Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 29 (2006) 3397–3415
computer communications
www.elsevier.com/locate/comcom

Energy efficiency of collision resolution protocols ☆

Aran Bergman, Moshe Sidi *

Electrical Engineering Department, Technion—Israel Institute of Technology, Haifa 32000, Israel

Available online 13 March 2006

Abstract

Energy consumption of the medium access control (MAC) algorithm is one of the key performance metrics in today's ubiquitous wireless networks of battery-operated devices. We concentrate on random access MAC algorithms called Collision Resolution Protocols (CRPs) that have the best stable properties and excellent delay characteristics for a large population of "bursty" users. The main concern of the analysis of CRPs has so far been the stability conditions, the throughput-delay tradeoffs and how the algorithms can be optimized for these properties. The contribution of our work is the introduction of a novel utility function that reflects the tradeoff between the energy consumption induced by a MAC protocol and its throughput, thus representing the energy efficiency of the algorithm. We exemplify the use of this utility function by analyzing several CRPs, including full and limited sensing algorithms. In particular, we introduce a modification of the "0.487" algorithm that improves its energy efficiency.

Keywords: Wireless MAC; Energy efficiency; Collision resolution protocols; Performance analysis

1. Introduction

In many wireless networks the preferred *medium access control* (MAC) mechanism is a random access algorithm. It is known that for a large group of "bursty" users, such algorithms display better delay characteristics than TDMA schemes. Moreover, these random access algorithms are usually easier to implement and deploy than an access algorithm that is based on *scheduling*. One needs only to consider the current deployment of the 802.11 wireless local area networks (WLANs). The standard [1–3] provides two methods of accessing the medium; one (namely, DCF) using the random access algorithm known as *carrier sense multiple access with collision avoidance* (CSMA/CA), and the other which is based on scheduled access and is termed PCF. To date, none or very few implementations support the PCF function. Thus, random access algorithms merit attention when energy efficiency is addressed.

Of all the random access algorithms suggested in the literature, the family of algorithms that displays the best stable properties is the *collision resolution protocols* (CRPs). Many papers ([5–9,12,13,15,16] to name a few) have been devoted to suggesting new and improved collision resolution algorithms and to devising methods of analyzing their performance. This analysis concentrated mainly on finding the conditions for the stability of the algorithms, the maximum sustainable throughput, the expected packet delay and how each could be optimized. Today, when hand-held, battery-operated devices are used more and more, the energy requirements of an algorithm are just as important. Users might be willing to sacrifice throughput or delay to make better use of the battery energy they have available, and extend the operation time of their devices.

In this paper, we propose a utility function that measures the energy efficiency of an algorithm, given its parameters and the system's requirements (e.g., required throughput). We aim to optimize this function and show how this optimum can be calculated and obtained.

To the best of our knowledge, only [18,14] address the energy efficiency of collision resolution protocols. In [18], a new method of splitting the allocation interval, based on the residual energy of each node is suggested. The model

☆ This research was supported by the Israel Short Range Communication (ISRC) consortium.
* Corresponding author. Tel.: +972 4 8294650; fax: +972 4 8295757.
E-mail addresses: aran@tx.technion.ac.il (A. Bergman), moshe@ee.technion.ac.il (M. Sidi).

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.01.016

used is of finite population with finite initial energy at each node, and the suggested algorithm is compared to the FCFS algorithm. In [14], a new protocol that is based on known multiplicity (i.e., when a receiver knows how many transmitters collide) through energy measurements and a rather elaborate information in the feedback is presented. The suggested algorithm is compared with several other algorithms, among which are 802.11 and FCFS, that do not assume the known multiplicity model. The comparison is based on the expected values of the participation time, number of transmissions, collision resolution interval and total power consumption, given that k packets collide in the beginning of the collision resolution interval. We utilize the infinite population model, and find the optimized energy efficiency of each algorithm we describe. We employ the classical ternary feedback and compare different algorithms according to the unconditional expected values of throughput and energy consumption. This is highly important, as most algorithms can be optimized so that the vast majority of collisions contain no more than 2 packets. Even if a given algorithm outperforms another for all $k > 2$, it can still be inferior to the latter.

This paper is organized as follows. In Section 2.1, we describe the underlying model we use. Section 2.2 presents the utility function we propose as a metric to the energy efficiency of a MAC protocol, which incorporates the tradeoff between the system's throughput and its energy requirements. Section 3.1 describes two versions of the *Window Access* (WA) algorithm. In Section 3.2, we describe the celebrated *first-come-first-served* (FCFS) algorithm and analyze its energy efficiency. Section 3.3 is devoted to presenting numerical data and to discussing these results. Section 4 describes and analyzes limited sensing algorithms, namely the *free access* algorithm in Section 4.1 and the *last-come-first-served* algorithm in Section 4.2. Numerical data of the limited sensing algorithms is presented in Section 4.3. We conclude with Section 5, where we also outline some future work on the subject of energy efficiency of CRPs.

2. Energy efficiency

2.1. Model description and definitions

Our model and assumptions are basically those of standard multiple access protocols [17]. We assume the channel is *slotted*; that is, the time is divided into identical intervals called *slots* that are the time units used. A user can attempt to transmit a packet, whose length is one slot, only at the beginning of slots. We define *slot* T as the interval $[T, T+1)$. At the end of slot T, the users get information about the activity in the channel in that slot. We denote this feedback by F_T. Two kinds of feedbacks are considered:

Binary feedback, where the feedback can discern between 0 or 1 ($F_T = NC$), or more than one ($F_T = C$) transmission attempts in a slot.

Ternary feedback, where the feedback can discern between 0 ($F_T = 0$), 1 ($F_T = 1$) or more than one ($F_T = C$) transmission attempts in a slot.

This feedback might be available to the users by means of a central station that detects all transmissions in the channel and informs the users at the end of a slot. However, if the users can detect all transmissions in the channel, such a central station is not necessary.

We assume that the channel is error-free, so the only reason for erroneous delivery of packets is simultaneous transmission of two or more packets. This event is called *collision*, and all packets participating in a collision must be re-transmitted. New packets arrive to the system according to a Poisson process with rate λ packets per slot, reflecting the behavior of an infinite population of "bursty" users.

An algorithm which requires all users to continuously monitor the feedback information is called *Full Sensing Algorithm* or *Continuous Sensing Algorithm*. An algorithm which requires users to monitor the feedback information only when they have a packet to transmit, is called *Limited Sensing Algorithm*.

Each user invests a certain amount of energy, e_x, in each transmission attempt. We assume that this amount is constant for all users and all transmissions and is equal to 1 energy unit ($e_x = 1$). Since collisions might occur, a packet may be transmitted several times before it is successfully delivered; let P be the number of transmissions. If the system has a steady state, we define $\overline{P}$ as the expected normalized energy required to successfully transmit a packet.

For the limited sensing algorithms analyzed in Section 4, we also consider the energy required for monitoring the channel feedback in a single slot, denoted e_m. We assume that this energy is also constant for all users and for all slots and that $e_m = \xi e_x$, where $\xi \geqslant 0$, or, when normalizing by e_x, $e_m = \xi$.

2.2. The utility function

Our goal is to optimize the performance of the MAC algorithm with respect to the expected energy required to successfully transmit a packet – $\overline{E}$. With this objective in mind, we would like to evaluate different CRPs, and calibrate their parameters so that we get minimal $\overline{E}$. However, taking only $\overline{E}$ into consideration could lead to unwanted results in other important performance metrics. For instance, transmitting no packets yields $\overline{E} = 0$ which is the best possible, but then no packets will be transmitted successfully.

Consequently, we propose a utility function that takes the tradeoff between the throughput and $\overline{E}$ into account

$$U = \frac{\lambda}{\overline{E}^{\mu}} \tag{1}$$

which we aim to maximize, where λ is the throughput. The parameter μ is determined according to how important the energy requirements are with respect to the throughput.

If the system designer wishes to sacrifice more throughput to get lower $\overline{E}$, she will set a higher value of μ. If, on the other hand, one is more interested in the system throughput, one will set a lower μ.

Since for limited sensing algorithms a user is required to process the feedback only when it has a packet to transmit, and stops listening when the packet is finally transmitted successfully, the energy required to "tune in" on the feedback is equal to ξD, where D is the delay of the packet, measured from the time it arrived to the system until it was successfully transmitted. Thus, we have $\overline{E} = \overline{P} + \xi\overline{D}$, and we can use the following utility function for the limited sensing algorithms:

$$U_L = \frac{\lambda}{\left[\overline{P} + \xi\overline{D}\right]^{\mu}}. \tag{2}$$

This might seem unnecessary, since many view the transmission as far more energy-consuming than the receiving process. However, in many situations the energy requirements of amplifying, demodulating and decoding the received signal are comparable with the energy required to transmit a packet. (See [19] and references therein.).

For the full sensing algorithms, we take $\xi = 0$. Thus, for the full sensing algorithms we use U_F as the utility function, which is defined as

$$U_F = \frac{\lambda}{\overline{P}^{\mu}}. \tag{3}$$

Note that we neglect the energy required to listen to the feedback in this suggested utility function since in all full sensing algorithms users have to listen to the feedback at every slot, so there is no real difference between the required "listening" energy of any two full sensing algorithms. Furthermore, since we are dealing with the infinite population model, $\overline{E}$ does not take a finite value for any other value of ξ.

Note that the utility function (1) is not limited to measuring the energy efficiency of CRPs only. It can be used to evaluate any MAC protocol, if we take E to be the energy required to deliver an information unit (which could be bit, byte or packet, for instance), and if we normalize the energy units so that $E \geqslant 1$ (otherwise, the meaning of μ might change).

3. Full sensing algorithms

In this section, we examine the energy efficiency of full sensing algorithms. We use the utility function presented in (3) to evaluate the performance of different algorithms, and we find the conditions for maximizing this function. We start with an algorithm that is easy to analyze, to illustrate the calculations.

3.1. The Window Access (WA) algorithm

The idea of controlling the access of new packets according to a window was first suggested by Gallager in [7]. This algorithm, sometimes called the *epoch mechanism* (as in [7,17]), divides the time axis into consecutive windows, or epochs, each of maximum length Δ slots. The ith window is the time interval $[i\Delta,(i+1)\Delta)$. Packets that arrive during the ith window are blocked until all the packets from the $(i-1)$th window have been resolved. In this section, we discuss the *simple* WA (SWA), suggested in [15], where packets delay the start of the next *collision resolution interval* (CRI), if necessary, until a "full" window can be chosen. This simplifies the analysis of the algorithm. The more sophisticated version, where a shorter window may be chosen, is, actually, the First-Come-First-Served algorithm, which is discussed in detail in Section 3.2. The window length, Δ, is a system parameter and could be tuned to optimize the performance of the system. We aim to maximize U_F. Apart from stating the first-time transmission rule, we must also define the conflict resolution algorithm used. Here, we use the *modified tree* algorithm, sometimes referred to as *level skipping*, also suggested in [15]. This means that definite collisions, i.e., collisions that follow an idle slot which immediately follows a collision, are avoided. We do not discuss or analyze the standard tree algorithm, also known as the Capetanakis–Tsybakov–Mikhailov (CTM) algorithm [5], since it obviously wastes more energy (due to avoidable definite collisions).

3.1.1. WA algorithm description

This is a full sensing ternary feedback algorithm and all users are always aware of the window being handled in the current CRI. Algorithm 1 is based on an implementation suggestion in [15] and is carried out by each user. The parameters used in the algorithm are CRICounter, that identifies the end of a CRI; WindowCounter, that identifies the current window; TransmitCounter, that marks the node's place in the stack; and Flag, that implements the level skipping. The coin flip used in the algorithm is a Bernoulli random variable with probability p of flipping 0.

Algorithm 1. The regular Window Access algorithm

```
Flag ← 0,  CRICounter ← 1,  WindowCounter ← 0,
TransmitCounter ← NULL
loop
  Wait for feedback
  if F_T = C then
    CRICounter ← CRICounter + 1
    Flag ← 1
    if this user transmitted in slot T then
      FlipResult ← result of coin flip
      if FlipResult = 0 then
        TransmitCounter ← 0
      else {FlipResult = 1}
        TransmitCounter ← 1
    else if TransmitCounter ≠ NULL then
      TransmitCounter ← TransmitCounter + 1
    else if F_T = 1 then
      CRICounter ← CRICounter - 1
      if this user transmitted in slot T then
```

```
        TransmitCounter ← NULL
    if TransmitCounter ≠ NULL then
        TransmitCounter ← TransmitCounter - 1
    Flag ← 0
else if F_T = 0 and Flag = 0 then
    CRICounter ← CRICounter - 1
    if TransmitCounter ≠ NULL then
        TransmitCounter ← TransmitCounter - 1
else {F_T = 0 and Flag = 1}
    if TransmitCounter = 1 then
        FlipResult ← result of coin flip
        if FlipResult = 0 then
            TransmitCounter ← TransmitCounter - 1
if TransmitCounter = 0 then
    Re-transmit in slot T + 1
if CRICounter = 0 then
    CRICounter ← 1
    if T > Δ (WindowCounter + 1) then
        WindowCounter ← WindowCounter + 1
        if this user has a packet that arrived in the
        WindowCounter^th window
        then
            Transmit at slot T + 1
            TransmitCounter ← 0
        else
            TransmitCounter ← NULL
```

3.1.2. Energy efficiency for the WA algorithm

The first step in calculating $\overline{P}$ is calculating the expected cumulative number of transmissions in a CRI, given that it starts with k packets. Let $\overline{X}_k$ be the expected value of X, given that the CRI starts with k packets. Let V denote the number of transmissions in a CRI. We have $\overline{V}_0 = 0$ and $\overline{V}_1 = 1$. Given that the CRI starts with $k \geqslant 2$, and that i users are on the left branch of the binary tree, according to the algorithm, we get

$$\overline{V}_{k,i} = \begin{cases} k + \overline{V}_i + \overline{V}_{k-i}, & 1 \leqslant i \leqslant k; \\ \overline{V}_k, & i = 0. \end{cases}$$

Taking the expectation over i, we have

$$\overline{V}_k = \frac{k(1-Q_0^k) + \sum_{i=1}^{k-1}(Q_i^k + Q_{k-i}^k)\overline{V}_i}{1 - Q_0^k - Q_k^k}, \quad \forall k \geqslant 2,$$

where $Q_i^k = \binom{k}{i} p^i (1-p)^{k-i}$. Since each CRI starts with a window of Δ, we can simply write

$$\overline{V} = \sum_{n=0}^{\infty} \overline{V}_n \Pr(k = n) = \sum_{n=0}^{\infty} \overline{V}_n \frac{(\lambda\Delta)^n}{n!} e^{-\lambda\Delta}. \tag{4}$$

Since each CRI is independent, thanks to the method used to choose the packets eligible for transmission at the beginning of each CRI, the expected number of transmissions per packet is the ratio between the expected cumulative number of transmissions in a CRI and the expected number of packets that are successfully transmitted in a CRI. In the WA algorithm all packets in the beginning of a CRI are eventually transmitted in the same CRI. Thus, the expected number of packets involved in a CRI is $\lambda\Delta$ and therefore

$$\overline{P} = \overline{V}/(\lambda\Delta). \tag{5}$$

We believe it can be shown that if we hold λ constant, $\overline{P}$ is a monotonically increasing function of Δ (see Fig. 2(a)), meaning that one would prefer to set Δ as small as possible in order to maximize U_F. However, to sustain a stable throughput of λ packets per slot, one cannot take any value of Δ. We know [17] that the algorithm is stable so long as

$$\overline{L} < \Delta, \tag{6}$$

where L is the length of a CRI. Thus, one should choose the smallest Δ that does not violate (6). This means that maximizing U_F, when λ is held constant, can be achieved by setting Δ to such a value that causes the system to operate near the algorithm's capacity, i.e., close to its stability threshold. If we can also control λ in some manner, we would like to know what would be the best choice of λ and Δ. If we define the expected number of packets in a window $z \triangleq \lambda\Delta$, we notice that $\overline{P}$ is a function of z only. The expected CRI length, $\overline{L}$, is also a function of z only. Noticing that (6) can be re-written as

$$\lambda < \frac{z}{\overline{L}} = \frac{z}{\sum_{n=0}^{\infty} \overline{L}_n \frac{z^n}{n!} e^{-z}} \tag{7}$$

we can write

$$U_F = \frac{\lambda}{\overline{P}^{\mu}} < \frac{z}{\overline{L}[\overline{P}]^{\mu}} = \frac{z^{\mu+1}}{\sum_{n=0}^{\infty} \overline{L}_n \frac{z^n}{n!} e^{-z} \left[\sum_{n=0}^{\infty} \overline{V}_n \frac{z^n}{n!} e^{-z}\right]^{\mu}} \triangleq G_{\mu}(z). \tag{8}$$

3.1.3. Tree pruning

As Gallager noticed in [7], if the arrival time of the packets is used to split them into subsets, when a collision is followed by another collision of the packets on the left branch, it is best if the right branch is incorporated into the next CRI. This algorithm is termed *tree pruning* or *clipped tree*. The method used to analyze the energy efficiency of this algorithm is similar to the method used in Section 3.1.2. Using the same notations, we have that $\overline{V}_0 = 0$ and $\overline{V}_1 = 1$. Given that the CRI starts with $k \geqslant 2$, and that i users are on the left subset of the allocation interval, according to the algorithm we get

$$\overline{V}_{k,i} = \begin{cases} k + \overline{V}_i, & 2 \leqslant i \leqslant k; \\ k + 1 + \overline{V}_{k-1}, & i = 1, \quad \forall k \geqslant 2; \\ \overline{V}_k, & i = 0. \end{cases} \tag{9}$$

Taking the expectation over i, we have

$$\overline{V}_k = \frac{k(1-Q_0^k) + Q_1^k(1 + \overline{V}_{k-1}) + \sum_{i=2}^{k-1} Q_i^k \overline{V}_i}{1 - Q_0^k - Q_k^k}, \quad \forall k \geqslant 2. \tag{10}$$

Since "clipping" the tree does not affect the independence between different CRIs, and as each CRI starts with packets chosen from a window of exactly Δ slots, we can still use (4) to calculate $\overline{V}$. However, (5) no longer holds for this algorithm, as some of the packets that collided in the first slot of a CRI might be incorporated into the next CRI, rather than resolved in the current CRI. To calculate $\overline{P}$, we must find the expected number of packets that are delivered in a CRI, denoted by N. This has already been achieved in [17], and found to be

$$\overline{N}_k = \frac{Q_1^k(1+\overline{V}_{k-1}) + \sum_{i=2}^{k-1} Q_i^k \overline{N}_i}{1 - Q_0^k - Q_k^k}, \quad \forall k \geqslant 2.$$

To find the unconditional value, $\overline{N}$, we exploit the independence between CRIs, and use

$$\overline{N} = \sum_{n=0}^{\infty} \overline{N}_n \frac{(\lambda\Delta)^n e^{-\lambda\Delta}}{n!}.$$

Now, that we have the expected number of packets transmitted in a CRI, and recalling that $z \triangleq \lambda\Delta$, we can calculate $\overline{P}$ using

$$\overline{P} = \frac{\overline{V}}{\overline{N}} = \frac{\sum_{n=0}^{\infty} \overline{V}_n \frac{z^n e^{-z}}{n!}}{\sum_{n=0}^{\infty} \overline{N}_n \frac{z^n e^{-z}}{n!}}.$$

Again, we believe $\overline{P}$ can be shown to be monotonically increasing with respect to z (see Fig. 2(a)), so if we hold λ constant, we get that it is monotonically increasing with Δ. Considering U_F, we notice that one would prefer that the system would operate near capacity. For this algorithm, stability is maintained as long as $\lambda < \overline{N}/\overline{L}$ [17], so we get

$$U_F = \frac{\lambda}{\overline{P}^{\mu}} < \frac{\overline{N}^{\mu+1}}{\overline{L} \cdot [\overline{V}]^{\mu}} = \frac{\left[\sum_{n=0}^{\infty} \overline{N}_n \frac{z^n e^{-z}}{n!}\right]^{\mu+1}}{\sum_{n=0}^{\infty} \overline{L}_n \frac{z^n e^{-z}}{n!} \left[\sum_{n=0}^{\infty} \overline{V}_n \frac{z^n e^{-z}}{n!}\right]^{\mu}} \triangleq G_{\mu}(z). \tag{11}$$

3.2. The first-come-first-served (FCFS) algorithm

The first-come-first-served algorithm was suggested by Gallager in [7]. This algorithm splits the colliding packets into two subsets according to the packet arrival time, rather than coin flips or node identity. The collision resolution algorithm used here is the modified tree algorithm with *tree pruning*, or *clipped tree* mechanism, as it is described in Section 3.1.3. The packets chosen for transmission in each slot are the packets that arrived in a time interval specified by the algorithm; this interval is called the *allocation interval*. When a new CRI begins, the allocation interval chosen for this CRI is of maximum length Δ, but can be shorter, if a "full" window cannot be chosen at the current slot and therein lies the difference from the algorithm analyzed in Section 3.1.3. The algorithm is called FCFS since the packets are delivered in the order they were generated. It is also known as the 0.487 algorithm, for its maximum achievable steady-state throughput, which is the highest known to date (apart for a minor improvement suggested by Mosely and Humblet in [16]).

3.2.1. The FCFS algorithm description

At each slot boundary, the algorithm specifies the allocation interval by two parameters, $S(T)$ and $\ell(T)$, as $[S(T), S(T)+\ell(T))$. The parameter $\sigma(T)$ is the status of the subset that should be transmitted in slot T. It can take either L (left subset) or R (right subset). Algorithm 2 describes what each node does.

Algorithm 2. The FCFS Algorithm

$\sigma(1) \leftarrow R$, $S(1) \leftarrow 0$, $\ell(1) \leftarrow 1$,
{T $\leftarrow 1$}
loop
  if this node has a packet that arrived in $[S(T), S(T)+\ell(T))$ **then**
    Transmit in slot T
  Wait for feedback
  if $F_T = C$ **then**
    $S(T+1) \leftarrow S(T)$
    $\ell(T+1) \leftarrow \frac{\ell(T)}{2}$
    $\sigma(T+1) \leftarrow L$
  if $F_T = 1$ and $\sigma(T) = L$ **then**
    $S(T+1) \leftarrow S(T) + \ell(T)$
    $\ell(T+1) \leftarrow \ell(T)$
    $\sigma(T+1) \leftarrow R$
  if $F_T = 0$ and $\sigma(T) = L$ **then**
    $S(T+1) \leftarrow S(T) + \ell(T)$
    $\ell(T+1) \leftarrow \frac{\ell(T)}{2}$
    $\sigma(T+1) \leftarrow L$
  if $F_T \neq C$ and $\sigma(T) = R$ **then**
    $S(T+1) \leftarrow S(T) + \ell(T)$
    $\ell(T+1) \leftarrow \min(\Delta, T - S(T))$
    $\sigma(T+1) \leftarrow R$
{$T \leftarrow T+1$}

3.2.2. Energy efficiency for the FCFS algorithm

The method used to calculate $\overline{P}$ for the FCFS algorithm is similar to the method used in [9] and in [8] for calculating the packet delay. This method takes advantage of the regenerative nature of the transmission process in order to calculate the expected number of transmissions per packet.

Suppose that at the beginning of slot T all packets that arrived before time $S(T)$, where $S(T) < T$, have been successfully transmitted, and there is no information regarding the packets in the interval $[S(T), T)$ (see Fig. 1). The time interval $d = T - S(T)$ is called the "lag". A new CRI begins at time T and lasts L slots. This CRI begins with packets chosen from the interval $[S(T), S(T)+\ell)$, where $\ell = \min(d, \Delta)$. According to the FCFS algorithm, not all packets in the allocation interval are necessarily resolved

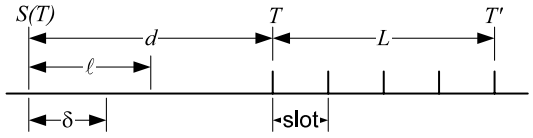

Fig. 1. Random variables used in the FCFS analysis.

in a CRI. However, all packets resolved in a CRI are packets that arrived in some time interval $[S(T), S(T)+\delta)$, which we refer to as the *resolved interval*. Moreover, at the end of the CRI, we have no a-priori knowledge about the packets that arrived in the interval $[S(T'), T')$, where $S(T') = S(T) + \delta$. Thus, at time T', we have a similar situation to what we had at time T, but with a new lag $d' = T' - S(T')$, and with a new allocation interval. Unlike the simple WA (with or without the tree pruning mechanism), the lag at the beginning of a CRI can be smaller than Δ, so ℓ does not necessarily equal Δ when a CRI begins. This is the source of the difficulty in analyzing this algorithm, as each CRI is no longer independent of the former CRI.

Let us label the packets according to the order of their arrival and denote the number of transmissions required to successfully deliver the ith packet by P_i. We would like to calculate the steady-state expected value of P_i, when it exists. According to the method in [9], we can calculate $\overline{P}$ via

$$\overline{P} = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} P_i = \overline{P_\infty} = \frac{Y}{C}, \tag{12}$$

where Y and C denote the expected cumulative number of transmissions and the expected number of packets successfully transmitted in a regeneration cycle, respectively. The regeneration points are CRI boundaries when $d = 1$. At these points, the random processes restart, including the process for P. The expected number of packets delivered in a regeneration cycle is

$$C = \lambda H, \tag{13}$$

where H is the expected length of the regeneration cycle. Upper and lower bounds on H were already found in [8,9], and are described in Appendix A. To find Y, we develop a system of equations and find upper and lower bounds on its solution, as in [8,9]. The derivation is described in Appendix A.

From (12) and (13) and the results of Appendix A, we have the following bounds on the expected number of transmissions until successful delivery of a packet:

$$\frac{Y^l}{\lambda H^u} = \overline{P}^l \leqslant \overline{P} \leqslant \overline{P}^u = \frac{Y^u}{\lambda H^l}$$

3.3. Numerical results and discussion

First, we plot $\overline{P}$ versus the expected number of packets per window, z, for the two versions of the WA algorithm (Fig. 2(a)).

The difference between the two versions of the WA algorithm can be explained by noticing that the most energy in a CRI is wasted in its first slot. The tree pruning increases the probability that a given slot will be the first slot of a CRI (for a given z), and since these slots waste most of the energy, $\overline{P}$ is higher when using the tree pruning mechanism. The value of $\overline{P}$ is similar for the two versions for small values of z since for $k \leqslant 2$, the algorithms behave exactly the same. For small values of z, the majority of the CRIs start with $k \leqslant 2$ (for $z = 1$, $\Pr(k \leqslant 2) \approx 0.92$).

We might be tempted to choose the simple WA algorithm over the WA with the pruning mechanism in view of Fig. 2(a), but we must consider the stability requirement. If we hold λ constant and control Δ, we know that to increase U_F we need to choose Δ as small as possible. To sustain a given throughput λ, we can choose a smaller Δ when using the tree pruning compared with the WA algorithm without tree pruning. This means that z for the tree pruning version can be smaller, but it does not automatically mean that $\overline{P}$ will be smaller. Consider an example where $\lambda = 0.2$ and suppose we could choose $\Delta = 20$ ($z = 4$) for the regular WA algorithm and $\Delta = 15$ ($z = 3$) for the tree pruning version. In this example, we get that $\overline{P}$ is still higher for the tree pruning version. Luckily, we know that all values of λ, for which the two algorithms are stable, can be achieved using $z < 1.3$. In this region, $\overline{P}$

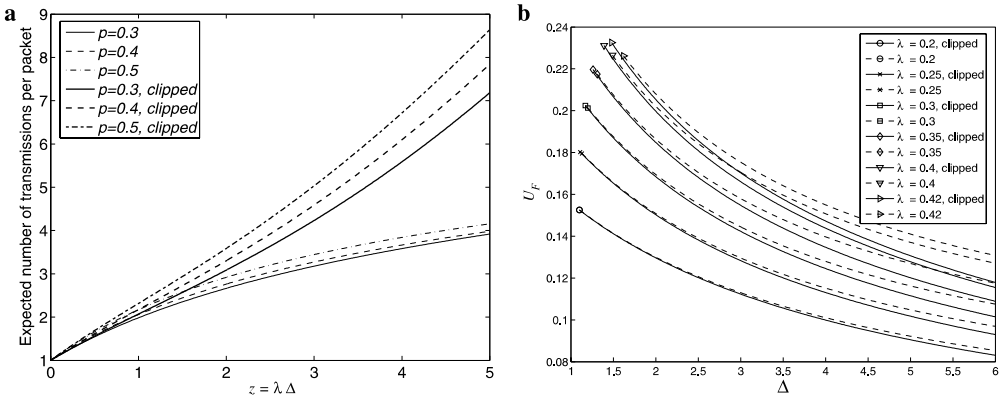

Fig. 2. (a) The expected number of transmissions per packet for the WA algorithms. (b) The utility function with $\mu = 1$ and $p = 0.5$ for the WA algorithms. The curves are presented only for Δ for which the algorithm is stable. The maximum value for each curve is marked with a symbol.

for the two versions of the WA algorithm is roughly the same, so the algorithm that can choose a smaller Δ (and thus smaller z) will be more energy efficient. This is also illustrated in Fig. 2(b), where we can see that for most values of Δ, the efficiency of the regular WA algorithm is better, but the pruning version can achieve higher values of U_F since for the same value of λ, it can choose lower Δ.

Note that, if we have to set Δ without knowing λ, and if we cannot control Δ after setting it, the algorithm we would choose is the simple WA algorithm, as its efficiency in this scenario will always be higher (for all λ for which the system is stable with the set Δ for both versions of the algorithm).

We would like to find the maximum value of U_F for a given μ for the WA algorithms, and what should be the values of λ, Δ and p to achieve this maximum. Using (8) (and similarly (11)), we can find the maximum U_F for a given μ by searching for the maximum of $G_\mu(z)$ (denoted by $G^*_\mu(z)$) and the maximizer, z^*_μ. Having found z^*_μ, we have only $\lambda^*_\mu\Delta^*_\mu$. However, we know that the maximum value of U_F is obtained when the system is operating near its capacity, so λ^*_μ is easily extracted by substituting z^*_μ in (7) (for the simple WA algorithm, for instance). Using standard optimization algorithms and taking the first 50 terms in the infinite series in (8) and (11), we have found the maximum and maximizers of U_F for the WA algorithms, for several values of μ. Table 1 contains these values, while Figs. 3(a) and (b) present $G_\mu(z)$ for some values of p and μ.

We note that the maximum achievable value of U_F for the WA algorithm with the tree pruning mechanism is slightly higher than the maximum U_F for the WA algorithm without tree pruning, for the same μ. We also note that as μ increases, z^*_μ decreases. This agrees with the way we view the tradeoff between the throughput and the energy invested in delivering a packet. When we are more interested in the throughput, i.e., when μ is smaller, we set z closer to the optimal z used to achieve the highest sustainable throughput. When, on the other hand, we are more interested in the energy consumption, we should set lower values for z, thus sacrificing the throughput to get lower energy consumption.

Table 1
Maximum Values and Maximizers of $G_\mu(z)$ for the simple WA algorithm (a) and the WA algorithm with tree pruning (b)

μ	$G^*_\mu(z)$	p^*_μ	z^*_μ	λ^*_μ	Δ^*_μ
(a)					
0.5	0.326	0.363	0.855	0.449	1.903
1	0.243	0.315	0.651	0.412	1.580
1.5	0.191	0.275	0.522	0.371	1.408
2	0.156	0.241	0.432	0.331	1.304
3	0.113	0.189	0.316	0.265	1.191
(b)					
0.5	0.332	0.393	0.867	0.464	1.867
1	0.246	0.332	0.659	0.422	1.562
1.5	0.192	0.284	0.526	0.376	1.398
2	0.157	0.246	0.434	0.334	1.298
3	0.113	0.191	0.316	0.266	1.189

Next, we compare the FCFS algorithm with the WA algorithms. Since for any given Δ, the capacity of the FCFS algorithm coincides with the capacity of the WA algorithm with tree pruning and $p = 0.5$, we compare only these two. Considering the results presented in Fig. 4, we see that, indeed, when Δ is set so that the algorithm operates near capacity, i.e., we set Δ to the smallest value that still permits stable behavior of the algorithm with the given λ, the efficiency of the two algorithms is identical (minor differences in the graph are due to simulation sampling errors and due to the step used in setting Δ), as is expected. When we set higher values for Δ, we see that the efficiency of the FCFS algorithm exceeds that of the WA algorithm. The reason lies in the fact that the FCFS does not "wait" so that a full window can be chosen. It can begin a CRI with an allocation window that is smaller than Δ. This happens when $d < \Delta$, and when it does, the expected number of packets that participate in such CRIs is smaller than $\lambda\Delta$, the expected number of packets participating in a CRI of the WA algorithm. In effect, the expected initial allocation interval in the FCFS algorithm is smaller, and we know that a smaller allocation interval means less collisions, when λ is held constant. Fig. 5 illustrates the same phenomena when we hold Δ constant and change the value of λ. Actually, we can deduce from Figs. 5(b) and 4, that if we set $\Delta \approx 1.3$, we get the best performance in the FCFS algorithm for any value of λ with respect to the energy efficiency. This does not mean, however, that for a given λ, we should set $\Delta = 1.3$. If we can change Δ according to a known λ, we would set Δ to the smallest value for which the system is stable. This value is smaller than 1.3 for $\lambda < 0.487$.

Considering the analysis of the FCFS presented above, we can suggest a different version of the FCFS with identical maximum stable throughput, but with better energy efficiency performance. Whenever a CRI begins while $d < \Delta$, the FCFS algorithm chooses $\ell = d$ (see Algorithm 2). We suggest changing the $\ell(T+1) \leftarrow \min(\Delta, T - S(T))$ line in the FCFS algorithm to

if $T - S(T) \geqslant \Delta$ **then**
  $\ell(T+1) \leftarrow \Delta$
else
  $\ell(T+1) \leftarrow \min(\zeta, T - S(T))$,

where ζ is a parameter of the algorithm, and $1 \leqslant \zeta \leqslant \Delta$. This change preserves the stability properties of the FCFS algorithm, while shortening the expected allocation interval for large Δ, thus we get lower $\overline{P}$ and higher values of U_F for the same values of Δ and λ. The results for $\zeta = 1.4$ are presented in Fig. 6. Choosing the optimum value for ζ has not been thoroughly investigated yet.

4. Limited sensing algorithms

In this section, we examine the energy efficiency of limited sensing algorithms, where a user has to listen to the feedback only when it has a packet to transmit. The utility

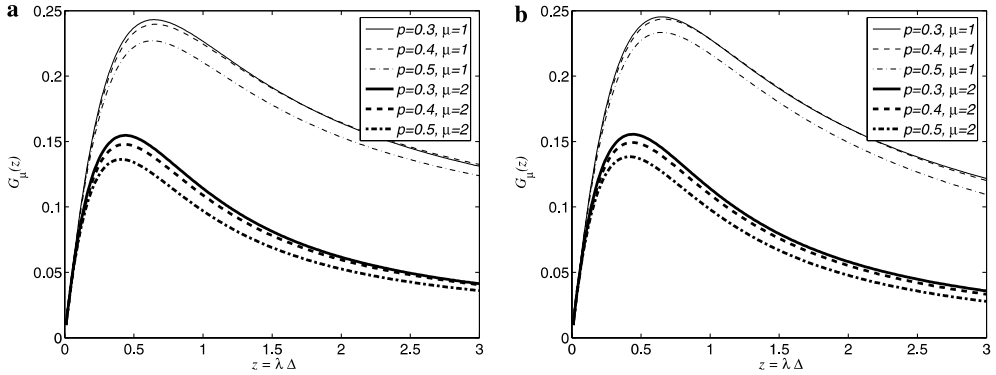

Fig. 3. The function $G_\mu(z)$ with $\mu = 1$ and $\mu = 2$ for the WA algorithm (a) and for the WA algorithm with the tree pruning mechanism (b).

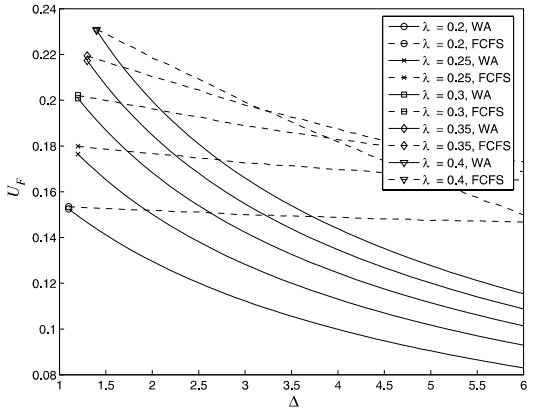

Fig. 4. The utility function with $\mu = 1$ and $p = 0.5$ for the WA algorithm with tree pruning and the FCFS algorithm. The curves are presented only for Δ for which the algorithm is stable. The maximum value for each curve is marked with a symbol.

function used to evaluate the energy-efficiency of limited sensing algorithms is presented in (2).

4.1. The free access (FA) algorithm

The free access algorithm was suggested and analyzed in [6,11]. The algorithm used to resolve collisions is the standard tree algorithm. However, in this algorithm, a user transmits a packet as soon as it is generated (at the next beginning of a slot, that is), as opposed to the obvious method, as described in [15], where a user waits until an ongoing CRI is terminated. The main advantage of taking this course of action is that a user does not have to monitor the feedback before it has a packet to transmit. The standard tree algorithm carried out by each user can be visualized as maintaining each user's level in a *stack*. A user transmits whenever it is at the first level (level 0), and updates its place in the stack (TransmitCounter in Algorithm 3) according to the feedback it hears (including feedbacks of slots in which the user did not transmit).

4.1.1. The FA algorithm description

For the simple version of the algorithm, only binary feedback is necessary. Ternary feedback should be used if we want to improve the algorithm by avoiding definite collisions, as was described in Section 3.1. Algorithm 3 is carried out by each node in the system.

Algorithm 3. The free access algorithm

loop
TransmitCounter ← 0
Wait for a packet to arrive
Transmit the packet at the next slot boundary and wait for feedback

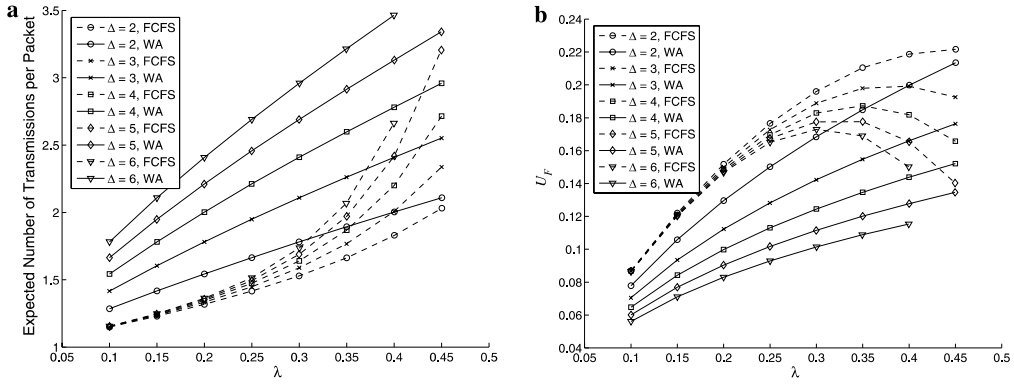

Fig. 5. Comparison between the WA algorithm with tree pruning and the FCFS algorithm with respect to $\overline{P}$ (a) and with respect to U_F when $\mu = 1$ (b). Only values of λ for which the system is stable are shown.

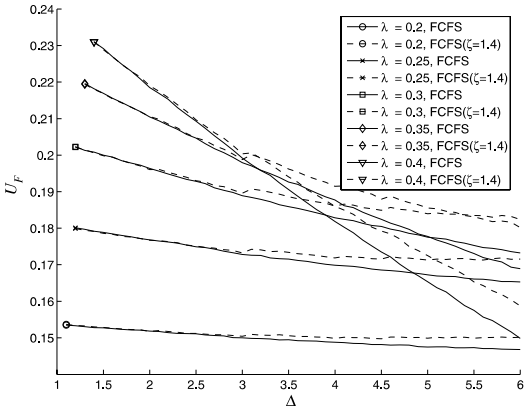

Fig. 6. Comparison of the utility function for the regular FCFS and the suggested improvement with $\zeta = 1.4$. Only values of Δ for which the system is stable are presented. The maximum value of each curve is marked with a symbol.

while $F_T \neq NC$ or TransmitCounter > 0 **do**
  if $F_T = C$ and TransmitCounter $= 0$ **then**
    Flip a binary coin with probability p for flipping 1
    if flipped 1 **then**
      TransmitCounter $\leftarrow 0$
    else
      TransmitCounter $\leftarrow$ TransmitCounter $+ 1$
  else if $F_T = C$ **then**
    TransmitCounter $\leftarrow$ TransmitCounter $+ 1$
  else $\{F_T = NC\}$
    TransmitCounter $\leftarrow$ TransmitCounter - 1
  if TransmitCounter $= 0$ **then**
    Transmit on slot $T + 1$
  Wait for feedback

4.1.2. Analyzing energy efficiency for the FA algorithm

Here, again, we exploit the regenerative nature of the algorithm. We notice that the regeneration points in this case are the CRI boundaries, as the lag of this algorithm is always 1. We therefore calculate the cumulative number of transmissions in a CRI and divide it by the expected number of packets transmitted in a CRI. Note that in this algorithm, new packets join an ongoing CRI. The analysis is based on the analysis presented in [11,6].

Define V_k as the cumulative number of transmissions in a CRI that starts with k packets transmitting on the first slot of the CRI. Following the algorithm's description, we have

$$V_{k,i} = \begin{cases} 0, & k = 0; \\ 1, & k = 1; \\ k + V_{i+X_L} + V_{k-i+X_R}, & k \geqslant 2, \end{cases} \tag{14}$$

where i is the number of packets (out of k) that flipped 1 after the collision, X_L is the number of packets that joined during the first slot of the CRI (the collision slot), and X_R is the number of packets that joined during the last slot of the "mini-CRI" that started with $i + X_L$ packets. Eq. (14) is the basis for calculating $\overline{V}$ and therefore $\overline{P}$. We must bear in mind that (14) is not a recursive formula, as X_R and X_L may take any non-negative integer value. A detailed derivation of a formula for $\overline{P}$ is described in [4] and produces

$$\overline{P} = 1 - \frac{K_a}{\lambda}\mathbf{S}(e^{-z}z;\lambda), \tag{15}$$

where $q = 1 - p$,

$$K_a = \frac{\frac{1}{p} - \frac{1}{q}}{\frac{1}{p}e^{-\lambda/p} - \frac{1}{q}e^{-\lambda/q}}, \quad K_a\left(p = \frac{1}{2}\right) = \frac{e^{2\lambda}}{1 - 2\lambda}$$

and the operator $\mathbf{S}$ is defined in [6,11], where a method for calculating $\mathbf{S}(f(\cdot);u)$ to a desired level of accuracy by using the Taylor expansion of $f(\cdot)$ and by exploiting the linearity of the $\mathbf{S}$ operator is also presented.

To evaluate the utility function we also need to calculate $\overline{D}$, the expected delay for a packet. Again, this was already solved in [6]. However, $\overline{D}$ as it is calculated in [6] is measured from the first transmission attempt. Since we are interested in the delay from the arrival time of the packet, and owing to the fact that all packets are transmitted for the first time in the first slot after the time they were generated, we can add $\frac{1}{2}$ to the delay calculated in [6], to obtain the required result.

4.2. The last-come-first-served (LCFS) algorithm

Limited sensing algorithms with a capacity that can be made arbitrarily close to the 0.487 throughput algorithm were independently suggested in [8,10] and [13]. The algorithm uses a parameter R, which determines how many consecutive idle slots are allowed during a CRI. This limits the number of consecutive times the "modified tree" mechanism (which prevents definite collisions) may be used, so it obviously limits the algorithm's performance. However, this is essential to prevent system deadlock. Without this limitation, a newly arrived packet will not be able to determine whether a CRI is in progress or not. When the number of consecutive idle slots during a CRI is limited to R, if a new packet observes $R + 1$ idle slots, it can determine that no CRI is in progress. As observed in [8,10,13], the capacity of the algorithm is 0.4493 for $R = 1$ and tends to Gallager's 0.487 algorithm as R increases. However, as R increases, so does the expected packet delay for low traffic, since, most likely, a packet waits $R + 1$ slots before it can be transmitted for the first time, and when it does, it most likely succeeds. Therefore, we have a tradeoff between the expected packet delay (for low traffic) and the algorithm's capacity. We also note that for $R = 1$, a binary-feedback version of the algorithm may be obtained, since the algorithm behaves exactly the same after $F_T = 0$ and $F_T = 1$.

Table 2
The last-come-first-served algorithm parameters

Parameter	Meaning
R	Maximum number of consecutive idle slots during a CRI.
σ	Status of the subset that should be transmitted in the next slot. Can take either L for Left subset or R for Right.
T	Time elapsed from the arrival of a packet, to the current time.
T_1	Time interval between the arrival time of the packet and the ending point of the allocation interval chosen at the beginning of the current CRI.
T_x	Collective amount of time already examined in the interval beginning with the time the packet arrived and ending at the current time.
L_A	Number of slots containing packets from class 1, from the arrival time of the packet until the current time.

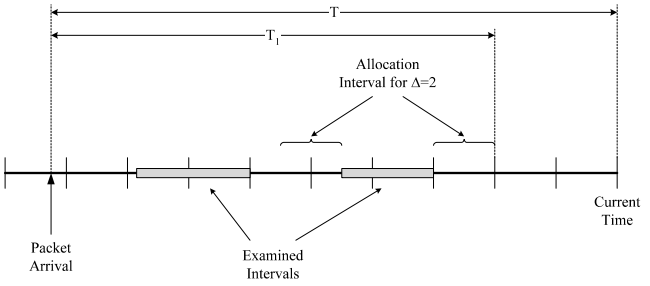

Fig. 7. Random variables used in the analysis and description of the LCFS algorithm.

4.2.1. The LCFS algorithm description

The algorithm executed by each user in the network is described in [8]. We give a different description of the same algorithm, which we feel is more readily implementable. Table 2 and Fig. 7 describe the parameters used in Algorithm 5. At every instance, the packets awaiting transmission are divided into two classes. Class 1 contains packets that cannot ascertain whether a CRI is in progress or not and packets that know a CRI is in progress, but do not know when this CRI started. Class 2 contains all the packets which already know that a CRI is in progress and also know when it started. Algorithm 5 uses the statement "a CRI just ended" in one of the conditions. This is established according to Algorithm 4. This algorithm should be executed only when the user is active (i.e., when it has a packet to transmit), and uses the same value of L_A as in Algorithm 5. The parameters which are subject to optimization are Δ, the maximum allocation interval, and R.

Algorithm 4. Identifying the End of a CRI

loop
  Wait for feedback
  EndOfCRI $\leftarrow$ FALSE
  if $F_T \neq C$ and $F_{T-1} = 1$ **then**
    EndOfCRI $\leftarrow$ TRUE
  if $F_T \neq C$ and last EndOfCRI was TRUE **then**
    EndOfCRI $\leftarrow$ TRUE
  if $F_T = 1$ and $L_A = R$ **then**
    EndOfCRI $\leftarrow$ TRUE

Algorithm 5. The LCFS Algorithm

loop
  $L_A \leftarrow 0$, Class $\leftarrow 1$, $\sigma \leftarrow L$, $T_x \leftarrow 0$
  Wait for a packet to arrive (suppose it arrives at time $T_a \in [T', T'+1)$)
  Wait for feedback of slot T'
  $T \leftarrow T' + 1 - T_a$
  repeat
    if $F_{T'} = 0$ **then**
      if $F_{T'-1} = 0$ **then**
        $L_A \leftarrow L_A + 1$
      else
        $L_A \leftarrow 1$
    else $\{F_{T'} \neq 0\}$
      if $F_{T'} = C$ **then**
        $L_A \leftarrow 0$
      else
        $L_A \leftarrow 1$
    if Class $= 2$ **then**
      if $F_{T'} = C$ **then**
        $\ell \leftarrow \frac{\ell}{2}$
        $\sigma \leftarrow R$
      else if $\sigma = R$ **then**
        if $F_{T'} = 1$ **then**
          $T_x \leftarrow T_x + \ell$
          $\sigma \leftarrow L$
      else $\{F_{T'} = 0\}$
        $T_x \leftarrow T_x + \ell$
        if $L_A < R$ **then**
          $\ell \leftarrow \frac{\ell}{2}$
        $\sigma \leftarrow R$
    else $\{\sigma = L\}$
      $T_x \leftarrow T_x + \ell$
      $T_1 \leftarrow T - \min(L_A, R)$
      $\ell \leftarrow \Delta$
      $\sigma \leftarrow L$
    if Class $= 1$ and a CRI just ended or $L_A > R$ **then**
      Class $\leftarrow 2$
      $\ell \leftarrow \Delta$
      $T_1 \leftarrow T - \min(L_A, R)$
    if Class $= 2$ and $T_1 - T_x \leq \ell$ **then**
      Transmit the packet on the next slot
    Wait for feedback
    $T \leftarrow T + 1$
  until Packet is transmitted successfully

4.2.2. Energy efficiency for the LCFS algorithm

We analyze the performance of the algorithm for $R = 1$, and present simulation results for other values of R. The method used is identical to the one used in Section 3.2.2. Unlike the FCFS algorithm, in the LCFS algorithm gaps can be formed in the examined intervals, thus our definition of the "lag" must change. We denote the "virtual lag" by d, and define it as the sum of all unexamined intervals from $T = 0$ as viewed by an external observer. An examined interval, for that matter, is an interval in which all packets have been resolved.

Using the same notations as in Section 3.2.2, we have that when $R = 1$, all equations used to calculate $\overline{P}$ for the LCFS algorithm are identical to the equations used for the FCFS algorithm (i.e., the bound for H and Y are calculated using the same formulas), but the conditional expectations $\overline{V}_d$, $\overline{L}_d$ and $\overline{\delta}_d$ should be calculated according to the equations in [4, Appendix B]. To calculate $\overline{D}$, using the same method, we use

$$\overline{D} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} D_i = \overline{D_\infty} = \frac{W}{C},$$

where D_i is the delay experienced by the ith packet when the packets are ordered according to the time of arrival, and W is the expected cumulative delay experienced by all the packets in a regeneration cycle. We already have an expression for H, so we can compute C using (13). The derivation of bounds on W for $R = 1$, and thus the bounds on $\overline{D}$, is presented in Appendix B.

After computing all the required quantities, we can bound U_L for $R = 1$, using

$$\frac{\lambda}{\left[\overline{P}^u + \xi\overline{D}^u\right]^\mu} = U_L^l \leqslant U_L \leqslant U_L^u = \frac{\lambda}{\left[\overline{P}^l + \xi\overline{D}^l\right]^\mu}$$

for λ that satisfies (A.13).

4.3. Numerical results and discussion

To verify our analysis of $\overline{P}$ for the FA algorithm and to evaluate the accuracy of using a finite number of terms from the Taylor expansions in λ, we compared the numerical data obtained using the analysis presented in Section 4.1.2 and the results of a simulation of the FA algorithm. We have found that the analysis and the simulation results coincide for all values of p and λ when we use 10 terms of the Taylor expansion in λ, and when comparing to the average of 30 simulation runs of $5 \cdot 10^5$ slots.

Table 3 presents some values of $\overline{P}$ for various values of p, as computed using the analysis in Section 4.1.2. We note that $\overline{P}$ is symmetrical around $p = 0.5$ for any given λ, and gets its minimum value when $p = 0.5$. This is expected, as K_a in (15) is symmetrical around $p = 0.5$ (exchanging p and q does not change the value of K_a), and this is also true for the $\mathbf{S}$ operator, when $f(\cdot)$ in $\mathbf{S}(f(\cdot);u)$ is independent of p (and q). Using MATLAB, we have programmed the algorithms used to evaluate the values of $\mathbf{S}$ for any p and λ, according to the procedures and expressions described in Section 4.1.2 and in [6,11], and in this manner obtained the following expression for the expected number of times a packet is transmitted before it is successfully delivered.

$$\overline{P}(\lambda) = 1 + \frac{1}{2pq}\lambda + \frac{4pq+3}{12p^2q^2}\lambda^2 + O(\lambda^3). \tag{16}$$

From (16), we can easily see that (at least for small values of λ) the expression for $\overline{P}$ is minimized for $p^* = 0.5$, for any given value of λ. This is no longer true for the delay, as described in [6]. For small values of λ, Fayolle et al. found that $\overline{D}$ is minimized at $p^* = 2 - \sqrt{2} \approx 0.586$ (while for larger values of λ, p^* tends to 0.5). This suggests that U_L is not maximized for $p = 0.5$, as the utility function now includes the expected packet delay.

To find the maximum value of U_L for the FA algorithm and the maximizers λ^* and p^* for given ξ and μ, we used standard numerical optimizing techniques, and produced Table 4. As we can see, the optimum value of p is between 0.5 and 0.586, i.e., between the value of p that minimizes $\overline{P}$ and the maximal value that minimizes $\overline{D}$. For $\xi = 0$, we see that $p^* = 0.5$ for all values of μ, which is what we expected, since for this case we do not consider the expected packet delay, and we know that $\overline{P}$ is minimized for $p = 0.5$. For this case, and for all other values of ξ, we see that increasing μ decreases the value of λ^*, as would have been expected—if we care more about $\overline{P}$, we would prefer to sacrifice the throughput to get a higher value of U_L. This, in turn, affects the value of p^*, since a lower value of λ^* means that p^* should be closer to the value that minimizes $\overline{D}$ for small λ. We also see that changing μ has much more impact on the values of p^* and λ^* than changing ξ. Moreover, for a given value of μ, the values of p^* and λ^* do not change by much when we change ξ, while $\xi > 0$. To understand why this is so, we should consider Fig. 8, where we can see that for any given λ we have $\overline{D} > \overline{P}$, and that the difference between them increases as λ increases. For this reason, when $\xi > 0$, the main factor that determines the value of the denominator of U_L is $\overline{D}$, so the values of (λ^*, p^*) are determined mainly due to its magnitude.

We now turn to investigate the performance of the LCFS algorithm. We first compare the bounds found

Table 3
Computed values of $\overline{P}$ for the FA algorithm, for various values of λ and p

λ	$p = 0.2$	$p = 0.3$	$p = 0.4$	$p = 0.5$	$p = 0.55$	$p = 0.6$	$p = 0.7$
.01	1.0325	1.0246	1.0214	1.0205	1.0208	1.0214	1.0246
.03	1.1055	1.0785	1.0680	1.0651	1.0658	1.0680	1.0785
.05	1.1911	1.1397	1.1203	1.1150	1.1162	1.1203	1.1397
.07	1.2920	1.2096	1.1791	1.1708	1.1728	1.1791	1.2096
.09	1.4119	1.2896	1.2455	1.2336	1.2365	1.2455	1.2896
.11	1.5554	1.3817	1.3208	1.3046	1.3085	1.3208	1.3817
.13	1.7287	1.4883	1.4066	1.3850	1.3902	1.4066	1.4883
.15	1.9405	1.6127	1.5049	1.4768	1.4835	1.5049	1.6127
.17	2.2027	1.7588	1.6181	1.5819	1.5905	1.6181	1.7588
.19	2.5328	1.9319	1.7494	1.7031	1.7141	1.7494	1.9319
.21	2.9568	2.1391	1.9029	1.8438	1.8579	1.9029	2.1391
.23	3.5160	2.3904	2.0839	2.0086	2.0264	2.0839	2.3904
.25	4.2790	2.6994	2.2996	2.2033	2.2261	2.2996	2.6994
.27	5.3695	3.0865	2.5597	2.4360	2.4651	2.5597	3.0865
.29	7.0360	3.5823	2.8782	2.7176	2.7553	2.8782	3.5823
.31	9.8595	4.2356	3.2750	3.0640	3.1133	3.2750	4.2356

The computations use 15 terms of the Taylor expansion in λ.

Table 4
Maximum value of U_L for the FA algorithm and the maximizers (λ^*,p^*)

ξ,μ	0.5	1	1.5	2	2.5	3
0	0.1771 (0.313,0.5)	0.1145 (0.2273,0.5)	0.0857 (0.1804,0.5)	0.0688 (0.1505,0.5)	0.0576 (0.1295,0.5)	0.0496 (0.1138,0.5)
0.5	0.1136 (0.2458,0.505)	0.0584 (0.1896,0.5111)	0.0339 (0.1558,0.515)	0.0209 (0.1329,0.5178)	0.0134 (0.1162,0.5199)	0.0088 (0.1034,0.5215)
1	0.0918 (0.237,0.507)	0.0397 (0.1821,0.515)	0.0193 (0.1499,0.5204)	0.01 (0.1283,0.5242)	0.0054 (0.1125,0.5271)	0.003 (0.1004,0.5294)
1.5	0.0792 (0.2331,0.508)	0.0301 (0.1787,0.5171)	0.0129 (0.1472,0.5232)	0.0059 (0.1261,0.5275)	0.0028 (0.1107,0.5309)	0.0013 (0.0989,0.5335)
2	0.0707 (0.2309,0.5086)	0.0242 (0.1767,0.5183)	0.0094 (0.1456,0.5248)	0.0038 (0.1248,0.5296)	0.0016 (0.1097,0.5332)	0.00071 (0.098,0.536)

The computation uses 10 terms of the Taylor expansion in λ.

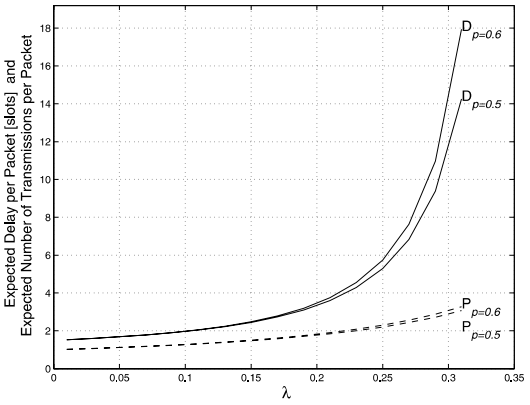

Fig. 8. Comparison between $\overline{D}$ and $\overline{P}$ for the FA algorithm, where 15 terms of the Taylor expansion in λ were used to compute $\overline{P}$ and 12 terms were used to computer $\overline{D}$.

through the analysis presented in Section 4.2.2 with the results of a simulation of the LCFS algorithm with $R = 1$. This comparison is presented in Fig. 9 for $\Delta = 2.58$, when using 20 terms of the infinite series from [4, Appendix B].

Fig. 10 presents U_L for $\Delta = 2.58$ and $R = 1$ using $\overline{P}$ and $\overline{D}$, as presented in Fig. 9, and compares the bound on the utility function as computed analytically, with its value as calculated using the simulation results. We can clearly see that the simulation results fall well within the bounds, and that the bounds are at least tight enough to give a good estimation of the maximizer, λ^*. With this notion in mind, we try to use the analytical bounds to establish an estimation of the optimal values of Δ and λ for the LCFS algorithm with $R = 1$. Table 5 presents the analytical estimation of the maximizers, the estimated maximum value of U_L, together with U_L as calculated using simulation results with the estimated maximizers. These estimations were established using approximated values of $\overline{D}$ and $\overline{P}$, where these approximation are the average between the corresponding upper and lower bounds. We note that the maximum values of U_L as calculated using this approximation are close to the values obtained through simulations. We believe this indicates that our approximated maximizers are close to the real maximizers. Again, we see the effect of μ on the tradeoff between the throughput requirement and the energy efficiency. As we increase μ, the optimum value of λ decreases (for any value of ξ). The value of λ^* is hardly affected by changes in ξ (at least for $\xi \geqslant 0.5$). An interesting outcome of the optimization is the value of Δ^*. We see that for a large set of values of ξ and μ, the optimum value of Δ stays within a rather small range of [2.16, 2.4]. Also, Δ^* increases with ξ and decreases with μ (as it did in the full sensing algorithms).

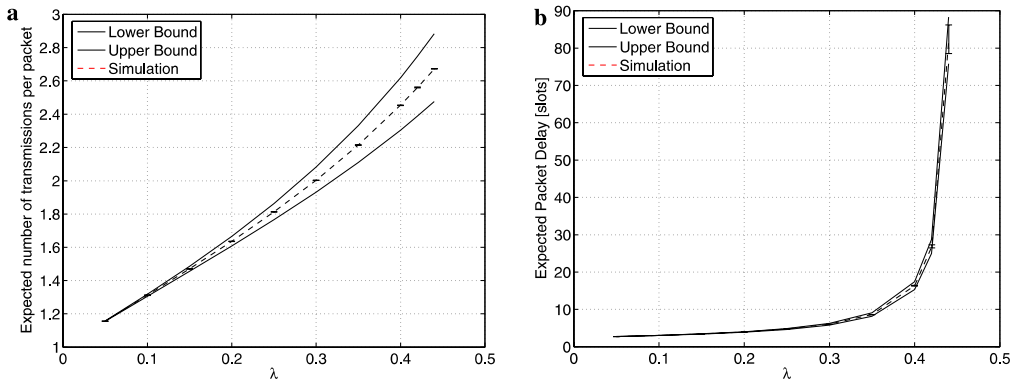

Fig. 9. Comparison between the computed bounds and simulation results of $\overline{P}$ and $\overline{D}$ for the LCFS algorithm with $R = 1$ and $\Delta = 2.58$. The simulation results are averaged over 30 simulation runs of 10^6 slots and the computed bounds use 20 terms of the infinite series for the conditional expectations in [4, Appendix B]. 95% confidence intervals are presented for the simulation samples.

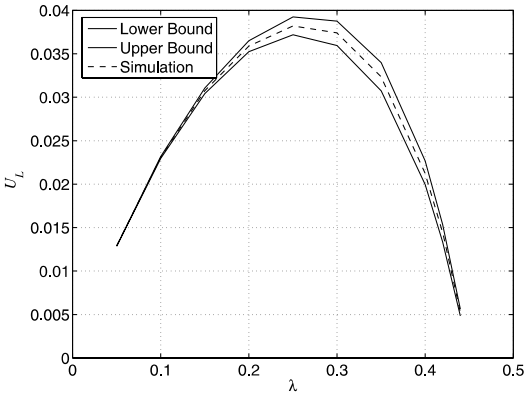

Fig. 10. Comparison between U_L as computed using the simulation data and the analytical bounds as computed using the data presented in Fig. 9, where $\mu = 1$ and $\xi = 1$.

Before we compare the FA algorithm with the LCFS algorithm with respect to the utility function, we would like to investigate how $\overline{P}$ and $\overline{D}$ compare between the two algorithms, as they are the dominant factors in U_L. There are three parameters that we must consider when evaluating the performance of the LCFS algorithm; namely, R, Δ and λ. Fig. 11–14 show the effect of each parameter on $\overline{P}$ and $\overline{D}$. In Fig. 11, we can see that for most values of R, $\overline{P}$ is, more or less, linearly dependent on λ, and that using different values of R did not change the curves by much. We also note that for small values of Δ, the LCFS algorithm outperforms the FA algorithms in terms of the expected number of transmissions per packet. For larger values of Δ (as in Fig. 11(b)), we see that for most values of λ, and for any value of R, we would prefer to use the FA algorithm. This is also evident in Fig. 13, where we can notice that bigger Δ increases the expected number of transmissions per packet, regardless of the R used or the throughput of the system.

In Fig. 12, we can see how the expected packet delay is influenced by λ and R. For small values of λ, where collisions are rare, increasing R increases the expected packet delay, as each new packet spends $R+1$ slots before it can determine that, indeed, there is no CRI in progress. In this range of λ, we notice that for all R and Δ, the FA outperforms the LCFS algorithm. For λ that approaches the capacity of the algorithm (namely 0.487), we see that taking a bigger value of R might produce better delay characteristics. We should also bear in mind that for a stable behavior of the algorithm for large values of λ, we cannot choose the FA algorithm, as its maximum stable throughput is ≈0.36. Fig. 14 displays the effect of changes in Δ on the expected packet delay. While the conclusion from

Table 5
Maximum value of U_L and the maximizers (λ^*,Δ^*) for the LCFS algorithm with $R = 1$ using approximations and the maximum values obtained through simulation using (λ^*,Δ^*)

ξ,μ	0.5	1	1.5	2	2.5	3
0.5	0.1381	0.0612	0.0301	0.0157	0.0086	0.0048
	(0.3383,2.2401)	(0.2752,2.1852)	(0.2282,2.1772)	(0.1919,2.1750)	(0.1638,2.1642)	(0.1419,2.1630)
	0.1413	0.0649	0.0332	0.0182	0.01053	0.00624
1	0.1078	0.0386	0.0154	0.0066	0.0029	0.0013
	(0.3286,2.3522)	(0.2630,2.2331)	(0.2176,2.2233)	(0.1840,2.2203)	(0.1585,2.2184)	(0.1387,2.2150)
	0.1106	0.0418	0.0177	0.0080	0.00377	0.00185
1.5	0.0915	0.0283	0.0098	0.0036	0.0014	5.3613e-004
	(0.3243,2.3615)	(0.2583,2.3450)	(0.2136,2.3420)	(0.1811,2.3408)	(0.1564,2.2301)	(0.1375,2.2287)
	0.0946	0.0309	0.0113	0.0045	0.00184	0.00078
2	0.0809	0.0223	0.0069	0.0023	7.6741e-004	2.6850e-004
	(0.3220,2.3858)	(0.2556,2.3535)	(0.2114,2.3517)	(0.1795,2.3452)	(0.1556,2.3439)	(0.1370,2.3417)
	0.0836	0.0245	0.0081	0.0028	0.00105	0.0004

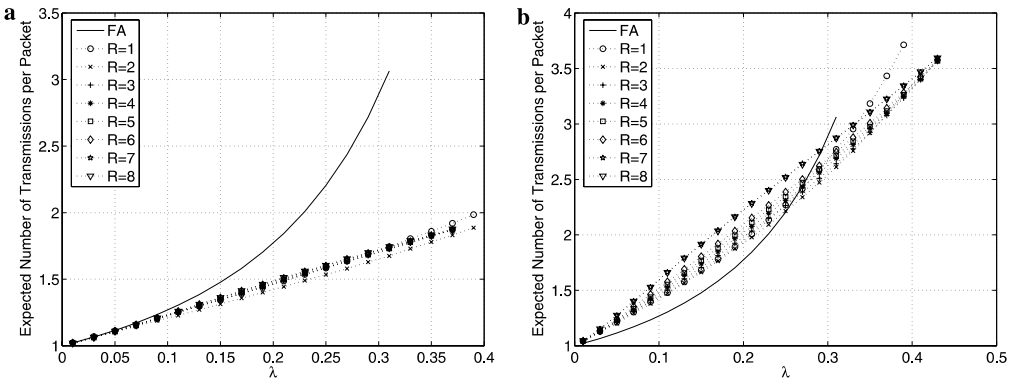

Fig. 11. Expected number of transmissions per packet for the LCFS algorithm, where $\Delta = 1.5$ (a) and $\Delta = 6$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots. The solid curve is of $\overline{P}$ for the FA algorithm with $p = 0.5$.

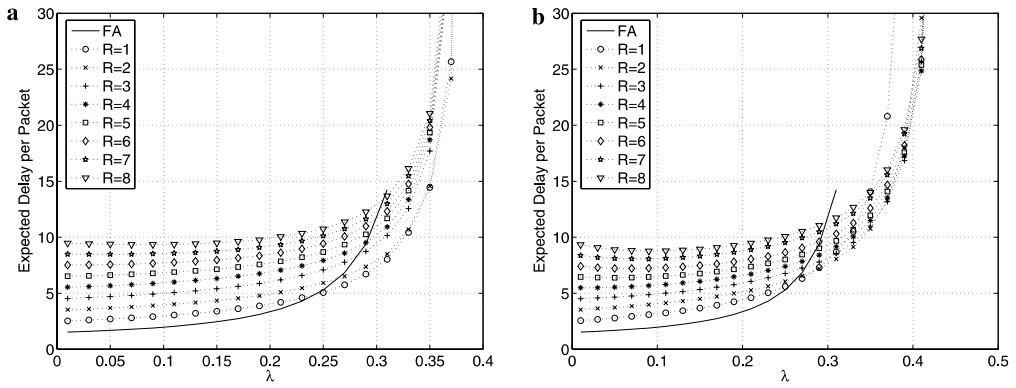

Fig. 12. Expected delay per packet for the LCFS algorithm, where $\Delta = 1.5$ (a) and $\Delta = 6$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots. The solid curve is of $\overline{D}$ for the FA algorithm with $p = 0.5$.

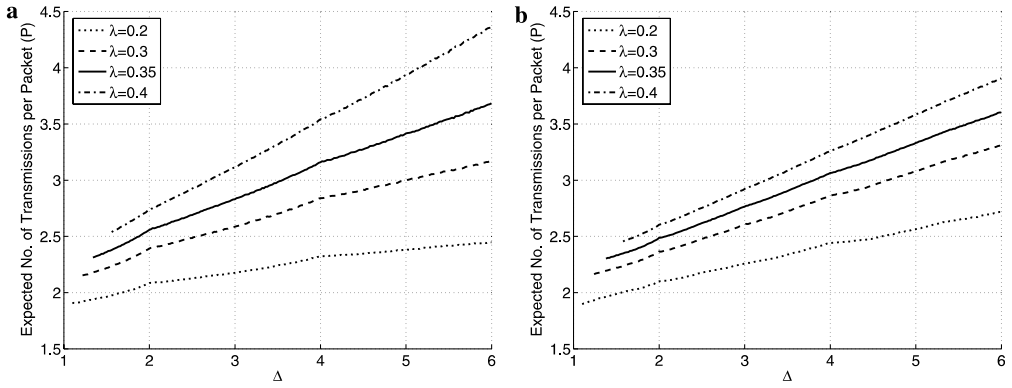

Fig. 13. Expected number of transmissions per packet for the LCFS algorithm, where $R = 1$ (a) and $R = 8$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots.

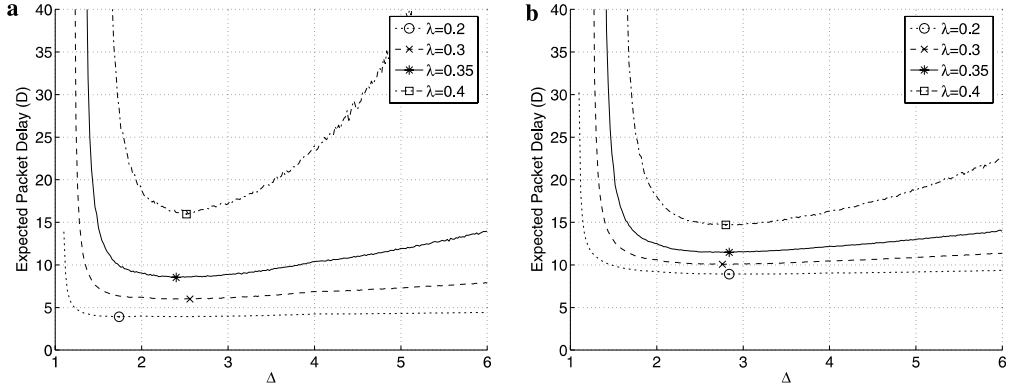

Fig. 14. Expected delay per packet for the LCFS algorithm, where $R = 1$ (a) and $R = 8$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots. The minimum values are marked with a symbol.

Fig. 13 is to use the smallest possible Δ, we can clearly see from Fig. 14 that there is an optimum value for Δ, if we want to minimize the expected packet delay. However, this optimum value is only slightly affected by changes of λ and R, and stays in the 2.5–2.8 range for most values of R and λ. This, and the fact that for any λ, $\overline{P}$ increases with Δ, ensures that the optimum value of Δ will not be greater than 2.8. On the other hand, the severe effect a decrease in Δ has on the expected packet delay is a guarantee that the optimum value of Δ will not be very small, so the optimum for all R and λ is centered in a rather small region.

If we compare the maximum values of U_L between the LCFS algorithm with $R = 1$, as they appear in Table 5 and the FA algorithm (from Table 4), we see that for $\mu \leqslant 1$, we get that the maximum value of U_L is higher when the LCFS algorithm is used, while for $\mu > 1$, U_L^* is higher for the FA algorithm. We also note that the optimum for the FA algorithm is achieved for lower values of λ.

To sum it all up, when the energy invested in delivering a packet is more costly, one would prefer to use the FA algorithm, thus sacrificing the throughput; if the energy is still important, but not as much, the LCFS should be used. Looking at other values of R, we can see in Fig. 15 that for $\mu = 1$ and $\xi = 1$, and for small values of λ, FA displays better energy efficiency than LCFS. If λ is known and set, and we must set the other parameters according to its value, we should first consider which algorithm is stable with the given λ. If both FA and LCFS are stable, we can use Fig. 15 to find out which algorithm to use, and how to set R, if LCFS performs better. We also deduce from Figs. 15, 17 and 18 that, most likely, the best choice for LCFS is either $R = 1$ or $R = 2$. Other values of R should probably be chosen only if the throughput demand cannot be met with these values of R. Figs. 17 and 18 display the advantage of the LCFS for small values of μ and the superiority of the FA for large values of μ, which extends our previous conclusion from the case with $R = 1$ to other values of R.

Fig. 16 is important since it depicts the behavior of U_L when Δ is changed around its optimal value. We can see that if we set a slightly higher Δ than the optimum, the energy efficiency is only slightly degraded, while we gain near-optimality for a broader range of λ and R. (the same can be noticed, to some extent, in Fig. 14.).

We note that for the LCFS algorithm, we can look at the energy-efficiency as it is defined in previous papers, i.e., $\overline{P} + \xi\overline{D}$, and try to minimize this quantity (without regard to the throughput of the system whatsoever). Notice that this is equivalent to maximizing U_L with $\mu \to \infty$, so our utility function, with the proper choice of parameter values, covers previous notions of energy-efficiency. It is interesting to observe that for the LCFS algorithm, when we take $\mu \to \infty$, the optimum value for Δ does not tend to zero, as it does in all other window-access-based algorithms discussed in this work.

5. Conclusions and future work

In this work, we address the energy efficiency of CRPs with infinite population, and propose a utility function to measure this efficiency that also incorporates the throughput requirements of the system. We analyzed three known full sensing CRPs, and found the maximum efficiency for each (as measured by the utility function) and the parameters used to achieve this maximum. We also found that the most efficient full sensing algorithm one could choose, out of the algorithms we examined, is the FCFS algorithm, also known as the 0.487 throughput algorithm. We concluded the discussion of full sensing algorithms with an outline

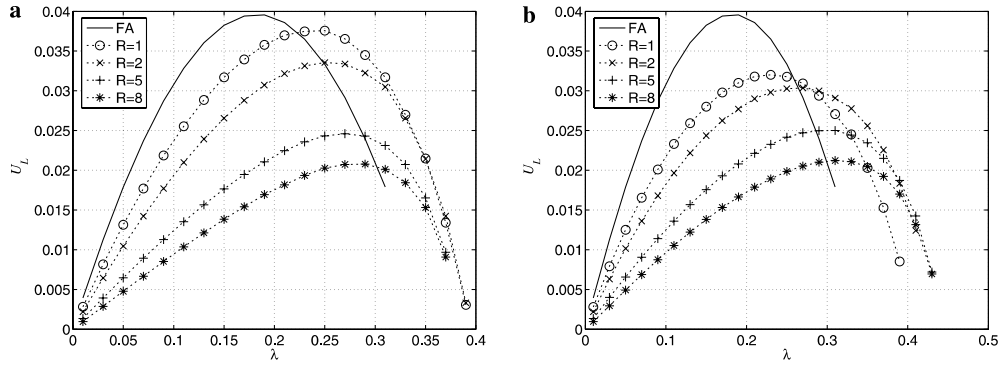

Fig. 15. U_L with $\xi = 1$ and $\mu = 1$ for the LCFS algorithm, where $\Delta = 1.5$ (a) and $\Delta = 6$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots. The solid curve is of U_L for the FA algorithm with $p = 0.5$.

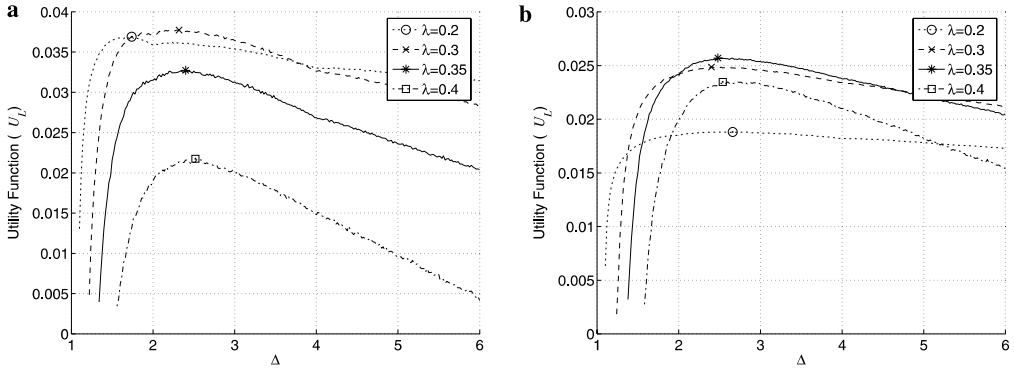

Fig. 16. U_L with $\xi = 1$ and $\mu = 1$ for the LCFS algorithm, where $R = 1$ (a) and $R = 8$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots. The maximum values are marked with a symbol.

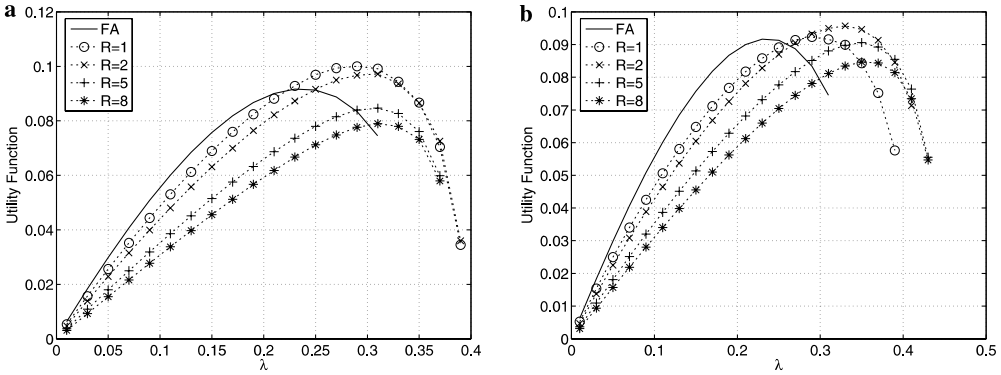

Fig. 17. U_L with $\xi = 1$ and $\mu = 0.5$ for the LCFS algorithm, where $\Delta = 1.5$ (a) and $\Delta = 6$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots.

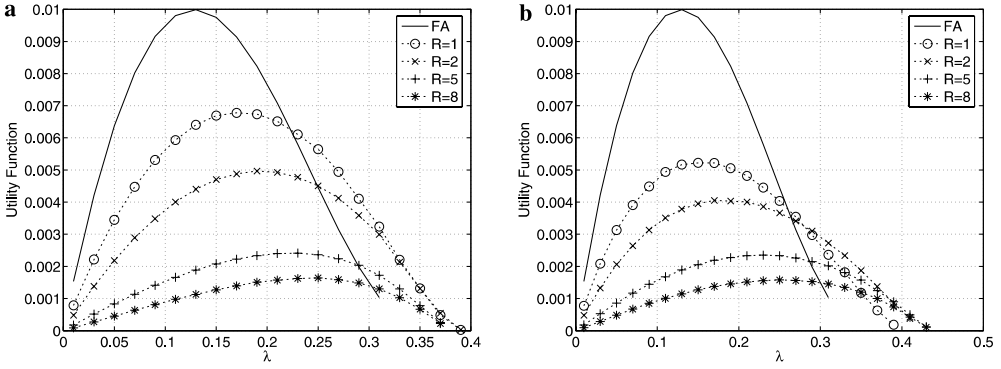

Fig. 18. U_L with $\xi = 1$ and $\mu = 2$ for the LCFS algorithm, where $\Delta = 1.5$ (a) and $\Delta = 6$ (b). All LCFS results are averaged over 30 simulation runs of $5 \cdot 10^5$ slots.

of a modification to this algorithm that might improve the energy efficiency for some scenarios. In the second part of the paper, we consider the energy efficiency of limited sensing algorithms, where a node has to listen to the feedback only when it has a packet to transmit. For these algorithms, we can compute the energy needed to "tune in" on the feedback signals, and combine this quantity with the transmission energy, to obtain the total energy needed to deliver a packet successfully and incorporate it into the utility function. We analyzed two such algorithms—the free access algorithm and the last-come-first-served algorithm, and found that if all parameters can be manipulated, the preferred algorithm depends on the importance placed on the energy compared to the system throughput. If the energy consumption is more important, the FA algorithm displays better properties, while the LCFS algorithm gets higher marks for scenarios in which the high steady-state throughput is more appreciated. If the required throughput is set, one can use our analysis to determine what is the best limited sensing algorithm to use, and how its parameters should be tuned. We further facilitate the choice of parameters and the optimum algorithm by observing that for most scenarios, only $R = 1$ and $R = 2$ should be considered for the LCFS algorithm.

For further study, we note the suggested improvement to the FCFS algorithm that hasn't been thoroughly investigated. Also, the finite population model could present options for optimizations and algorithms that could possibly achieve better energy efficiency, since we have more information in this model (e.g., the number of contending packets is known to be bounded).

Appendix A. FCFS energy efficiency calculations

Here, we develop a system of equations to find Y and upper and lower bounds on its solution. We use the following notations for the analysis:

L,	length of the CRI;
δ,	length of the resolved interval;
ℓ,	length of the allocation interval;
d,	the lag of the algorithm;
V,	the cumulative number of transmissions in a CRI.

We first define the random variable y_d as the cumulative number of transmissions when we count starting from the beginning of a CRI with lag d, until we reach a CRI with $d = 1$. From the operation of the algorithm, we have

$$y_d = \begin{cases} V, & L = 1; \\ V + y_{d-\delta+L}, & L > 1; \end{cases} \quad \forall 1 \leqslant d \leqslant \Delta \tag{A.1}$$

$$y_d = V + y_{d-\delta+L}, \quad \forall d > \Delta, \tag{A.2}$$

where $d \in \mathscr{F}$, $\mathscr{F}$ being the set of all possible values d can take, which was found to be a denumerable dense subset of the interval [1,∞) in [12]. We denote $Y_d = \overline{y_d}$. Taking the expectation in (A.1), and denoting $\overline{V}_x$ as the expected number of transmissions in a CRI, given that at the beginning of the CRI $\ell = x$, we get

$$Y_d = \begin{cases} \overline{V}_d + \sum_{\substack{r,s \\ s\neq 1}} \Pr(r,s|d) Y_{d-r+s}, & 1 \leqslant d \leqslant \Delta; \\ \overline{V}_\Delta + \sum_{r,s} \Pr(r,s|\Delta) Y_{d-r+s}, & d > \Delta, \end{cases} \tag{A.3}$$

where $\Pr(r,s|x)$ is the joint conditional probability distribution of δ and L, at $\delta = r$ and $L = s$, given that $\ell = x$. We note that, by definition, $Y = Y_1$, so if we had the solution for (A.3), we could easily find Y. Using the method suggested in [9], we develop lower and upper bounds for the solution of (A.3) by looking for sequences $\{\mathscr{L}_d^0, d \in \mathscr{F}\}$ and $\{\mathscr{U}_d^0,\ d \in \mathscr{F}\}$ that will satisfy the following inequalities:

$$\mathscr{U}_d^0 \geqslant b_d + \sum_{j\in\mathscr{F}} p_{dj} \mathscr{U}_j^0 \geqslant 0, \quad \forall d \in \mathscr{F}; \tag{A.4}$$

$$\mathscr{L}_d^0 \leqslant \mathscr{U}_d^0, \quad \forall d \in \mathscr{F}, \tag{A.5}$$

$$\mathscr{L}_d^0 \leqslant b_d + \sum_{j\in\mathscr{F}} p_{dj} \mathscr{U}_j^0, \quad \forall d \in \mathscr{F}; \tag{A.6}$$

where

$$b_d = \begin{cases} \overline{V}_d, & 1 \leqslant d \leqslant \Delta; \\ \overline{V}_\Delta, & d > \Delta. \end{cases} \quad p_{dj} = \begin{cases} \sum_{s\neq 1} \Pr(d+s-j, s|d), & 1 \leqslant d \leqslant \Delta; \\ \sum_{s} \Pr(d+s-j, s|\Delta), & d > \Delta. \end{cases} \tag{A.7}$$

Once we find these sequences, [9] assures us that a finite solution to (A.3) exists and that $\{\mathscr{U}_d^0, d \in \mathscr{F}\}$ is an upper bound to this solution, while $\{\mathscr{L}_d^0, d \in \mathscr{F}\}$ is its lower bound. Moreover, if we generate the sequences

$$\mathscr{U}_d^1 = b_d + \sum_{j\in\mathscr{F}} p_{dj} \mathscr{U}_j^0, \quad \forall d \in \mathscr{F}; \tag{A.8}$$

$$\mathscr{L}_d^1 = b_d + \sum_{j\in\mathscr{F}} p_{dj} \mathscr{L}_j^0, \quad \forall d \in \mathscr{F} \tag{A.9}$$

then $\{\mathscr{U}_d^1, d \in \mathscr{F}\}$ and $\{\mathscr{L}_d^1, d \in \mathscr{F}\}$, are tighter bounds on the solution to (A.3).

Let $\mathscr{U}_d^0 = c_1^y d + c_2^y, d \in \mathscr{F}$, where c_1^y and c_2^y are real constants, and define

$$\mathscr{U}_d^1 = b_d + \sum_{j\in\mathscr{F}} p_{dj} \mathscr{U}_j^0, \quad \forall d \in \mathscr{F}. \tag{A.10}$$

Substituting (A.7) into (A.10), and noting that

$$\Pr(r,1|x) = \begin{cases} (1+\lambda d)e^{-\lambda d}, & \text{if } r = x; \\ 0, & \text{otherwise}, \end{cases} \tag{A.11}$$

yields

$$\mathscr{U}_d^1 = \mathscr{U}_d^0 + \begin{cases} \overline{V}_d + c_1^y(\overline{L}_d - \overline{\delta}_d - (1+\lambda d)e^{-\lambda d}) - c_2^y(1+\lambda d)e^{-\lambda d}), & 1 \leqslant d \leqslant \Delta; \\ \overline{V}_\Delta - c_1^y(\overline{\delta}_\Delta - \overline{L}_\Delta), & d > \Delta. \end{cases} \tag{A.12}$$

The conditional expectancies in (A.12) can be computed according to the formulas in [8,9] and in (10). It is known [7] that the algorithm is stable if

$$\overline{\delta}_\Delta > \overline{L}_\Delta. \tag{A.13}$$

We note that when (A.13) holds, and c_1^y, c_2^y are chosen as

$$c_1^y = \frac{\overline{V}_\Delta}{\overline{\delta}_\Delta - \overline{L}_\Delta}, \tag{A.14}$$

$$c_2^y = \max\left\{-c_1^y, \sup_{1\leqslant d\leqslant\Delta}(\rho(d))\right\}, \tag{A.15}$$

where

$$\rho(d) = \frac{\overline{V}_d + c_1^y(\overline{L}_d - \overline{\delta}_d - (1+\lambda d)e^{-\lambda d})}{(1+\lambda d)e^{-\lambda d}} \tag{A.16}$$

we get that inequality (A.4) is satisfied and $\mathscr{U}_d^1 \leqslant \mathscr{U}_d^0, \forall d \in \mathscr{F}$. Similarly, it can be shown that if we define $\mathscr{L}_d^0 = c_3^y d + c_4^y, d \in \mathscr{F}$, inequalities (A.5) and (A.6) can be satisfied if we choose

$$c_3^y = c_1^y, \qquad c_4^y = \inf_{1\leqslant d\leqslant\Delta}(\rho(d)), \tag{A.17}$$

where c_1^y and $\rho(d)$ are as given in (A.14) and in (A.16). Collecting it all, we have that

$$\mathscr{L}_d^1 \leqslant Y_d \leqslant \mathscr{U}_d^1, \quad \forall d \in \mathscr{F} \tag{A.18}$$

or more specifically, $Y^l \leqslant Y \leqslant Y^u$, where we define $Y^u \triangleq \mathscr{U}_1^1$ and $Y^l \triangleq \mathscr{L}_1^1$, and where

$$Y^u = \overline{V}_1 + c_1^y[1 + \overline{L}_1 - \overline{\delta}_1 - (1+\lambda)e^{-\lambda}] + c_2^y[1 - (1+\lambda)e^{-\lambda}]; \tag{A.19}$$

$$Y^l = Y^u - (c_2^y - c_4^y)[1 - (1+\lambda)e^{-\lambda}]. \tag{A.20}$$

Using the same method, it was found in [8,9] that $H^l \leqslant H \leqslant H^u$, where

$$H^u = \overline{L}_1 + c_1^h[1 + \overline{L}_1 - \overline{\delta}_1 - (1+\lambda)e^{-\lambda}] + c_2^h[1 - (1+\lambda)e^{-\lambda}]; \tag{A.21}$$

$$H^l = H^u - (c_2^h - c_4^h)[1 - (1+\lambda)e^{-\lambda}]; \tag{A.22}$$

$$c_1^h = c_3^h = \frac{\overline{L}_\Delta}{\overline{\delta}_\Delta - \overline{L}_\Delta};$$

$$c_2^h = \max\left\{-c_1^h, \sup_{1\leqslant d\leqslant\Delta}(\rho'(d))\right\};\ c_4^h = \inf_{1\leqslant d\leqslant\Delta}(\rho'(d));$$

$$\rho'(d) = \frac{\overline{L}_d + c_1^y(\overline{L}_d - \overline{\delta}_d - (1+\lambda d)e^{-\lambda d})}{(1+\lambda d)e^{-\lambda d}}.$$

Appendix B. LCFS energy efficiency calculations

To find W, we follow [8] and define the random variable w_d as the cumulative delay of all packets when we count starting from the beginning of a CRI with virtual lag d, until we reach a CRI with $d = 1$. From the operation of the algorithm, we have

$$w_d = \begin{cases} \psi + \omega + N, & L = 1; \\ \psi + \omega + N + \delta N^0 + w_{d-\delta+L}, & L > 1. \end{cases} \quad \forall 1 \leqslant d \leqslant \Delta \tag{B.1}$$

$$w_d = \psi + \omega + N + \delta N^d + w_{d-\delta+L};\ \forall d > \Delta, \tag{B.2}$$

Table B.1
Notations used in the analysis of the LCFS algorithm

Notation	Meaning
δ	Length of the resolved interval,
N	Number of packets transmitted in a CRI (i.e, the number of packets in δ),
N^0	Number of packets that where included in the beginning of the CRI, but were not resolved (i.e., the number of packets in $\ell-\delta$),
N^d	Number of packets that are unresolved and are not included in the current CRI,
L	Length of a CRI,
ω	Cumulative delay of all N packets from the beginning of the CRI to the time a packet is transmitted,
ψ	Cumulative delay of all N packets from their arrival until the beginning of the next CRI.

where the notations used in the last two equations are explained in Table B.1. Denoting $W_d=\overline{w_d}$ and taking the expectation in (B.1) and (B.2) yields

$$W_d=\begin{cases}\overline{\psi}_d+\overline{\omega}_d+\overline{N}_d+\overline{\delta N^0}_d+\sum\limits_{\substack{r,s\\ s\neq 1}}\Pr(r,s|d)W_{d-r+s}, & 1\leqslant d\leqslant\Delta;\\ \overline{\psi}_\Delta+\overline{\omega}_\Delta+\overline{N}_\Delta+\overline{\delta N^0}_\Delta+\overline{\delta N^d}_\Delta+\sum\limits_{r,s}\Pr(r,s|\Delta)W_{d-r+s}, & d>\Delta.\end{cases}\tag{B.3}$$

Substituting $\overline{N^d\delta}_\Delta=\lambda(d-\Delta)\overline{\delta}_\Delta$ in (B.3) gives

$$W_d=\begin{cases}\overline{\psi}_d+\overline{\omega}_d+\overline{N}_d+\overline{\delta N^0}_d+\sum\limits_{\substack{r,s\\ s\neq 1}}\Pr(r,s|d)W_{d-r+s}, & 1\leqslant d\leqslant\Delta;\\ \overline{\psi}_\Delta+\overline{\omega}_\Delta+\overline{N}_\Delta+\overline{\delta N^0}_\Delta+\lambda(d-\Delta)\overline{\delta}_\Delta+\sum\limits_{r,s}\Pr(r,s|\Delta)W_{d-r+s}, & d>\Delta.\end{cases}\tag{B.4}$$

Let $\mathscr{U}_d^0=c_1^wd^2+c_2^wd+c_3^w, d\in\mathscr{F}$, where c_1^w, c_2^w, and c_3^w are real constants. Following a similar procedure to the one taken in Section 3.2.2, and after some algebra, we get for $d>\Delta$,

$$\begin{aligned}\mathscr{U}_d^1=\mathscr{U}_d^0&+\overline{\psi}_\Delta+\overline{\omega}_\Delta+\overline{N}_\Delta-\lambda\Delta\overline{\delta}_\Delta+\overline{\delta N^0}_\Delta\\&+d\left[2c_1^w(\overline{L}_\Delta-\overline{\delta}_\Delta)+\lambda\overline{\delta}_\Delta\right]+c_1^w\overline{(L-\delta)^2}_\Delta+c_2^w\overline{(L-\delta)}_\Delta\end{aligned}$$

and for $1\leqslant d\leqslant\Delta$,

$$\begin{aligned}\mathscr{U}_d^1=\mathscr{U}_d^0&+\overline{\psi}_d+\overline{\omega}_d+\overline{N}_d+\overline{\delta N^0}_d\\&+c_1^w\left[2d(\overline{L}_d-\overline{\delta}_d)+\overline{(L-\delta)^2}_d-(1+\lambda d)e^{-\lambda d}\right]\\&+c_2^w\left[\overline{(L-\delta)}_d-(1+\lambda d)e^{-\lambda d}\right]-c_3^w(1+\lambda d)e^{-\lambda d}.\end{aligned}$$

For this algorithm, stability is also maintained if (A.13) is satisfied. When the algorithm is stable, if we choose

$$c_1^w=\frac{\lambda\overline{\delta}_\Delta}{2\left(\overline{\delta}_\Delta-\overline{L}_\Delta\right)};\tag{B.5}$$

$$c_2^w=\frac{\overline{\psi}_\Delta+\overline{\omega}_\Delta+\overline{N}_\Delta-\lambda\Delta\overline{\delta}_\Delta+\overline{\delta N^0}_\Delta+c_1^w\overline{(L-\delta)^2}_\Delta}{\overline{\delta}_\Delta-\overline{L}_\Delta};\tag{B.6}$$

$$c_3^w=\sup_{1\leqslant d\leqslant\Delta}(\rho(d)),$$

where

$$\begin{aligned}\rho(d)=\frac{1}{(1+\lambda d)e^{-\lambda d}}&\left\{\overline{\psi}_d+\overline{\omega}_d+\overline{N}_d+\overline{\delta N^0}_d\right.\\&+c_1^w\left[\overline{(L-\delta)^2}_d-2d\overline{(\delta-L)}_d-(1+\lambda d)e^{-\lambda d}\right]\\&\left.+c_2^w\left[\overline{(L-\delta)}_d-(1+\lambda d)e^{-\lambda d}\right]\right\}\end{aligned}\tag{B.7}$$

we get that (A.4) holds and $\mathscr{U}_d^1\leqslant\mathscr{U}_d^0,\forall d\in\mathscr{F}$. Similarly, it can be shown that if we define $\mathscr{L}_d^0=c_4^wd^2+c_5^wd+c_6^w, d\in\mathscr{F}$, inequalities (A.5) and (A.6) can be satisfied if we choose

$$c_4^w=c_1^w,\qquad c_5^w=c_2^w,\qquad c_6^w=\inf_{1\leqslant d\leqslant\Delta}(\rho(d)),\tag{B.8}$$

where c_1^w, c_2^w and $\rho(d)$ are as given in (B.5), (B.6) and in (B.7). Thus we can write

$$\frac{W^l}{\lambda H^u}=\overline{D}^l\leqslant\overline{D}\leqslant\overline{D}^u=\frac{W^u}{\lambda H^l}\tag{B.9}$$

where H^l and H^u are given in (A.21) and (A.22), and with

$$\begin{aligned}W^u=&\overline{\psi}_1+\overline{\omega}_1+\overline{N}_1+\overline{\delta N^0}_1+c_1^w\left[1+\overline{(L-\delta)^2}+2(\overline{L}_1-\overline{\delta}_1)-(1+\lambda)e^{-\lambda}\right]\\&+c_2^w\left[1+\overline{(L-\delta)}_1-(1+\lambda)e^{-\lambda}\right]+c_3^w\left[1-(1+\lambda)e^{-\lambda}\right];\\W^l=&W^u-(c_3^w-c_6^w)\left[1-(1+\lambda)e^{-\lambda}\right].\end{aligned}\tag{B.10}$$

References

[1] ISO/IEC 8802-11; ANSI/IEEE Std 802.11, 1999 edn, 1999.
[2] IEEE Std 802.11a-1999, 1999.
[3] IEEE Std 802.11b-1999, 2000.
[4] A. Bergman, Energy efficiency of collision resolution protocols. Master's thesis, Technion—Israel Institute of Technology, May 2005.
[5] J.I. Capetanakis, Tree algorithms for packet broadcast channels, IEEE Transactions on Information Theory 25 (5) (September 1979) 505–515.
[6] G. Fayolle, P. Flajolet, M. Hofri, P. Jacquet, Analysis of a stack algorithm for random multiple-access communication, IEEE Transactions on Information Theory 31 (2) (March 1985) 244–254.
[7] R.G. Gallager, Conflict resolution in random access broadcast networks, in: Proceedings AFOSR Workshop in Comm. Theory and Applications, Provincetown, MA, September 1978, pp. 74–76.
[8] L. Georgiadis, Limited Sensing Random Access Algorithms and Unified Methods for their Analysis, PhD thesis, The University of Connecticut, 1986.
[9] L. Georgiadis, L.F. Merakos, P. Papantoni-Kazakos, A method for the delay analysis of random multiple-access algorithms whose delay process is regenerative, IEEE Journal on Selected Areas In Communications 5 (6) (July 1987) 1051–1062.
[10] L. Georgiadis, P. Papantoni-Kazakos, A 0.487 throughput limited sensing algorithm, IEEE Transactions on Information Theory 33 (2) (March 1987) 233–237.

[11] M. Hofri, Analysis of Algorithms: Computational Methods and Mathematical Tools, Oxford University Press, Oxford, New York, 1995.
[12] J.C. Huang, T. Berger, Delay analysis of 0.487 contention resolution algorithms, IEEE Transactions on Communications 34 (9) (September 1986) 916–926.
[13] P.A. Humblet, On the throughput of channel access algorithms with limited sensing, IEEE Transactions on Communications 34 (4) (April 1986) 345–347.
[14] S. Khanna, S. Sarkar, I. Shin, An energy management based collision resolution protocol, in: Proceedings of International Teletraffic Congress (ITC18), Berlin, Germany, September 2003, pp. 951–960.
[15] J. Massey, Collision-resolution algorithms and random-access communications, in: G. Longo (Ed.), Multi-User Communication Systems, CISM Courses and Lectures No. 265, Springer, 1981, pp. 73–137.
[16] J. Mosely, P.A. Humblet, A class of efficient contention resolution algorithms for multiple access channels, IEEE Transactions on Communications 33 (2) (February 1985) 145–151.
[17] R. Rom, M. Sidi, Multiple Access Protocols—Performance and analysis, Springer, 1990, chapter 5, pp. 107–148.
[18] Y.E. Sagduyu, A. Ephremides, Energy-efficient collision resolution in wireless ad-hoc networks, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, April 2003, pp. 492–502.
[19] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, June 2002, pp. 1567–1576.

Aran Bergman is a system engineer at Intel (Envara) Israel, currently engaged in developing WiMAX chipsets. His research interests include design optimization and performance evaluation of computer networks and wireless medium access protocols. He received his B.Sc. (1996) and M.Sc. (2005) in Electrical Engineering from the Technion—Israel Institute of Technology.

Moshe Sidi received the B.Sc., M.Sc., and the D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, Israel, in 1975, 1979, and 1982, respectively, all in electrical engineering. In 1982, he joined the faculty of Electrical Engineering Department at the Technion where he is currently a Chaired Professor and Dean. During the academic year 1983–1984 he was a Post-Doctoral Associate at MIT. During 1986–1987 he was a visiting scientist at IBM, Thomas J. Watson Research Center. He coauthors the book "Multiple Access Protocols: Performance and Analysis", Springer Verlag 1990. He served as the Editor for Communication Networks in the IEEE Trans. on Communications from 1989 until 1993, as the Associate Editor for Communication Networks and Computer Networks in the IEEE Trans. on Information Theory from 1991 until 1994, as a founding Editor in the IEEE/ACM Trans. on Networking from 1993 until 1997, and as an Editor in the Wireless Journal 1993–2001. He also served as the General Chair for Infocom 2000.