Adaptive Batch Resolution Algorithm with Deferred Feedback for Wireless Systems

Andrea Zanella

***Abstract*—A batch resolution algorithm (BRA) is a channel access policy used by a group of nodes (the *batch*) that *simultaneously* generate a packet for a common receiver. The aim is to minimize the batch resolution interval (BRI), i.e., the time it takes for all nodes in the batch to successfully deliver their packet. Most of existing BRAs require immediate feedback after each packet transmission, and typically assume the feedback time is negligible. This conjecture, however, fails to apply in practical high rate wireless systems, so that the classical performance analysis of BRAs may be overoptimistic. In this paper we propose and analyze a novel BRA named Adaptive Batch Resolution Algorithm with Deferred Feedback (*ABRADE*), which waives the immediate feedback approach in favor of a deferred feedback method, based on a framed ALOHA access scheme. The frame length is optimized by using a dynamic programming technique in order to minimize the BRI, under the assumption that the batch size is known. Successively, we remove this assumption by coupling ABRADE with a batch size estimate module. The new algorithm, called *ABRADE+*, is compared against the best performing BRAs based on the immediate feedback paradigm, showing better performance both in case of partial and no prior knowledge of the batch multiplicity.**

***Index Terms*—Batch resolution, conflict resolution, collision resolution, framed Aloha, RFID, RF-TAG, node counting, multiplicity estimate, 6LowPAN neighbor discovery, multicast ACK collision problem.**

I. Introduction

The batch resolution problem, also known in the literature as collision or conflict resolution problem, arises whenever a group of nodes, which form the *batch* or conflict set,[1] are *simultaneously* requested to send a packet to a common receiver, using a shared channel. The simultaneous transmission of two or more packets will likely produce the loss of all transmitted packets because of the mutual interference generated by the overlapping signals at the receiver, an event that is typically referred to as *packet collision*.

This problem emerges in wireless networks with densely populated clouds of terminals, where certain events may solicit the simultaneous transmission from a bunch of nodes. For instance, in wireless sensor network deployments for environmental-monitoring or surveillance, the occurrence of atypical events as fires, gas leakages, or intrusions, may trigger the simultaneous transmission of alert messages from multiple wireless sensor nodes. Moreover, when the control node (sink) broadcasts an inquiry message for data reports, all the active sensor nodes in the surrounding are simultaneously requested to reply. Similarly, multiple nodes in a wireless mobile ad-hoc network, or in a 6LowPAN wireless sensor network (see RFC 4944), are required to send `HELLO` messages in response to a neighbor-discovery message broadcasted by a new node that just entered the network. Also, in wireless multicast transmissions, ACK/NAK packets returned by nodes may yield a batch resolution problem.

A batch resolution algorithm (BRA) is a scheme that arbitrates the transmissions of the nodes in the batch in order to minimize the *batch resolution interval* (BRI), i.e., the time it takes for all nodes in the batch to successfully deliver their packet to the intended receiver and, hence, be *resolved*.

The batch resolution problem resembles, in some aspects, the medium access control (MAC) problem in that both consist in managing the channel access of a group of nodes. However, MAC protocols generally look at the channel contention as a steady-state phenomenon, where the traffic offered to the medium is a stationary process. Conversely, BRAs address scenarios where contention has a bursty nature [2]: the medium is typically idle, until a particular event solicits packet transmission from a bunch of nodes simultaneously. Furthermore, nodes that successfully deliver their message to the inquirer are resolved and leave the batch, so that the *batch size*, i.e., the number of nodes that still compete to access the channel, progressively decreases over time.

The design and analysis of efficient BRAs have been deeply studied in the literature and their interest has been recently reawaken in relation with the RFID tags [3]. Landmarks in this area are the *splitting-tree* algorithms devised by Hayes [4] and Capetanakis [5], and successively improved by many others, e.g., [6]–[9]. A comprehensive overview of the splitting-tree algorithms can be found in [10], [11].

The execution flow of the classical BRAs is driven by feedback messages returned after each transmission, according to the *immediate feedback* paradigm. Despite of this, the performance analysis of these algorithms usually neglects or underestimates the time duration of the feedback messages, which is included in the slot duration or approximated as the time length of an idle slot [12]. In practical radio systems, however, each packet transmission (included feedbacks) takes a fixed amount of time for basic operations, such as RX/TX switching, signal detection and synchronization, and so on. This time may represent a significant part of the overall packet transmission time, in particular when the transmission rate is large. For instance, in the IEEE 802.11g standard, the feedback time is about 20% of the time to transmit a full data frame

Manuscript received September 1, 2011; revised January 18 and April 3, 2012; accepted June 21, 2012. The associate editor coordinating the review of this paper and approving it for publication was S. Valaee.

A preliminary version of this work, with title "A dynamic framed ALOHA scheme for batch resolution in practical CSMA-based wireless networks," appeared in the Proceedings of the IEEE Globeom2010, 6-10 December 2010, Miami, Florida, USA.

The author is with the Dept. of Information Engineering, University of Padova, Italy (e-mail: zanella@dei.unipd.it).

Digital Object Identifier 10.1109/TWC.2012.081312.111641

[1]The two terms will be used interchangeably throughout the paper.

at the maximum transmission rate. Therefore, the time cost of feedbacks may have non-negligible impact on the performance of BRAs.

Motivated by this observation, in this paper we design and evaluate a BRA named *Adaptive Batch Resolution Algorithm with Deferred Feedback* (ABRADE) that waives the immediate feedback paradigm in favor of a framed ALOHA contention scheme with deferred feedback, in order to reduce the impact of feedback messages in the BRI. ABRADE works in successive resolution rounds. Each round consists of a *contention frame* of a certain number of slots, followed by a *single aggregate feedback message*, called *probe*. The core of ABRADE is the algorithm used to dynamically adjust the frame length to the number of still unresolved nodes after each round, in order to minimize the BRI. This optimization problem is resolved by applying a dynamic programming argument, under the assumption of perfect knowledge of the residual batch size at each step of the algorithm.

Successively, we relax this assumption and propose ABRADE$^+$, a complete resolution scheme for batches of unknown size obtained by combining ABRADE with a suitable batch size estimate module. We then compare ABRADE$^+$ with the best performing BRAs based on the immediate feedback paradigm proposed in the literature, namely FCFS [6], [7], IECR and Sift/IECR [8], [9], in some practical scenarios. Results confirm that, in practical CSMA-based wireless networks, ABRADE$^+$ outperforms the existing solutions, irrespective of the statistical distribution of the initial batch size.

Summing up, the original contributions of this paper are: i) the design and optimization of ABRADE, a frame-based BRA with deferred feedback that applies a novel frame length adaptation strategy to minimize the resolution cost of batches of known size; ii) the use of a novel batch size estimate method to deal with batches of unknown size, iii) the definition and evaluation of ABRADE$^+$, an extension of ABRADE to resolve batches of unknown size; iv) the performance comparison of the proposed BRAs with the best performing BRAs based on the immediate feedback paradigm, in practical scenarios.

The rest of the paper is organized as follows. Sec. II details the system model and defines the main notation and the performance measures considered in this work. Sec. III provides an overview of the main BRAs with immediate feedback, here considered for comparison purposes. Sec. IV describes ABRADE and derives the dynamic frame length optimization strategy that is the core of the algorithm. The mathematical and asymptotic analysis of ABRADE performance, under the assumption of perfect knowledge of the batch size, is developed in Sec. V, whereas in Sec. VI we introduce the batch size estimate module and the ABRADE$^+$ scheme for batches of unknown size. Sec. VII presents a simulative comparison of ABRADE$^+$ with the best performing BRAs with immediate feedback in two practical scenarios, which are based on the common IEEE 802.11g and IEEE 802.15.4 radio standards. Finally, in Sec. VIII we sum up the main results and discuss possible extensions of the proposed schemes.

II. System Model and Performance Measures

Before going in more depth into the analysis of BRAs, it is convenient to formally define the system model and the performance measures considered in this study. For reader convenience, Tab. I collects the main notation used throughout the paper.

A. System model

We consider a scenario where all nodes in the batch are within the transmission/reception range of the inquirer and capable of carrier-sensing each other transmission [13]. Channel access is ruled by the standard CSMA scheme: nodes are allowed to transmit only when the channel has been idle for a certain time interval β, called *idle slot*. If two or more nodes transmit simultaneously, their packets collide and none of them is successfully decoded. For the sake of simplicity, we make the classical assumption of noiseless radio channel, so that packet transmissions are always successful, except in case of collision. Successful and collided transmissions take constant time β_s and β_c, respectively.[2] As customary, we normalize all the time intervals to the transmission time of a data packet, so that we assume $\beta_s = 1$. The time axis can then be logically partitioned in slots of unequal duration, which are referred to as *idle*, *successful* or *collided* according to whether they host zero, one or more than one simultaneous transmission, respectively.

As graphically exemplified in Fig. 1, in the deferred feedback case the time slots are organized in frames, and the feedback is provided only at the end of the frame by means of a probe message of duration β_p, whereas in the immediate feedback case, feedbacks are returned after each slot. The transmission time of the feedbacks for idle, successful and collided slots are denoted by ϕ_i, ϕ_s, and ϕ_c, respectively. Note that, in practical CSMA-based systems, idle feedback is implicitly provided by the carrier-sense mechanism, so that we have $\phi_i = 0$. Conversely, successful feedback is typically provided by means of a short acknowledgement packet (ACK), for which $0 < \phi_s < 1$. Finally, collided slots are recognized because the packet transmission is not followed by an ACK within a certain time interval that, hence, can be considered as the collision feedback time ϕ_c. A more accurate evaluation of these parameters for some practical scenarios is given in Appendix A-I.

B. Performance measures

The number of nodes in the batch is referred to as *batch size* and denoted by n. The *batch resolution interval* (BRI), denoted by $\mathcal{T}(n)$, is the mean time required to resolve all the n nodes. The *batch throughput* or, simply, throughput is the average nodes resolution rate, given by

$$\lambda(n) = \frac{n}{\mathcal{T}(n)} . \tag{1}$$

By letting n approach infinity, we get the *asymptotic (batch) throughput*,

$$\lambda_{\max} = \lim_{n \to \infty} \lambda(n) . \tag{2}$$

[2]For the sake of generality, we distinguish between the time duration of successful and collided transmissions. However, when the packet transmission time is constant, as in this study, we can safely assume $\beta_c = \beta_s$.

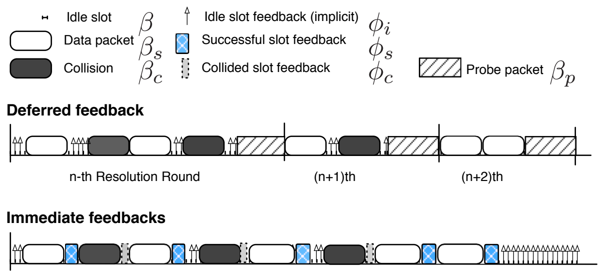

Fig. 1. Example of transmission events with related feedbacks in the deferred (upper) and immediate (lower) feedback scenarios.

TABLE I
MAIN NOTATION.

β_s, β_c, β	time duration of successful, collided and idle slot, resp.
ϕ_s, ϕ_c, ϕ_i	time duration of the feedback message for successful, collided, and idle slots, resp., with immediate feedback
β_p	time duration of probe message in ABRADE and ABRADE$^+$
n	current batch size (number of still unresolved nodes)
N	random variable modeling the batch size
$P_N(\cdot)$	mass distribution function of the batch size N
$\mathrm{m}_N = \mathrm{E}[N]$	mean batch size
$\mathcal{T}(n)$	mean batch resolution interval for batches of size n
$\lambda(n)$	throughput for batches of size n
$\bar{\lambda}(\mathrm{m}_N)$	throughput for batches of random size with mean m_N
$\lambda_{\max}$	asymptotic throughput as $n \to \infty$
w	frame length (in number of slots)
$w_n^\star$	optimal contention frame length with batch size n
μ_∞	asymptotic mean number of transmissions per slot
S, C, I	random variables denoting the number of successful, collided and idle slots in a frame, resp.
$P_{w,n}(s,c)$	joint probability mass distribution of S and C with n nodes transmitting in a frame of w slots
$p_{w,n}(s)$	probability mass distribution of S
p	contention probability
w_0	initial frame size
Δ	normalized mean square estimate error
$P_N^\circ(\cdot)$	arbitrary mass distribution function of the batch size, typically uniform in $[0, N_{\max}-1]$ (design parameter)
m_N°	arbitrary mean batch size (design parameter)

The asymptotic throughput corresponds to the maximal packet arrival rate that can be sustained by the system when the BRA is used as an *obvious MAC*, according to which all packets generated during the resolution of a batch are held in queue and processed as a new batch once the previous batch has been resolved [12]. A stochastic batch size is represented as an integer random variable N, with probability mass distribution $P_N(n)$ and mean $\mathrm{m}_N = \mathrm{E}[N]$. In this case, we define the *mean batch throughput* as

$$\bar{\lambda}(\mathrm{m}_N) = \frac{\mathrm{E}[N]}{\mathrm{E}[\mathcal{T}(N)]} = \frac{\mathrm{m}_N}{\sum_{n=0}^{\infty} P_N(n)\mathcal{T}(n)} = \frac{\mathrm{m}_N}{\sum_{n=0}^{\infty} \frac{nP_N(n)}{\lambda(n)}} . \tag{3}$$

III. BATCH RESOLUTION ALGORITHMS WITH IMMEDIATE FEEDBACK

The common basic idea behind the splitting-tree algorithms consists in stochastically splitting the initial conflict set in smaller subsets, which are then resolved one after the other. The resolution algorithm arbitrates the channel access of the nodes, progressively eliminating (resolving) the nodes that transmit successfully. In case of packet collision, the resolution algorithm is recursively applied to the set of collided nodes, according to the *divide and conquer* paradigm. When all nodes involved in a collision are resolved, the resolution process proceeds with the other nodes in the batch. The various schemes differ in the policy used to split the conflict set in subsets and to resolve the collisions.

In many practical cases, the *batch size* (or conflict multiplicity) n, is unknown. This lack of information generally results into some performance loss. To counteract this inefficiency, in [14] and [15] authors define hybrid algorithms consisting of two parts, the first devoted to the estimation of the batch size n, and the second to resolve the (residual) batch by using classical splitting-tree algorithms. More recently, Popovski *et al.* [8], [9] proposed an entire class of collision resolution algorithms that closely integrates the estimation and resolution phases.

For space constraints, in the following of this section we will focus on the most-representative BRAs only, which will be successively considered for performance comparison with ABRADE and ABRADE$^+$. We first describe FCFS, the best-performing BRA with immediate feedback for batches of known size. We also revise the classical performance analysis of FCFS by including the feedback costs and thus showing that the asymptotic throughput of FCFS may be significantly lower than what reported in the classical literature, which neglected the feedbacks cost. Successively, we discuss the BRAs proposed by Popovski *et al.* for batches of unknown size.

A. The FCFS/CMBT algorithm

The First-Come First-Served (FCFS) algorithm was proposed by Gallager in [6] as an obvious MAC scheme for systems with Poisson packet arrival process. The initial batch is split according to the packets arrival time: time axis is divided in intervals of duration h and nodes are resolved sequentially, starting from those that generated packets in the first interval, then in the second interval and so on.

At every slot, a different time interval $I = t_0 + [0, h)$ is *activated*, which means that all packets generated in the interval I are transmitted in that slot. If the slot remains idle, or contains a single packet transmission (which is hence successfully received), the following interval $t_0 + [h, 2h)$ will be activated in the next slot. Conversely, in case of collision, the original time interval I is split in two subintervals, namely the left interval $I_L = t_0 + [0, f\,h)$ and the right interval $I_R = t_0 + [f\,h, h)$, where $f \in (0, 1)$ is a design parameter. The left interval I_L is then activated, and all packets generated in I_L are transmitted in the next slot. The procedure is repeated recursively, till one node is resolved. Then, the right interval is activated and so on.

A simple improvement to the algorithm consists in avoiding *guaranteed* collisions, which occur when an interval I contains multiple packets, all concentrated in the right part of I. In this case, the activation of I yields a collision with the consequent splitting of I in the left and right subintervals and the activation of I_L in the next slot. However, I_L does not contain any packets, so that the slot remains idle. Then, the activation of I_R, as in the original algorithm, would certainly produce a collision. To avoid this inefficiency, any time a collided slot is followed by an empty one, the right subinterval is not activated, but immediately split in two new subintervals, and the left one is activated. The splitting procedure is repeated for any successive idle slot, thus progressively reducing the length of the activated interval, till the first successful or collided slot. At this point, the original algorithm execution is resumed and the resolution process proceeds as usual.

Another improvement is brought about by the *clipping* mechanism that aims at avoiding the activation of intervals of suboptimal size. The rationale is that, for Poisson arrivals, two (or more) collisions in a run "erase" any prior information on the number of nodes in the right subsets. Therefore, after a left subinterval is resolved, the clipping mechanism breaks the recursion and activates a new time interval of optimal length h, rather than a shorter interval. (Note that, this rationale is valid only under the assumption of Poisson arrivals [12].)

The version of FCFS that includes these improvements, also known as *clipped modified binary tree* (CMBT), is the fastest conflict resolution algorithm in the literature for Poisson arrivals [7].

1) The impact of feedback costs: in [12], Gallager characterizes FCFS for CSMA channels and derive an approximate expression of the achievable asymptotic throughput $\lambda_{\max}$, under the classical assumption

$$\phi_i = 0\,, \quad \phi_c = \phi_s = \beta \ll 1\,. \tag{4}$$

Unfortunately, (4) may not hold in practical systems, where feedback times are not negligible, so that the classical performance analysis turns out to be overoptimistic. We thus revised the derivation of $\lambda_{\max}$ proposed in [12] removing the assumption that ϕ_s and ϕ_c are negligible. The resulting expression is

$$\lambda_{\max} \simeq \frac{g+g^2}{2\beta + (1+\phi_s)(g+g^2)}\,, \tag{5}$$

where

$$g = \sqrt{\frac{2\beta}{1+\phi_c+\sqrt{\beta}}} \tag{6}$$

is the optimal mean number of transmissions per slot. Furthermore, the optimal value of the splitting coefficient f turned out to be equal to

$$f = -\beta + \sqrt{a^2+a}\,, \quad \text{with} \quad a = \frac{\beta}{1-\beta+\phi_c}\,. \tag{7}$$

Note that, setting the parameters as in (4), Eq. (5) returns the original expressions provided by Gallager in [12]. However, if $\phi_s > \beta$, as in many practical systems, the maximal throughput given by (5) becomes notably lower than the value found in literature, in particular for systems with high transmission rates, for which $\phi_s \to \beta_s = 1$. A more detailed analysis of the asymptotic throughput of FCFS with $\phi_s \geq \beta$ will be presented in Sec. V.

We remark that FCFS has been designed to resolve conflicts among messages that arrive according to a *Poisson* process with *known* arrival rate λ. Therefore, the plain application of FCFS algorithm in case of batch arrivals yields poor performance [14]. However, the optimality of the algorithm can be easily re-established for batch arrivals, provided that the batch size is Poisson distributed with known mean m_N. In fact, as explained in [9], it suffices that each node in the batch generates a random *virtual arrival instant* in the time interval $[0, \mathrm{m}_N/\lambda_{\max}]$. The FCFS algorithm can then be applied by considering the virtual arrival instants in place of the actual message generation times.

B. The IECR algorithm

FCFS optimality requires to know beforehand the mean batch size m_N, an information that may not be available in many practical situations. To overcome this limit, Popovsky *et al.* proposed in [9] the Interval Estimation Collision Resolution (IECR) algorithm. The basic idea consists in estimating the batch size while the batch resolution is in progress. In practice, IECR starts with all nodes setting their virtual arrival epochs uniformly in the unit interval $(0,1)$ and the FCFS algorithm is performed. In case of collision, the last activated interval is split in two subintervals, the left one corresponding to a fraction f of the original interval. The algorithm then proceeds by activating the left subinterval, as in FCFS. When a collision is successfully solved, however, the algorithm activates a new interval of length

$$h_{IECR} = \frac{g}{\hat{\lambda}}\,, \tag{8}$$

with g as in (6), whereas $\hat{\lambda}$ is the current estimate of the average arrival rate given by

$$\hat{\lambda} = \frac{k}{f_{lim}}\,,$$

where k and f_{lim} are the number of nodes already resolved and the length of the resolved interval, respectively. IECR is proved to reach the same asymptotic throughput of FCFS, without requiring any *a priori* knowledge of the batch size.

C. The Sift and Sift/IECR algorithms

Another interesting approach to collision resolution in CSMA channels is proposed in [16], where authors introduce the Sift algorithm. Sift is not properly a BRA because it aims at minimizing the time till the *first successful* transmission, rather than the batch resolution interval. Furthermore, conversely to the binary-tree BRAs, Sift applies a modified framed ALOHA channel access scheme, still with immediate feedback.

More specifically, time is divided in frames, each consisting of $w = 32$ slots. At the beginning of a frame, each unresolved node chooses at random one slot in the frame where to transmit. Slot $j \in \{1, 2, \ldots, w\}$ is picked with probability

$$p_{Sift}(j) = \frac{(1-\alpha)\alpha^w}{1-\alpha^w}\alpha^{-j}\,, \tag{9}$$

where $0 < \alpha < 1$ is a design parameter that depends on the maximum expected batch size value $N_{\max}$. The algorithm ends at the first transmission, which always occurs within the frame because $\sum_{j=1}^{w} p_{Sift}(j) = 1$. The probabilities (9) are designed in order to maximize the chance that the first transmission is successful, regardless of the batch size.[3]

[3]Actually, Sift was engineered for $N_{\max} = 512$ nodes, for which $\alpha = 512^{-1/31}$ that is the value considered in this paper. However, authors showed that performance degrades gracefully when the batch size exceeds this limit.

Popovsky suggested to merge Sift with IECR, thus generating the hybrid Sift/IECR algorithm. The idea is to use Sift to get a first estimate of the batch size and, then, apply IECR. Suppose that Sift ends at slot $m \in [1, w]$ and let

$$f_{lim} = \sum_{j=1}^{m} p_{Sift}(j)\,, \quad f_{low} = f_{lim} - p_{Sift}(m)\,.$$

If Sift ends with a successful transmission, then IECR is started having the interval $[0, f_{lim})$ resolved, and $k = 1$. Conversely, if Sift ends with a collision, then IECR is started as if a collision occurred in the interval $[f_{low}, f_{lim})$. The Sift/IECR algorithm is expected to yield better performance than IECR, in particular for small values of the batch size.

IV. Adaptive Batch Resolution Algorithm with Deferred Feedback: ABRADE

In this section we describe ABRADE and the dynamic frame length adaptation strategy that is the core of the algorithm. The theoretical analysis of ABRADE is developed under the assumption that the batch size n is known, whereas Sec. VI discusses the extension to the case of unknown n.

As mentioned in the introduction, ABRADE works in successive *resolution rounds*. In each round, every single unresolved node, independently of the others, picks a random slot in the *contention frame* of w slots, with uniform probability, and transmits its packet in that slot, according to a CSMA-based framed ALOHA scheme. At the end of the contention frame, the inquirer broadcasts a *probe message* that carries the aggregate feedback field (ACK), together with other control information, included the frame length w to be used in the next round. Acknowledged nodes are resolved and leave the batch, whereas the others transmit in the next round. The batch resolution process is completed when all nodes have been resolved.

The core of ABRADE consists in adjusting the frame length w after each resolution round, according to the residual batch multiplicity n. The adaptation aims at maximizing the batch throughput $\lambda(n)$ given by (1) that, in turn, corresponds to minimizing the BRI $\mathcal{T}(n)$. In other words, for each n we wish to find the frame length $w_n^\star$ such that

$$w_n^\star = \arg\min_{w} \{\mathcal{T}(n)\} \tag{10}$$

with

$$\mathcal{T}(n) = \mathrm{E}[\tau(n)]\,, \tag{11}$$

where $\tau(n)$ is the random variable that represents the actual time required to resolve a batch of size n. To this end, we express the BRI in a recursive form that lends itself to dynamic programming optimization.

Then, let $\chi_S(j)$, and $\chi_I(j)$ denote binary random variables that take the value 1 if the jth slot in the frame is successful or idle, respectively, and zero otherwise. Furthermore, let S, C and I be integer random variables that denote the number of successful, collided and idle slots in a frame of w slots, respectively, so that we can write

$$S = \sum_{j=1}^{w} \chi_S(j); \quad I = \sum_{j=1}^{w} \chi_I(j); \quad C = w - S - I\,. \tag{12}$$

Since each slot in the frame is independently selected by any of the n nodes with equal probability $1/w$, we have that $\mathrm{E}[\chi_S(j)] = \frac{n}{w}(1-\frac{1}{w})^{n-1}$ and $\mathrm{E}[\chi_I(j)] = (1-\frac{1}{w})^n$ for any $j = 1, \ldots, w$, so that the conditional expectations of S, C and I, given w and n, can be readily determined from (12) as

$$\mathrm{E}[S|w,n] = n\left(1-\frac{1}{w}\right)^{n-1}; \quad \mathrm{E}[I|w,n] = w\left(1-\frac{1}{w}\right)^{n};$$
$$\mathrm{E}[C|w,n] = w\left[1 - \frac{n}{w}\left(1-\frac{1}{w}\right)^{n-1} - \left(1-\frac{1}{w}\right)^{n}\right]. \tag{13}$$

Furthermore, let $P_{w,n}(s,c) = \mathrm{P}[S=s, C=c|w,n]$ be the conditional joint probability mass distribution of S and C given that the frame size is w slots and there are n transmitting nodes. In other words, $P_{w,n}(s,c)$ is the probability of having S successful and C collided slots in a frame of w slots, when n nodes independently transmit a packet in a random slot. Clearly, $P_{w,n}(s,c) = 0$ when any argument is negative, or $s + 2c > n$, or $s + c > w$, whereas $P_{w,0}(0,0) = 1$. In the other cases, the expression of $P_{w,n}(s,c)$ can be obtained by combinatorial analysis (see, e.g., [17], [18]) or, more conveniently, through the recursive form given in [19]:

$$\begin{aligned} P_{w,n}(s,c) = & P_{w,n-1}(s,c)\frac{c}{w} + P_{w,n-1}(s+1,c-1)\frac{s+1}{w} \\ & + P_{w,n-1}(s-1,c)\frac{w-c-s+1}{w}\,. \end{aligned} \tag{14}$$

The terms on the right-hand side of (14) account for the probabilities of the following three disjoint events that may occur when n nodes randomly transmit over a frame of w slots: i) $n-1$ transmissions result in s successful and c collided slots, and the nth node picks one of the collided slots; ii) $n-1$ transmissions result in $s+1$ successful and $c-1$ collided slots, and the nth node transmits in one of the successful slots, which then becomes collided; iii) $n-1$ transmissions result in $s-1$ successful and c collided slots, and the nth node transmits in an idle slot, which then becomes successful.

Finally, let $p_{w,n}(s) = \mathrm{P}[S=s|w,n]$ be the probability mass distribution of S, given w and n, which can be obtained from (14) through the marginal law:

$$p_{w,n}(s) = \sum_{c=0}^{w} P_{w,n}(s,c)\,. \tag{15}$$

The mean batch resolution time $\mathcal{T}(n)$ defined in (11) can then be expressed in the following recursive form

$$\mathcal{T}(n) = \mathrm{E}[y_{w,n} + \tau(n-S)] = \mathrm{E}[y_{w,n}] + \mathrm{E}[\tau(n-S)] \tag{16}$$

where S is the random variable denoting the number of successful slots in the first resolution round, whereas $y_{w,n}$ denotes the duration of that resolution round that, in turn, can be expressed as

$$y_{w,n} = S + C\beta_c + I\beta + \beta_p\,. \tag{17}$$

Recalling (13), the statistical mean of (17) turns out to be equal to

$$\begin{aligned}\mathrm{E}\left[y_{w,n}\right] =& \beta_p + w\beta_c + n\left(1-\frac{1}{w}\right)^{n-1}(1-\beta_c) \\ &+ w\left(1-\frac{1}{w}\right)^{n}(\beta-\beta_c)\,.\end{aligned} \tag{18}$$

Applying the total probability theorem with respect to S to the right-most term of (16) we thus obtain[4]

$$\begin{aligned}\mathcal{T}(n) &= \mathrm{E}\left[y_{w,n}\right] + \sum_{s=0}^{\infty} \mathrm{E}\left[\tau(n-S)|S=s\right] p_{w,n}(s) \\ &= \frac{\mathrm{E}\left[y_{w,n}\right] + \sum_{s=1}^{\infty} \mathcal{T}(n-s) p_{w,n}(s)}{1-p_{w,n}(0)}\,,\end{aligned} \tag{19}$$

where the last step is obtained by collecting the terms $\mathcal{T}(n)$ on the left-hand side of the equation. Finally, using (18) in (19), we get the optimization problem (10) expressed in the following recursive Bellman-equation form

$$\begin{aligned}w_n^\star = \arg\min_w \Bigg\{ & \frac{\beta_p + w\beta_c + n\left(1-\frac{1}{w}\right)^{n-1}(1-\beta_c)}{1-p_{w,n}(0)} \\ &+ \frac{w\left(1-\frac{1}{w}\right)^{n}(\beta-\beta_c) + \sum_{s=1}^{\infty} p_{w,n}(s)\mathcal{T}^\star(n-s)}{1-p_{w,n}(0)} \Bigg\}\,,\end{aligned} \tag{20}$$

where $\mathcal{T}^\star(k)$ denotes the optimal BRI for a batch of size k. This minimization problem can be solved recursively, starting from $k=1$, for which $\mathcal{T}^\star(1) = 1+\beta_p$ and $w_1^\star = 1$, and using (19) to obtain $\mathcal{T}^\star(k)$ for $k>1$.

V. Performance Analysis of ABRADE with Known Batch Size

In this section we analyze the performance of ABRADE under the assumption that the batch size is known, so that the optimal frame length can be used throughout the entire batch resolution process.

To begin with, in Fig. 2 we plot the optimal frame length $w_n^\star$ as a function of the batch size n for different values of β, with $\beta_c = 1$, $\beta_p = 1$ (solid lines) and $\beta_p = 2$ (dashed lines). We can observe that $w_n^\star$ grows almost linearly with n, though the optimal contention frame is proportionally longer for small batches than for large ones. The reason is that, with small batches, the cost of probe messages is determinant, so that it is advantageous to choose longer frames to reduce the collision probability and, in turn, the number of resolution rounds required to resolve the batch, though at the cost of a larger number of idle slots.

In Fig. 3 we report the throughput curves as a function of β, for different values of n. All curves have been obtained by setting $\beta_p = \beta_c = 1$. We note that the throughput increases with n, which confirms that the overhead due to the control traffic becomes progressively less relevant as the size of the batch grows. As n grows to infinity, the throughput rapidly approaches the asymptotic throughput $\lambda_{\max} = \lim_{n\to\infty}\lambda(n)$, represented by the dashed curve in the figure. In Appendix A-II we derive the expression of $\lambda_{\max}$, which turns out to be equal to

$$\boxed{\lambda_{\max} = \frac{e^{-\mu_\infty}}{b_p + \beta_c + e^{-\mu_\infty}(1-\beta_c)}}\,, \tag{21}$$

where b_p accounts for the fact that, the longer the frame, the longer the aggregate ACK field and, in turn, the size of the probe message,[5] whereas μ_∞ is the asymptotically optimal mean number of transmissions per slot, which is given by (see Appendix A-I)

$$\mu_\infty = \lim_{n\to\infty}\frac{n}{w_n^\star} = 1 + \mathcal{W}_0\left(-\frac{\beta_c-\beta}{(b_p+\beta_c)e}\right), \tag{22}$$

where $\mathcal{W}_0(\cdot)$ is the Lambert$W$ function [20]. Note that μ_∞ strictly depends on the idle slot time β: when β grows from 0 to β_c the asymptotic mean number of transmissions per slot μ_∞ also increases from 0 to 1, as shown in Fig. 4. This behavior reflects the principle according to which the channel access strategy shall be more aggressive when idle and busy slots have comparable duration.

In Fig. 5 we compare the asymptotic throughput of FCFS (solid black lines) and ABRADE (dashed red line) when

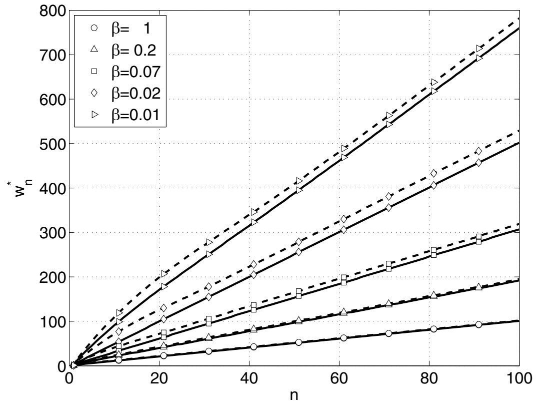

Fig. 2. Optimal frame length $w_n^\star$ as a function of the batch size n, for different values of β, with $\beta_c = 1$, $\beta_p = 1$ (solid) and $\beta_p = 2$ (dashed).

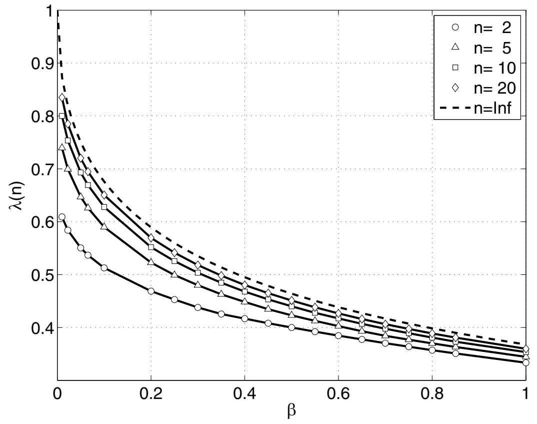

Fig. 3. Throughput $\lambda(n)$ *vs* β, for different values of the batch size n, with $\beta_p = \beta_c = 1$. Dashed curve represents $\lambda_{\max}$.

[4]For ease of notation we extend the sum in s to infinity, though $p_{w,n}(s) = 0$ for $s > \min\{w,n\}$.

[5]As we will see in Appendix A-I, b_p is generally negligible in practical scenarios.

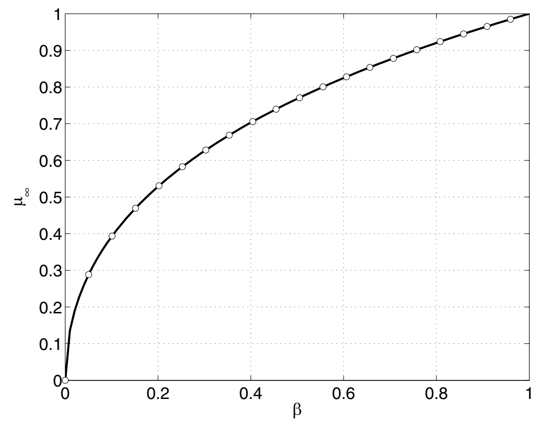

Fig. 4. Asymptotic number of transmissions per slot, μ_∞, as a function of the normalized idle slot duration β, with $\beta_c = 1$, $b_p = 0$.

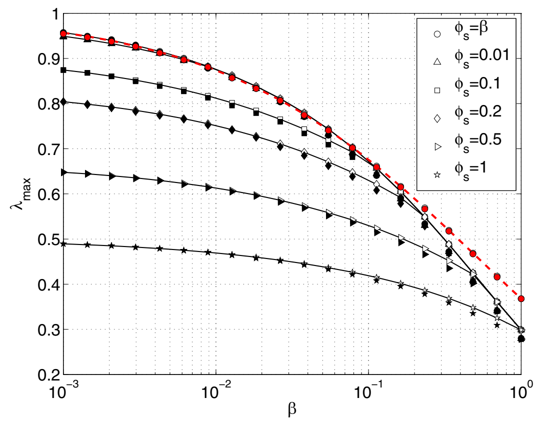

Fig. 5. Asymptotic throughput $\lambda_{\max}$ of FCFS (solid) and ABRADE (dashed) *vs* the idle slot duration β, for different values of the success-feedback duration ϕ_s ($\phi_c = \beta, b_p = 0$, $\beta_c = 1$). Empty markers refer to theoretical results, filled markers to simulation results, obtained with batch size $n = 1500$.

varying the idle slot duration β, and for different values of the success-feedback time ϕ_s.[6] Points obtained with the theoretical expressions (5) and (21) are represented by empty markers, whereas filled markers denote the simulation results obtained with batch size $n = 1500$. The good matching between simulation and theoretical values confirms the validity of the asymptotic throughput analysis for both FCFS and ABRADE. The top-most throughput curve for FCFS, which is almost completely superimposed to the ABRADE's curve, reports the asymptotic throughput according to the original equation provided in [12], under the classical assumption $\phi_s = \beta$. We note that, whenever this assumption fails to apply, i.e., $\phi_s > \beta$, the asymptotic FCFS throughput turns out to be significantly lower than the value obtained by neglecting the feedbacks cost. Conversely, ABRADE achieves the maximum asymptotic throughput because it avoids the overhead introduced by the immediate feedback paradigm.

[6]Note that, all points in the graph have been obtained by setting ϕ_s to the minimum between the value reported in the legend and β, since in practical systems the successful feedback time is never shorter than the idle slot time. Furthermore, we set $\phi_c = \beta$ for FCFS, and $\beta_c = 1, b_p = 0$ for ABRADE.

```
  //* Inquirer side *//
Input: $\mathrm{m}_N, p, w_0$
  //* Initialize frame size $w$, $ACK$ field, stopping flag *//
  $w \leftarrow w_0$; $ACK$ ←null; $resolved \leftarrow$ false
  send_probe_pck($w, p, ACK$)
  while not($resolved$) do
    //* Listen to the channel for $w$ slots *//
    $\{s, c, i\} \leftarrow$receive_from_channel($w$)
    //* estimate batch size using (29) *//
    $\hat{n} \leftarrow \mathcal{H}_{w,p}(s, c)$
    $n_{est} \leftarrow \lceil \hat{n} - s \rceil$
    if $\{n_{est} = \infty\}$ or $\{p < 1, n_{est} = 0\}$ then
      Update $p$ and $w$ as described in Sec. VI-B
    else
      //* Dynamically adjust the frame length according to (20) *//
      $p \leftarrow 1, w \leftarrow w^\star_{n_{est}}$
      $resolved \leftarrow \{n_{est} = 0\}$
    end if
    ACK ← $s$ //* Update ACK field *//
    send_probe_pck($w, p, ACK$)
  end while

  //* Node side *//
Input: $\{w, p,$ACK$\} \leftarrow$ receive_probe_pck
  $resolved \leftarrow$ false
  while not($resolved$) do
    $k \leftarrow 0$
    $compete \leftarrow$ (rand $\leq p$) //* Compete with probability $p$ *//
    if $compete$ then
      $k \leftarrow \lceil$rand $\times$ $w\rceil$ //* Choose one random slot among $w$ *//
      receive_from_channel($k - 1$) //* Wait for $k - 1$ slots *//
      send_data_pck() //* Send packet on the selected kth slot *//
    end if
    $\{w, p,$ACK$\} \leftarrow$ receive_probe_pck
    if slot $k$ ACKed then
      $resolved \leftarrow$ true
    end if
  end while
```

Fig. 6. ABRADE$^+$ pseudocode at the inquirer and node side.

VI. Extending ABRADE to Batches of Unknown Size

When the exact number of nodes in the conflict set is not known beforehand, ABRADE needs to be coupled with a batch size estimate module that provides an estimate $\hat{n}$ of the number of nodes in the batch. This estimation problem is interesting by its own and has received considerable attention in the literature, e.g. [8], [9], [14], [21], [22] just to cite a few.

Here we propose a novel estimate procedure that, integrated with ABRADE, provides a complete solution for the batch resolution problem. This new algorithm, named ABRADE$^+$, inherits the contention scheme and the frame length adaptation strategy of ABRADE, with two important add ons: the *batch size estimate function* (BSEF) and the *contention probability* p.

The BSEF is invoked after each resolution round to get an updated estimate of the residual batch size, which is then used in place of the exact value of n to select the frame length for the next round, as in ABRADE.

The contention probability p, which is broadcasted in the probe message together with the frame length w, determines the probability that a node in the batch transmits in the upcoming round. As we will see later, this parameter is particularly useful at the very beginning of the resolution process, when the estimate of the batch size is unreliable.

In Fig. 6 we provide the pseudo-code of ABRADE$^+$ for both inquirer and nodes, whereas in the following, we first detail the BSEF and, then, discuss the estimate start up problem.

A. The batch size estimate function

The BSEF, denoted by $\mathcal{H}_{w,p}(s,c)$, is a deterministic multivariate function that, given the frame length w, the contention probability p, and the number s and c of successful and collided slots at the end of the frame, returns the following estimate $\hat{n}$ of the batch size:

$$\hat{n} = \mathcal{H}_{w,p}(s,c) = \frac{\hat{n}_t}{p}\,, \tag{23}$$

where $\hat{n}_t$ is the estimate of the number of nodes that have actually transmitted in the frame.[7] This last estimate, in turn, is given by

$$\hat{n}_t = s + c\,n_c\,, \tag{24}$$

where $n_c \geq 2$ denotes the mean number of nodes involved in each collision. To determine n_c, we assume that the number of transmissions per slot can be approximated as a Poisson-distributed random variable with parameter μ, as done in [21]. Note that this assumption does not necessarily reflect the actual distribution of the number of transmissions per slot, in particular when the batch size or the frame length are small. Nonetheless, it makes it possible to derive a simple estimator since, under this assumption, n_c can be expressed as

$$n_c = \frac{\mu - \mu\exp(-\mu)}{1 - \exp(-\mu) - \mu\exp(-\mu)}\,, \tag{25}$$

that, replaced in (24), gives

$$\hat{n}_t = s + c\,\frac{\mu - \mu\exp(-\mu)}{1 - \exp(-\mu) - \mu\exp(-\mu)}\,. \tag{26}$$

The value of μ is finally obtained by equalling (26) to μw, which is the expected number of transmissions in the frame, so that we obtain

$$s + c\frac{\mu - \mu\exp(-\mu)}{1 - \exp(-\mu) - \mu\exp(-\mu)} = \mu w\,. \tag{27}$$

Although transcendent, (27) is well behaved and admits a single positive solution $\hat{\mu}$ that can be quickly determined by using standard methods, as bisection search. Hence, from (26) and (27), it follows

$$\hat{n}_t = \hat{\mu} w\,, \tag{28}$$

that applied to (23) yieds

$$\boxed{\hat{n} = \mathcal{H}_{w,p}(s,c) = \frac{\hat{\mu} w}{p}\,,} \tag{29}$$

where $\hat{\mu}$ is the solution of (27).

For space constraints we cannot present the performance analysis of (28) and (29). However, we observed that the proposed method performs very closely to the maximum-likelihood estimator, though with much lower computational complexity. For more details, we refer the interested reader to [17].

[7]For compactness, we omit to indicate the dependency of $\hat{n}$ and $\hat{n}_t$ on s, c, w, and p.

B. The estimate startup problem

At the very beginning of the resolution process, when the inquirer has only partial or no information about the batch multiplicity, the frame length has to be set to an arbitrary value w_0. In this condition of uncertainty, it is convenient to choose a small value of the initial frame size w_0, in order to have a first estimate of the batch size in the shortest time possible. However, if the batch is large, the frame may experience a lot of collisions that will inflate the BRI. To reduce the risk of collisions, ABRADE$^+$ statistically reduces the number of transmitting nodes through the contention probability p. The choice of p and w_0 shall strike a balance between the performance loss incurred in the first resolution round, due to suboptimal parameters setting, and the accuracy of the estimate $\hat{n}$ given by (29) at the end of the round. In the following we describe a possible strategy to set p and w_0 in order to cut this tradeoff.

Let $\overset{\circ}{P}_N(\cdot)$ denote the *a priori* probability mass distribution of the initial batch size N assumed at the inquirer. If the actual statistical distribution $P_N(\cdot)$ of the batch size is known at the inquirer, we clearly set $\overset{\circ}{P}_N(n) = P_N(n)$ for each n. However, if the inquirer has no prior knowledge about the system, $\overset{\circ}{P}_N(\cdot)$ is arbitrarily assumed to be uniform in the integer interval $[0, N_{\max}-1]$, where $N_{\max}$ is a design parameter. Let m°_N be the statistical mean of N according to the distribution $\overset{\circ}{P}_N(\cdot)$.

The contention probability p is set in such a way that the expected number of transmitting nodes in the first round, $p\mathrm{m}^{\circ}_N$, equals the expected value of the *optimal* number of transmissions in the frame, given by $w_0 \sum_n \mu_n \overset{\circ}{P}_N(n)$. We thus obtain

$$p = \frac{w_0 \sum_{n=0}^{\infty} \mu_n \overset{\circ}{P}_N(n)}{\mathrm{m}^{\circ}_N} \simeq \frac{w_0 \mu_\infty}{\mathrm{m}^{\circ}_N}\,, \tag{30}$$

where the right-most expression holds for $\mathrm{m}^{\circ}_N \gg 1$. If (30) returns $p > 1$, we clearly set $p = 1$.

We now need to set w_0 in order to have a sufficiently accurate estimate $\hat{n}$ of the actual batch size after the first resolution round without inflating much the BRI. To this end, we introduce the mean square estimation error that, for a batch size n, is defined as

$$\varepsilon^2(n) = \sum_{m=0}^{n} P_M(m|n) \sum_{s=0}^{w} \sum_{c=0}^{w-s} P_{w,m}(s,c)(\hat{n}-n)^2 \tag{31}$$

where $\hat{n}$ is given by (29), whereas

$$P_M(m|n) = \sum_{m=0}^{n} \binom{n}{m} p^m (1-p)^{n-m}, \quad m = 0, 1, \ldots, n\,. \tag{32}$$

is the probability mass distribution of the number of nodes that do transmit in the first slot. We then set w_0 such that

$$w_0 = \min\left\{ w : \mathrm{E}\left[\varepsilon^2(n)\right] = \sum_{n=0}^{\infty} \overset{\circ}{P}_N(n)\varepsilon^2(n) \leq \Delta {\mathrm{m}^{\circ}_N}^2 \right\}, \tag{33}$$

where Δ is a design parameter called *normalized mean square estimate error threshold*. Eq. (33) is evaluated by setting p as in (30). This minimization problem can be solved numerically, starting from $w = 0$ and adding one till the inequality is satisfied. As an example, in Fig. 7 we report the result obtained

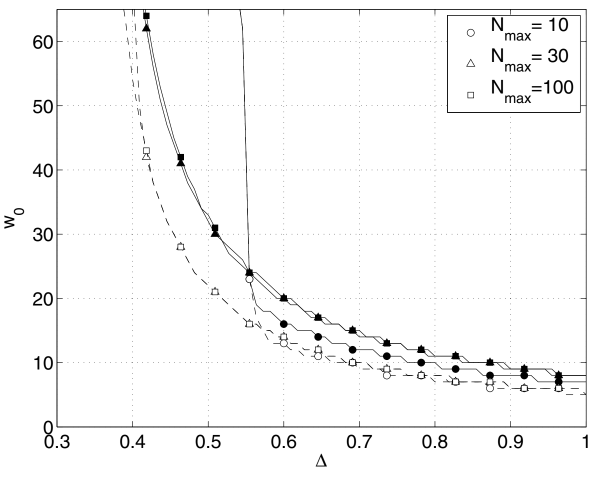

Fig. 7. Estimate-startup contention frame size w_0 *vs* error threshold Δ with $\beta = 0.0225$ (black markers) and $\beta = 0.0654$ (white markers).

when varying Δ, assuming $\overset{\circ}{P}_N(\cdot)$ is uniform in the interval $[0, N_{\max} - 1]$ and setting $\beta = 0.0225$ (black markers) and $\beta = 0.0654$ (white markers), as for the case study scenarios that will be described in Sec. VII. It is interesting to note that w_0 grows very quickly for $\Delta < 0.5$, which means that any further improvement of the estimate accuracy beyond this threshold would require a much larger increment of the frame length.

Finally, the estimate startup phase requires some empirical adjustments to avoid pitfalls. First of all, when a round ends with no transmissions, but $p < 1$, ABRADE$^+$ shall check whether the residual batch is actually empty or there are still unresolved nodes that have not transmitted in the last round. To this end, we update the value of m_N° by considering that the first round was empty, i.e., we set

$$\mathrm{m}_N^\circ = \mathrm{E}[N|S=0, C=0] = \frac{\sum_n n(1-p)^n \overset{\circ}{P}_N(n)}{\sum_n (1-p)^n \overset{\circ}{P}_N(n)} . \quad (34)$$

If $\overset{\circ}{P}_N(n)$ is uniform in $[0, N_{\max} - 1]$, (34) turns out to be equal to

$$\mathrm{m}_N^\circ = \frac{1-p}{p} - \frac{N_{\max}(1-p)^{N_{\max}}}{1-(1-p)^{N_{\max}}} . \quad (35)$$

The distribution $\overset{\circ}{P}_N(n)$ is then set to be uniform in $[0, 2\mathrm{m}_N^\circ]$, irrespective of any prior information about $P_N(\cdot)$, and w_0 and p are updated according to (33) and (30), respectively, using these new parameters.

A last adjustment is required when the actual batch size is much larger than expected and all the slots in the frame turns out to be collided, so that (29) returns an infinite estimate. In this case, we set $\mathrm{m}_N^\circ = \mathcal{H}_{w,p}(1, w-1) + \frac{1}{p}$ and update $\overset{\circ}{P}_N(n)$, p and w_0 as above.

VII. Case Study

In this section, we compare ABRADE$^+$ against the best-performing BRAs in the literature. To give practical significance to the comparison, we set the system parameters according to the specifications of two popular off-the-shelf radio technologies, namely IEEE 802.11 and IEEE 802.15.4, that, for ease of notation, will be henceforth referred to as WF (WiFi) and ZB (ZigBee), respectively. The values of the system parameters for the two reference scenarios are derived in Appendix A-I and collected in Tab. II. For ABRADE$^+$, we fixed $\Delta = 0.6$ because, according to the results in Fig. 7, smaller values of Δ would require larger initial frame sizes w_0 that, as explained in the previous section, shall be avoided.

TABLE II
System parameters setting in WF and ZB scenarios
($\beta_p = h_0 + b_p w$).

	T_{data} [μs]	β	ϕ_i	ϕ_s [10^{-3}]	ϕ_c	h_0	b_p
WF	399	22.5	0	131.9	131.9	143.2	0.05
ZB	4896	65.4	0	111.1	45.8	248.4	0.82

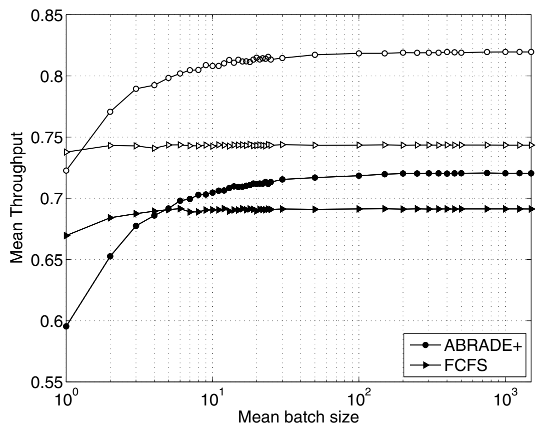

Fig. 8. Mean throughput $\bar{\lambda}(\mathrm{m}_N)$ of ABRADE$^+$ and FCFS as a function of the mean batch size m_N, in WF (white markers) and ZB (black markers) scenarios.

A. Poisson-distributed batch size with known mean

We first compare ABRADE$^+$ against the FCFS algorithm that, as explained in Sec. III, is the best known collision resolution algorithm for batches with Poisson-distributed multiplicity of known mean. Then, for fair comparison, we here assume that the batch size is Poisson-distributed with mean m_N that is varied from 1 to 1500. The value of m_N is assumed to be known to the algorithms, so that FCFS can generate the virtual arrival instants in the optimal manner, i.e., uniformly in the interval $[0, \mathrm{m}_N/\lambda_{\max}]$, whereas ABRADE$^+$ can set the parameters p and w_0 considering $\mathrm{m}_N^\circ = \mathrm{m}_N$ and $\overset{\circ}{P}_N(n) = \frac{\mathrm{m}_N{}^n}{n!}\exp(-\mathrm{m}_N)$. For each value of m_N, we run 20000 independent instances of both the algorithms, generating at each simulation a new batch, with multiplicity drawn at random from the Poisson distribution of mean m_N. The 99% confidence intervals are tight-fitting the curves and, for clarity, have not been reported in the graph.

Fig. 8 shows the *mean* throughput (3) in WF (white markers) and ZB (black markers) scenarios. Throughput curves rapidly grows with m_N, to bend toward the asymptotic values when $\mathrm{m}_N \geq 150$. For very small batches, FCFS is slightly better than ABRADE$^+$, which suffers the longer duration of the probe message with respect to the immediate feedback messages. However, when m_N is larger than a few units, ABRADE$^+$ outperforms FCFS in both scenarios, though the performance gap is less marked in the ZB scenario for which the feedback costs are smaller.

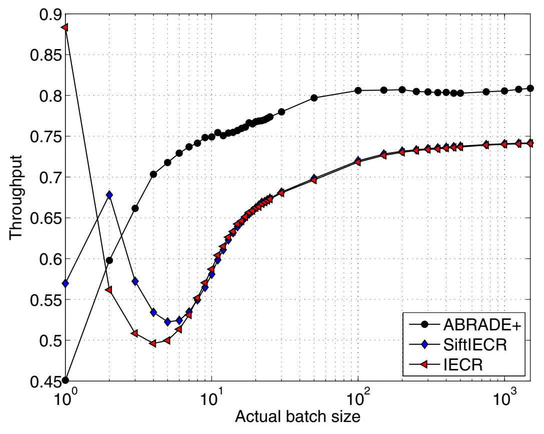

Fig. 9. Throughput $\lambda(n)$ of ABRADE$^+$, Sift/IECR, and IECR as a function of the batch size n, in WF scenario.

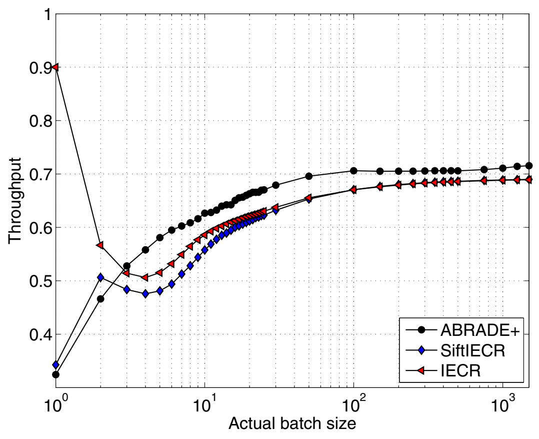

Fig. 10. Throughput $\lambda(n)$ of ABRADE$^+$, Sift/IECR, and IECR as a function of the batch size n, in ZB scenario.

B. Unknown batch size

Here we consider the case where the inquirer does not have *any* prior knowledge about the batch multiplicity, not even in a statistical sense. In this condition, the parameters w_0 and p of ABRADE$^+$ are set under the *arbitrary* supposition that $P_N^{\circ}(\cdot)$ is uniform in the (integer) interval $[0, 2\mathrm{m}_N^{\circ}]$, with $\mathrm{m}_N^{\circ} = 50$. We remark that this choice is totally arbitrary, since the size N of the batch can actually have any distribution $P_N(\cdot)$.

We compare ABRADE$^+$ with the other BRAs that perform a dynamic batch size estimate, namely IECR and Sift/IECR. The FCFS algorithm, instead, is not considered since it needs to know the mean batch size beforehand. Without this information, FCFS is outperformed by IECR and Sift/IECR, as proved in [9].

Results have been obtained by varying the batch size n from 1 to 1500. We remark that, conversely to the previous experiments, the value of n is now assumed to be unknown to the algorithms that, hence, always start with the same parameters setting. For each value of n, we run 10000 independent instances of all the algorithms and report the mean achieved throughput $\lambda(n)$ as a function of n. Note that these results are not biased by any specific distribution of the batch size. The throughput achieved by the algorithms for a certain distribution $P_N(\cdot)$ of the batch size can then be found by averaging the results provided in this section over $P_N(\cdot)$, according to (3).

In Fig. 9 and Fig. 10 we report the algorithms throughput $\lambda(n)$ for different values of n, in the WF and ZB scenarios, respectively. Once again, the 99% confidence intervals are not reported being tight-fitting the curves. The results obtained in the two scenarios are qualitatively the same, though the throughput gaps are bigger in the WF scenario than in the ZB scenario, as already observed in Sec. VII-A.

For batches with a single node, IECR algorithm outperforms all the others, being able to solve the batch in just one slot when the other algorithms generally take in some idle slots. However, as soon as the number of nodes in the batch is greater than one, IECR will experience collisions in the very first slots, which determine the sharp throughput loss that can be observed in the range $1 < n \leq 5$. For larger values of the batch size, the throughput of IECR progressively improves, because the algorithm is capable of dynamically adjusting its parameters to the residual batch size before the batch resolution process is completed. As expected, furthermore, Sift/IECR performs better than IECR for small values of the batch size, whereas for large batches the throughput achieved by the two algorithms is practically the same and equal to the asymptotic throughput achieved by FCFS.

From the figures we see that ABRADE$^+$ outperforms all the other algorithms for batches greater than a few units. ABRADE$^+$ throughput grows almost monotonically with the batch size, except in an interval of values after $n = 100$, where the throughput undergoes a small contraction due to the setting $\mathrm{m}_N^{\circ} = 50$. In fact, when the actual batch size n is significantly larger than m_N°, the first resolution rounds will likely experience more collisions than expected, determining a small performance loss. However, for very large batch sizes, ABRADE$^+$ is capable of reaching the optimal regime behavior and recovering the initial performance loss.

VIII. Discussion

In this paper we presented ABRADE and ABRADE$^+$, two algorithms for the resolution of batches with known and unknown size, respectively. Conversely to most conflict resolution algorithms presented in the literature, ABRADE and ABRADE$^+$ waive the immediate feedback paradigm in favor of an aggregate feedback, in order to reduce the overhead. ABRADE implements a framed ALOHA access scheme, whose frame length is dynamically optimized to the residual batch size. ABRADE$^+$ results from the combination of ABRADE with a suitable batch size estimation module.

ABRADE and ABRADE$^+$ have been analyzed, both mathematically and by simulations, and compared against the best performing algorithms in the literature based on the immediate feedback paradigm. The analysis revealed that, by setting the system parameters in accordance with the specifications of common radio standards, such as IEEE 802.11 (WF) and IEEE 802.15.4 (ZB), the throughput achieved by BRAs with immediate feedback is actually lower than predicted by the classical theoretical analysis, which generally neglects or underestimates the cost of feedbacks. The deferred feedback approach adopted by ABRADE and ABRADE$^+$, conversely, yields higher throughput, especially when the feedback costs are comparable to the packet transmission time, as occurs in high speed wireless LAN, whereas the advantage of the deferred feedback approach is more contained in slow speed systems, such as ZB.

ABRADE$^+$ may seem to be more complex than other solutions, such as FCFS or IECR. However, we observe that the complexity is all concentrated at the inquirer, whereas the other nodes only have to perform the same basic operations as for the classical algorithms. Furthermore, knowing the specifications of the wireless technology, the optimal parameters can be computed offline and stored in a table at the inquirer. Alternatively, it is possible to drastically simplify the dynamic computation of the frame length by approximating the optimal value $w_n^\star$ as the asymptotically optimal value $\lceil \mu_\infty n \rceil$. In this way, the frame length adaptation can be performed by a simple rounded multiplication, at the cost of a marginal performance penalty for small batches.

We finally observe that ABRADE$^+$ may be further ameliorated and extended in different ways. For instance, the contention frame may be slid forward in case of runs of idle slots, in order to avoid almost-certain collisions in the last part of the frame. In quasi static scenarios, it is possible to design the estimate module in order to consider the past events, for instance by using a Bayesian approach, thus speeding up the convergence of the estimate to the actual batch size. Increasing the complexity of the algorithm at the nodes side, furthermore, it is also possible to conceive extensions of the approach to more complex scenarios, involving hidden nodes, multi-hop communications, or priority-based service differentiation. Such extensions are left for future investigation.

Appendix

A-I. Characterization of System Parameters in the Reference Scenarios

The data structures and carrier-sense mechanism of WF and ZB are similar and will be described in a unified manner. Let t_{BCK} denote the reference idle (backoff) slot time. Furthermore, let T_{PCK} be the transmission time of a data packet, including PHY and MAC headers, and payload. In case of unacknowledged transmissions, successive data frames have to be separated by (at least) a certain Inter Frame Space (IFS), here denoted by t_{IFS}. In case of acknowledged transmission, instead, an ACK frame of duration T_{ACK} shall be returned within a time interval $t_{ACK} < t_{IFS}$ after the packet reception. Moreover, an IFS shall follow any ACK frame transmission. If a valid ACK is not received within a certain timeout t_{TO} the transmission is assumed to be collided and a new data frame can be transmitted immediately after. With reference to the system model defined in Sec. II, the system parameters can thus be set as follows

$$T_{data} = T_{PCK} + t_{IFS}\,; \quad \beta = \frac{t_{BCK}}{T_{data}}\,;$$
$$\phi_i = 0\,; \quad \phi_s = \frac{T_{ACK} + t_{ACK}}{T_{data}}\,; \quad \phi_c = \frac{t_{TO} - t_{IFS}}{T_{data}}\,. \tag{A-1}$$

The probe message can be realized by using a standard data frame, whose payload is structured in order to carry the required system parameters. We suppose that a fixed number of bits, that we set to 24 bits, are used to encode ABRADE$^+$ parameters, including the contention frame size w and the contention probability p to be used in the upcoming resolution round. Note, that p is always equal to one, except at the estimate startup phase during which, though, w is generally small. Therefore, at regime, most of these bits can be used to encode the value of w. The encoding of the aggregate feedback field may consist of a bit-mask in one-to-one correspondence with the slots in the contention frame, in such a way that bits 1 denote successful slots, whereas bits 0 indicate collided and idle slots. Therefore, the number of bits in this field is equal to the length of the previous contention frame.[8] In our simulations, we conservatively set the length of the probe message to

$$\beta_p = h_0 + b_p w\,,$$

where h_0 accounts for the fixed duration of the probe message (PHY and MAC packet headers, fixed payload subfields, IFS), whereas $b_p = \frac{1}{RT_{data}}$ is the normalized bit transmission time, being R the transmission rate of the payload field. The resulting parameters for the WF and ZB scenarios are reported in Tab. II.

A-II. Derivation of ABRADE's Asymptotic Throughput

To determine an analytical expression of $\lambda_{\max}$, we introduce the error function $e(n)$ such that

$$\mathcal{T}^\star(n) = \frac{n}{\lambda_{\max}} + e(n)\,. \tag{A-2}$$

Hence, $e(n)$ is the difference between the optimal BRI for batches of size n, $\mathcal{T}^\star(n)$, and the BRI that would be achieved if the node resolution occurred at the asymptotic throughput $\lambda_{\max}$. Dividing by n both sides of (A-2), and letting n grow to infinity we get

$$\lim_{n\to\infty} \frac{e(n)}{n} = \lim_{n\to\infty} \frac{\mathcal{T}^\star(n)}{n} - \frac{1}{\lambda_{\max}} = \lim_{n\to\infty} \frac{1}{\lambda(n)} - \frac{1}{\lambda_{\max}} = 0\,, \tag{A-3}$$

where the second equality follows from (1), and the last from (2). Now, replacing $\mathcal{T}(n-s)$ in the right-hand side of (19) with (A-2) we get

$$\begin{aligned}\mathcal{T}^\star(n) &= \mathrm{E}\left[y_{w_n^\star,n}\right] + \sum_{s=0}^{\infty}\left(\frac{n-s}{\lambda_{\max}} + e(n-s)\right)p_{w_n^\star,n}(s)\\ &= \mathrm{E}\left[y_{w_n^\star,n}\right] + \frac{n}{\lambda_{\max}} - \frac{\mathrm{E}\left[S|w_n^\star,n\right]}{\lambda_{\max}} + \sum_{s=0}^{\infty} e(n-s)p_{w_n^\star,n}(s)\,,\end{aligned} \tag{A-4}$$

that, divided by n and plugged in place of $\mathcal{T}^\star(n)/n$ in (A-3), yields

$$\begin{aligned}0 = &\lim_{n\to\infty}\frac{\mathrm{E}\left[y_{w_n^\star,n}\right]}{n} - \frac{1}{\lambda_{\max}}\lim_{n\to\infty}\frac{\mathrm{E}\left[S|w_n^\star,n\right]}{n}\\ &+ \lim_{n\to\infty}\sum_{s=0}^{\infty}\frac{e(n-s)}{n}p_{w_n^\star,n}(s)\,.\end{aligned} \tag{A-5}$$

Now, recalling (18), the first limit in (A-5) can be expressed as

[8]More efficient codings of the aggregate ACK field are possible. For instance, exploiting the sensing capability of nodes, the aggregate ACK field may be compacted by using a bit-map for busy slots only, where bits 1 denote success and 0 collision.

$$\begin{aligned}
&\lim_{n\to\infty}\frac{\mathrm{E}\left[y_{w_n^\star,n}\right]}{n} = \\
&= \lim_{n\to\infty}\frac{\beta_p + w_n^\star\beta_c + \left(\frac{w_n^\star-1}{w_n^\star}\right)^{n-1}\left((\beta-\beta_c)(w_n^\star-1)+n(1-\beta_c)\right)}{n} \\
&= \lim_{n\to\infty}\frac{w_n^\star}{n}\left(\frac{\beta_p}{w_n^\star}+\left(1-\frac{1}{w_n^\star}\right)^n(\beta-\beta_c)+\beta_c\right)+ \\
&\quad \lim_{n\to\infty}\left(1-\frac{1}{w_n^\star}\right)^{n-1}(1-\beta_c) \\
&= \frac{1}{\mu_\infty}\left(b_p + e^{-\mu_\infty}(\beta-\beta_c)+\beta_c\right) + e^{-\mu_\infty}(1-\beta_c)
\end{aligned}$$

where in the last row we have used $b_p = \lim_{n\to\infty}\frac{\beta_p}{w_n^\star}$, $\mu_\infty = \lim_{n\to\infty}\frac{n}{w_n^\star}$, and the well-known limit

$$\lim_{n\to\infty}(1-1/w_n^\star)^n = \lim_{n\to\infty}\left(1-\frac{n/w_n^\star}{n}\right)^n = e^{-\mu_\infty}\ .$$

Replacing $\mathrm{E}\left[S|w_n^\star,n\right]$ in the second limit of (A-5) with the expression in (13) we get

$$\lim_{n\to\infty}\frac{\mathrm{E}\left[S|w_n^\star,n\right]}{n} = \lim_{n\to\infty}\frac{n\left(1-\frac{1}{w_n^\star}\right)^{n-1}}{n} = e^{-\mu_\infty}\ .$$

Finally, the third limit in (A-5) turns out to be zero. In fact, we observe that

$$\sum_{s=0}^{\infty}\frac{e(n-s)}{n}p_{w_n^\star,n}(s) \le \frac{e^*(n)}{n}$$

where $e^*(n) = \max_{k\le n}\{e(k)\}$. The function $e^*(n)$ is, by construction, monotonically non-decreasing in n. Hence, it can either reach a *finite* absolute maximum $e^* = e(n^*)$ for some finite batch size n^*, and maintain this value for any larger n, or grow to infinity with $e(n)$. In the first case we have $\lim_{n\to\infty} e^*(n)/n = \lim_{n\to\infty} e^*/n = 0$. In the second case, $e^*(n)$ will grow to infinity as quickly as $e(n)$, so that, according with (A-3), we have $\lim_{n\to\infty}\frac{e^*(n)}{n} = \lim_{n\to\infty}\frac{e(n)}{n} = 0$.

Putting all the pieces together, (A-5) yields

$$\lambda_{\max} = \frac{\mu_\infty e^{-\mu_\infty}}{b_p+\beta_c+e^{-\mu_\infty}(\beta-\beta_c)+\mu_\infty e^{-\mu_\infty}(1-\beta_c)}\ . \tag{A-6}$$

It now remains to determine the expression of μ_∞ that maximizes the asymptotic throughput $\lambda_{\max}$. Setting to zero the derivative of the right-hand side of (A-6) with respect to μ_∞, we get

$$\mu_\infty = 1 - \frac{\beta_c-\beta}{b_p+\beta_c}e^{-\mu_\infty}\ , \tag{A-7}$$

whose solution is given by (22). Replacing (A-7) into (A-6) we finally obtain (21), which is the asymptotic ABRADE's throughput.

Acknowledgment

The authors would like to thank Dr. Nicola Bui for some useful early discussions on this topic, which motivated us to conduct this study. The research leading to these results has received funding from the Seventh Framework Programme under grant agreement n. 257521, Internet of Things - Architecture (IoT-A) http://www.iot-a.eu/public.

References

[1] M. Rossi, M. Zorzi, and F. H. R. Fitzek, "Link layer algorithms for efficient multicast service provisioning in 3G cellular systems," in *Proc. 2004 IEEE Global Telecommun. Conf.*, vol. 6, pp. 3855–3860

[2] D. R. Hush and C. Wood, "Analysis of tree algorithms for RFID arbitration," in *Proc. 1998 IEEE International Symp. Inf. Theory*, p. 107.

[3] P. Popovski, "Tree protocols for RFID tags with generalized arbitration spaces," in *Proc. 2008 IEEE International Symp. Spread Spectrum Techniques Appl.*, pp. 18–22.

[4] J. Hayes, "An adaptive technique for local distribution," *IEEE Trans. Commun.*, vol. 26, no. 8, pp. 1178–1186, Aug. 1978.

[5] J. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inf. Theory*, vol. 25, no. 5, pp. 505–515, Sep. 1979.

[6] R. Gallager, "Conflict resolution in random access broadcast networks," in *Proc. 1978 AFOSR Workshop Commun. Theory Appl.*, pp. 74–76.

[7] J. Mosely and P. Humblet, "A class of efficient contention resolution algorithms for multiple access channels," *IEEE Trans. Commun.*, vol. 33, no. 2, pp. 145–151, Feb. 1985.

[8] P. Popovski, F. H. P. Fitzek, and R. Prasad, "Batch conflict resolution algorithm with progressively accurate multiplicity estimation," in *Proc. 2004 DIALM-POMC*, pp. 31–40.

[9] ——, "A class of algorithms for collision resolution with multiplicity estimation," *Algorithmica*, vol. 49, no. 4, pp. 286–317, 2007.

[10] M. Molle and G. Polyzos, "Conflict resolution algorithms and their performance analysis," Department of Computer Science and Engineering, UCSD, East Lansing, Michigan, Tech. Rep. CS93-300, July 1993.

[11] D. Klair, K.-W. Chin, and R. Raad, "A survey and tutorial of RFID anti-collision protocols," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 400–421, 2010.

[12] D. Bertsekas and R. Gallager, *Data Networks*, 2nd edition. Prentice-Hall, Inc., 1992.

[13] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan, "Understanding the real-world performance of carrier sense," in *2005 ACM SIGCOMM Workshop Experimental Approaches Wireless Netw. Design Analysis*.

[14] I. Cidon and M. Sidi, "Conflict multiplicity estimation and batch resolution algorithms," *IEEE Trans. Inf. Theory*, vol. 34, no. 1, pp. 101–110, Jan. 1988.

[15] A. G. Greenberg, P. Flajolet, and R. E. Ladner, "Estimating the multiplicities of conflicts to speed their resolution in multiple access channels," *J. ACM*, vol. 34, no. 2, pp. 289–325, 1987.

[16] Y. Tay, K. Jamieson, and H. Balakrishnan, "Collision-minimizing CSMA and its applications to wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1048–1057, Aug. 2004.

[17] A. Zanella, "Estimating collision set size in framed slotted aloha wireless networks and RFID systems," *IEEE Commun. Lett.*, vol. 16, no. 3, pp. 300–303, Mar. 2012

[18] G. Pierobon, A. Zanella, and A. Salloum, "Contention-TDMA protocol: performance evaluation," *IEEE Trans. Veh. Technol.*, vol. 51, pp. 781–788, 2002.

[19] F. C. Schoute, "Dynamic frame length ALOHA," *IEEE Trans. Commun.*, vol. COM-31, no. 4, pp. 565–568, Apr. 1983.

[20] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the Lambert*W* function," *Advances in Computational Mathematics*, pp. 329–359, 10.1007/BF02124750.

[21] M. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," in *Proc. 2006 MobiCom International Conf. Mobile Comput. Netwo.*, pp. 322–333.

[22] Q. Tong, X. Zou, and H. Tong, "Dynamic framed slotted aloha algorithm based on Bayesian estimation in RFID system," in *Proc. World Congress Comput. Science Inf. Eng.*, vol. 1, pp. 384–388.

Andrea Zanella Andrea Zanella is Assistant Professor at the Department of Information Engineering (DEI), University of Padova (ITALY). He received the Laurea degree in Computer Engineering in 1998, from the same University, and the Ph.D. degree in Electronic and Telecommunications Engineering, in 2002. Before that, he spent nine months as visiting scholar at the Department of Computer Science of the University of California, Los Angeles (UCLA), where he worked with Prof. Mario Gerla on Wireless Networks and Wireless Access to Internet.