_i \\ \textbf{with these bounds} \\ \forall i \in N : \mathcal{F}_i \leq \mathcal{C}_i \times \sum_{s\in S} \mathcal{T}_i^{sl} \\ \forall i \in \mathcal{CH}_{base-station} : \mathcal{F}_i = \sum_{s\in S} \mathcal{P}_i^{sl} \end{cases} \tag{5}$$

In the next section, we perform extensive simulation of our schedule in a single channel scenario. Also, REFIACC is compared with two pertinent works using single channel. For future works, we plan to compare our conception with other works using multichannel behaviour.

9. Protocol evaluation

In this section, *REFIACC* is evaluated by simulation using TOSSIM [23]. The number of nodes, N, is varied up to 100. Before running the network simulator, a random connected graph with an expected node degree $d \in \{3, 5, 10, 15\}$ has been generated for every number of nodes. Similarly to [24], an edge between two nodes, u and v, is stochastically selected with a probability: $P = \frac{d}{N}$. This is called edge probability and results on node's neighbors construction. Further, a different capacity is assigned to each edge in the communication graph. This value varied between the ideal link (reception link probability=1), to the worst one (reception link probability=0.05). We used CPLX of IBM [25] to resolve our linear programming part to optimize the slot maximum utilisation.

We consider an application where each node generates 100 packets periodically. This is usual in an application where each node takes a picture that contains 100 ko from the environment. This data is forwarded after being fragmented into packets where the size of each packet does not exceed 1 ko. This is a typical scenario for monitoring applications that use periodic traffic sampling (snapshot of the monitored zone in a video surveillance application). All nodes in the network act as sources for traffic generation, i.e. $V_S = V \setminus \{B\}$. The simulation stops when every node transmits 100 packets. Each generated packet is buffered in a waiting queue, from which the transport protocol picks up a packet from the head. We have compared the performance of *REFIACC* with two candidate protocols, namely Flush [4] and TARA [5]. Each experiment is repeated 33 times and the results are presented with 95% confidence interval.

We have used in our simulations 4 different densities namely: 3, 5, 10 and 15. These densities have a direct effect on the interferences and therefore on the slots re-utilization. To quantify the performance of the candidate protocols, we have chosen 4 parameters that reflect congestion happening or control curb sending, namely *Throughput, Reception Ratio, Number of Packets Received and Average Retransmission*. For space restrictions, we will show only the simulation results for densities 3 and 15.

Candidate protocols characteristics:

1. Each protocol is a schedule based one.
2. Each one takes interferences into account:
   - we calculate interferers of each node before the schedule calculation.
   - We have enhanced TARA with this information as it took a simplistic uniform interference in its article description.
   - Flush supposed that interferences change during execution. We have supposed them fixed, gave this information to Flush and freeze its hearing interference mechanism.
3. We have focused only on the schedule based part of TARA, and not continue with the second part of changing paths, as routing change is out of the interest of our study.
4. We have constructed the schedule for Flush as it does not do it after the rate calculation, even describing in its article that nodes send in their slots.

Sending packets policy:

1. For each protocol, each node tries to send 100 packets. The slot period allows to send one packet and to retransmit in case of loss. REFIACC allows the transmission of many packets in the same slot according to its capacity. On the other hand Flush and TARA send one packet. Before the slot period ends, and in the case of negative acknowledgement of the receiver, the sender retransmits the packet(s) until the packet is well received or the slot period is elapsed.
2. For TARA and Flush, we have chosen three different variants depending on the slot period resulted from the link quality.

Table 2
Simulation parameters.

Parameter	Value
Simulator	Tossim
Number of base-station Nodes	1
Number of Sensor Nodes	1 to 100 Nodes.
Network Density (Average Neighbors)	3,5,10,15
MAC Protocol	CSMA/CA
Channel Bandwidth	146 Pckts/Second
Channel Loss Probability	[0.05, 1]
Data Header Payload Size	9 bytes
Number of Flows	All nodes apart the base-station
Number of Packets to send	100

- The first one chooses the smallest period (named TARA-min, respectively Flush-min), and corresponds to the better link with no loss. This period is sufficient to send exactly one packet and receive its Ack.
- The second one chooses the largest period (named TARA-max, respectively Flush-max), and corresponds to the link that has a high loss probability (in our case we chose a link with a 0.05 receive probability). This period allows to send many times the same packet in order to have a positive Ack.
- The last one chooses the average period (named TARA-aver, respectively Flush-aver), and corresponds to the link that has an average loss probability.

In Table 2, our simulation parameters are depicted. In the following sections, simulation results thorough the performance parameters are presented.

9.1. Throughput

The average throughput at the base-station is the number of packets received during a time unit (Packets/Second in our case). For evaluating this parameter, we have calculated two values. The first one analytically, just after the scheduling built. The second one is resulted from simulation. This is in order to show the divergence between the planned one (analytical) and the simulation. REFIACC is the realistic one as there is a very light divergence between its analytical and simulation results.

9.1.1. In the analytical throughput graphs

(Figs. 3(*a*) and 4(*a*)): we can observe that TARA-min has the highest calculated throughput value. This is because it has scheduled adequately the interferer nodes, as done in REFIACC, in addition to its minimal slot period that results on a short frame size which explains the high analytical throughput. But this value is wrong calculated as all links are supposed having ideal capacities. The second high throughput value is for Flush-min, it may have a better analytical throughput than REFIACC analytical one. Flush drawback is that of using sequential sending thorough the sources. But FLUSH-min supposition that links have ideal capacities, compared to REFIACC, makes its analytical throughput better than REFIACC for dense topologies (10 and 15). This can be explained because with dense topologies the slots reuse in REFIACC become less feasible and the nodes will send more or less sequentially added to the long period of REFIACC slot. This justifies its lower analytical throughput than Flush-min.

For TARA-max and Flush-max, they have the lowest throughput but TARA-max is better, because of their long slot period and TARA-max senders use parallel sending if possible, contrary to Flush-max.

For TARA-aver and Flush-aver, the same reasoning can be applied. TARA-aver may sometimes have better throughput than REFIACC, because this depends on the resulted topology with its links quality. REFIACC, takes into account the link quality to have the appropriate slot period, which may be for some topologies longer than the average value (bad links topologies). For Flush-aver, it can reach the REFIACC analytical throughput just for the small dense topologies. This can be explained that for dense topologies slots reuse is hard, and as the topology is small the reuse is more difficult, which justifies that REFIACC may have the same throughput than Flush-aver, but as the topology grows, the slots reuse is possible and REFIACC outperforms Flush-aver. We remember that these results are just envisaged ones and that TARA-min and Flush-min have better results because they do not take real link parameters into account. But when going to simulation, their envisaged results will be wrong and the throughput degrades a lot contrary to REFIACC results which keep the same (a light divergence).

9.1.2. In the simulation throughput graphs

(Figs. 3(*b*) and 4(*b*)): It is clear that REFIACC will react better than the other protocols. This is justified by the fact that its parameters (slots period, slot capacity) are well calculated contrary to TARA and Flush that lose many packets. Also the throughput graph that combines analytical and simulated throughput (Figs. 3(*c*) and 4(*c*)) shows the convergence between REFIACC analytical and simulation one. On the other hand a clear deviation is seen for the other protocols, especially for TARA-min and Flush-min. This is due to their wrong calculation of slot period that supposes an ideal link, and the reality is that the links are lossy, and the deviation becomes larger if the resulted topology contains many lossy links. The same logic applies to TARA-aver and Flush-aver, but with a less intensity, as their values are not so short, which allow to retransmit in the loss case, thus the convergence between the analytical and simulation graphs. But with high loss percentage, their retransmissions are insufficient.

9.2. Packet reception ratio

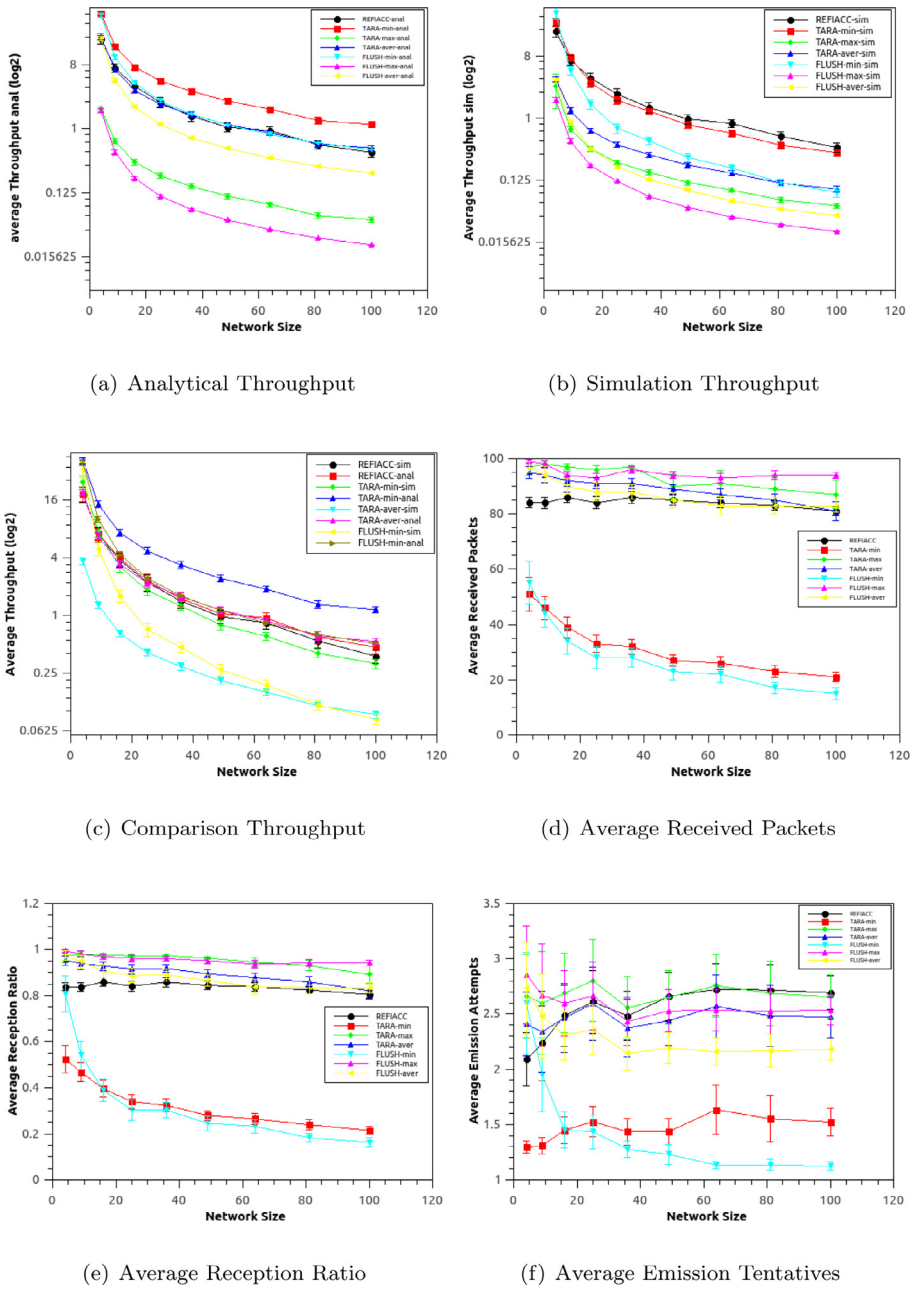
The average Packet Reception Ratio is the ratio between packets successfully received at base-station, to the total number of packets generated by the sources. This metric reflects the impact of interference, congestion and links loss on the packets goodput, while the number of received packets in comparison to the ratio adds a mean of efficiency.

9.2.1. Average received packets graphs

It is clear through Figs. 3(*d*) and 4(*d*) that TARA-max and Flush-max represent the better received packets results. This can be explained by the fact that when the slot period is maximal, the number of retransmission will be sufficient even if the link is very bad. On the other hand for TARA-min and Flush-min, the results are the worst; this is for the same explanation of the precedent graphs. In fact, their slot period is so short and they apply the same rate for every node in the network without considering real link capacities. Thus, they don't have enough time to retransmit the lost packets. For REFIACC, its slot period is well calculated to assign the appropriate rate for every node while avoiding interferences and allowing the retransmission of the lost packets in order to ensure an acceptable number of received packets.

9.2.2. In the average reception ratio graphs

(Figs. 3(*e*) and 4(*e*)): the results came to ensure the previous graph. TARA-max and Flush-max have the better ratio while TARA-min and Flush-min have the worst ones. On the other hand, REFIACC has a very acceptable ratio (more than 80%) because its conception takes into account the real links characteristics.



(a) Analytical Throughput

(b) Simulation Throughput

(c) Comparison Throughput

(d) Average Received Packets

(e) Average Reception Ratio

(f) Average Emission Tentatives

Fig. 3. Evaluation Results for Network Density 3.

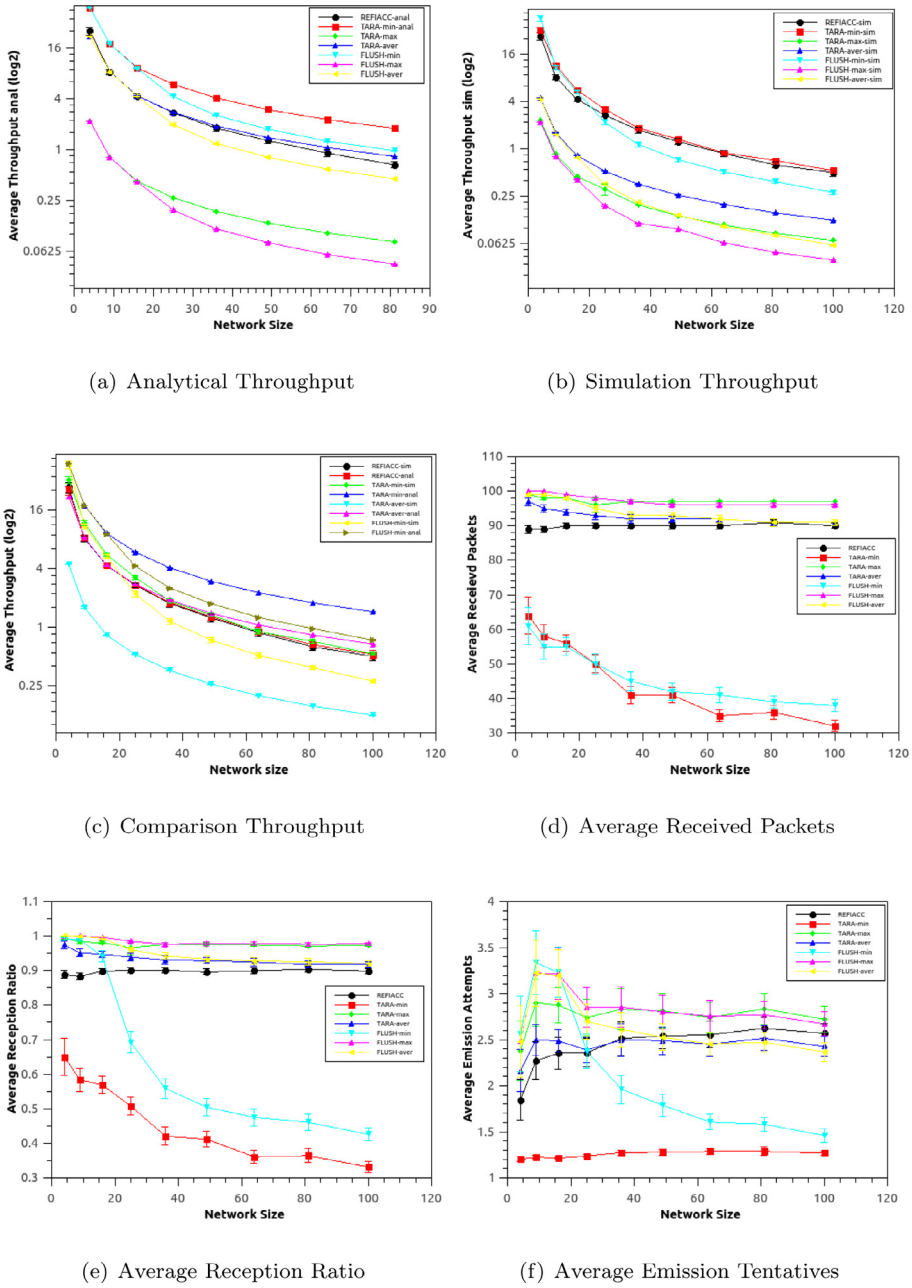
9.3. Number of retransmissions:

As depicted in Fig. 3(*f*) and Fig. 4(*f*), TARA-min and Flush-min represent less number of retransmissions than REFIACC. Generally speaking, the number of retransmission (or transmission tentatives) has a close relation with the resulted topology and more exactly the links quality. If the link quality is good, no need to retransmit packets, as they will arrive to the one hop receiver in the first send. But if the links are lossy (0.05% link reception probability), more emission tentatives are needed, even if the protocol conception is good, which is the case of REFIACC that retransmits packets as needed until the packet reaches the one hop receiver. The slot period of REFIACC is calculated in relation to the link real capacity to allow these retransmissions. On the other hand, and especially for TARA-min and Flush-min, their slot period is sufficient to send one packet, and sometimes maximum two packets. So, globally they success to transmit only on good links and they never arrive to transmit over bad links, which justify that their average transmission attempts is always reduced, but when looking on the previous graphs (number of received packets and average Reception Ratio) it confirms our logic interpretations as the number of received packets for TARA-min and Flush-min are always less than the others.

Finally, the comparison of the graphs shows that REFIACC is the best one as it ensures a very acceptable trade-off (maximal) between the throughput values, the number of received packets and reception ratio. This is due to its well slot calculation.

10. Conclusion and future work

With the increasing use of WSNs, new applications requirements appear asking for more throughput and reliability over



(a) Analytical Throughput

(b) Simulation Throughput

(c) Comparison Throughput

(d) Average Received Packets

(e) Average Reception Ratio

(f) Average Emission Tentatives

Fig. 4. Evaluation Results for Network Density 15.

the lossy and capacity-diversity nature networks. These challenges become harder when the network is dense as the nodes contend for the shared medium leading to more packets loss. In this study, a scheme named REFIACC (Reliable, Efficient, Fair and Interference-Aware Congestion Control) was presented that takes into account the links diversity and interferences when constructing the scheduling send policy in order to reach maximum fair throughput. The extensive simulations we have done show its effectiveness in terms of throughput and packet reception ratio compared to pertinent literature works (Flush and TARA).

For our future work, we plan to enhance REFIACC with a better reliable and recovery scheme for loss-intolerant applications. Also, we plan to take into account transient congestion and interference happening by a mitigating scheme that reacts through an adequate congestion detection method. On the other hand, we plan to construct a scheme that deal with persistent change in interference and capacity parameters by a threshold-based detection policy in order to invoke re-scheduling rather than its happening in a periodic manner like in QCRA scheme. The schedule has to be built in a manner that minimizes as possible the changes by avoiding from a scratch construction in order to reduce the calculus and communication overhead. In addition, we plan to focus in more detail on the conception of our schedule scheme in multichannel scenario in order to enhance throughput and slot reuse. Also a comparison with multi channel works is in our agenda. Finally, we are considering a deeper evaluation of REFIACC or the future enhancement versions with testbed experiments.

References

[1] M. Kafi, D. Djenouri, J. Ben-Othman, N. Badache, Congestion control protocols in wireless sensor networks: a survey, Commun. Surv. Tut. IEEE 16 (3) (2014) 1369–1390.

[2] M.A. Kafi, D. Djenouri, J. Ben Othman, A. Ouadjaout, N. Badache, Congestion detection strategies in wireless sensor networks: a comparative study with testbed experiments, Procedia Comput. Sci. 37 (2014a) 168–175. The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014).
[3] M.A. Kafi, D. Djenouri, J. Ben Othman, A. Ouadjaout, M. Bagaa, N. Lasla, N. Badache, Interference-aware congestion control protocol for wireless sensor networks, Procedia Comput. Sci. 37 (2014b) 181–188. The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014).
[4] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, I. Stoica, Flush: a reliable bulk transport protocol for multihop wireless networks, in: Proceeding of the 5th International Conference on Embedded Networked Sensor Systems, SenSys, 2007, pp. 351–365.
[5] J. Kang, Y. Zhang, B. Nath, TARA: topology-aware resource adaptation to alleviate congestion in sensor networks, IEEE Trans. Parallel Distrib. Syst. 18 (7) (2007) 919–931.
[6] C.T. Ee, R. Bajcsy, Congestion control and fairness for many-to-one routing in sensor networks, in: ACM, SenSys, 2004, pp. 148–161.
[7] F. Bian, S. Rangwala, R. Govindan, Quasi-static centralized rate allocation for sensor networks, in: Proceeding of 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON, 2007, pp. 361–370.
[8] F.B. Hussain, Y. Cebi, G.A. Shah, A multievent congestion control protocol for wireless sensor networks, EURASIP J. Wirel. Commun. Netw. 2008 (2008) 1–12.
[9] M. Rahman, M. Monowar, C. Hong, A capacity aware data transport protocol for wireless sensor network, in: Proceeding of the International Conference on Computational Science and Its Applications, ICCSA, in: Lecture Notes in Computer Science, 2009, pp. 491–502.
[10] C.-Y. Wan, S.B. Eisenman, A.T. Campbell, Energy-efficient congestion detection and avoidance in sensor networks, ACM Trans. Sen. Netw. 7 (4) (2011) 1–31.
[11] M. Yao, C. Lin, P. Zhang, Y. Tian, S. Xu, TDMA scheduling with maximum throughput and fair rate allocation in wireless sensor networks, IEEE ICC 2013 - Ad-hoc and Sensor Networking Symposium, 2013.
[12] M. Palattella, P. Thubert, T. Watteyne, Q. Wang, Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e, Technical Report, 2015.
[13] P. Thubert, An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4, Technical Report, 2015.
[14] D. Dujovne, T. Watteyne, X. Vilajosana, P. Thubert, 6TiSCH: deterministic IP-enabled industrial internet (of things), Commun. Mag. IEEE 52 (12) (2014) 36–41.
[15] M. Palattella, N. Accettura, M. Dohler, L. Grieco, G. Boggia, Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks, in: Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on, 2012, pp. 327–332.
[16] M. Palattella, N. Accettura, L. Grieco, G. Boggia, M. Dohler, T. Engel, On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH, Sens. J. IEEE 13 (10) (2013) 3655–3666.
[17] N. Accettura, M. Palattella, G. Boggia, L. Grieco, M. Dohler, Decentralized traffic aware scheduling for multi-hop Low power lossy networks in the internet of things, in: IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013, pp. 1–6.
[18] M. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. Grieco, T. Engel, On-the-Fly bandwidth reservation for 6TiSCH wireless industrial networks, Sens. J. IEEE 16 (2) (2016) 550–560.
[19] X. Vilajosana, K. Pister, Minimal 6TiSCH Configuration, Technical Report, 2016.
[20] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, K. Pister, A realistic energy consumption model for TSCH networks, Sens. J. IEEE 14 (2) (2014) 482–489.
[21] M.A. Kafi, J. Ben-Othman, M. Bagaa, N. Badache, CCS_WHMS: a congestion control scheme for wearable health management system, J. Med. Syst. accepted for publication 39 (2015) 1–10.
[22] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, Sensys, 2009.
[23] H.L. A. C., P. Levis, Improving wireless simulation through noise modeling, IPSN, 2007.
[24] F. Chung, L. Lu, Connected components in random graphs with given expected degree sequences, Ann. Comb. 6 (2002) 125–145.
[25] www-304.ibm.com/ibm/university/academic.