Analysis of an Identifier Splitting Algorithm Combined with Polling for Contention Resolution in a Wireless ATM Access Network

B. Van Houdt, C. Blondia

University of Antwerp
Department of Mathematics and Computer Science
Performance Analysis of Telecommunication Systems Research Group
Universiteitsplein, 1, B-2610 Antwerp - Belgium
{vanhoudt,blondia}@uia.ua.ac.be
http://win-www.uia.ac.be/u/pats/

Abstract

In this paper a contention resolution scheme for an uplink contention channel in a wireless ATM access network is presented. The scheme consists of a tree algorithm, namely the identifier splitting algorithm (ISA), combined with a polling scheme. Initially ISA is used, but at a certain level of the tree, the scheme switches to polling of the stations. This scheme is further enhanced by skipping a few levels in the tree when starting the algorithm (both in a static and a dynamic way) and by allowing multiple instants simultaneously. An analytical model of the system and its variants leads to the evaluation of its performance, by means of the delay density function and the throughput characteristics. This model is used to investigate the influence of the packet arrival rate, the instant at which the ISA schemes switches to polling, the starting level of the ISA scheme and the use of multiple instances on the mean delay, the delay quantiles and the throughput.

1 Introduction

Due to the rapid development of powerful high performance portable computers and other mobile devices, there is an increasing interest in wireless communication systems, in particular for Local Area Networks (e.g. in an office environment). These wireless LANs need to be connected in a seamless fashion to the fixed network. For these fixed networks, the Asynchronous Transfer Mode (ATM) has been standardized as the transmission and switching technology supporting the Quality of Service (QoS) requirements of different service categories in an efficient way. In order to offer end-to-end ATM service, an ATM based transport architecture for an integrated services wireless network has to be defined. The ability to support ATM services over a wireless link, will heavily depend on the definition of the Medium Access Control protocol, needed to arbitrate

access to the shared radio medium.
The importance of these technological challenges is also made evident by the number of projects within the European program ACTS concerning this topic (e.g. MEDIAN [14], Magic WAND [7], SAMBA [9], AWACS, AMUSE) and a number of Medium Access Control (MAC) protocols have been proposed for such systems, such as MASCARA [8], PRMA [5], DSA++ [9], DQRUMA [6] and others [3, 4, 17, 13, 15].
The system that is considered in this paper has the following characteristics (see Figure 1). Consider a cell in a wireless ATM network, consisting of a base station (BS) serving a finite set of mobile stations (MS) by means of a shared radio channel. The BS is connected to an ATM switch which supports mobility, realizing access to the wired ATM network. ATM PDUs (i.e. ATM cells, but to avoid confusion with the notion of cell in a wireless network, we use ATM PDU) arriving at the BS are broadcasted downlink. The ATM PDUs originating from an MS share the radio medium using a MAC protocol. The access technique is Time Division Multiple Access (TDMA) and Frequency Division Duplex (FDD). Each MS in a cell receives a unique address that is used by the MAC layer in the BS.
The MAC protocols considered in the above references have a centralized control located in the BS. The uplink channel is used by the MSs to inform the BS about their bandwidth needs (through requests) and to transmit ATM-PDUs. The downlink channel is used for acknowledgments, information about the permission to use the uplink channel (permits) and ATM-PDUs. Both the information on the uplink and the downlink channels is grouped into fixed length frames, leading to significant reductions in the power consumption. The requests issued by the MSs are usually piggybacked with the uplink ATM-PDUs that have already been scheduled. Unfortunately, such a piggybacking schemes fail for new terminals entering the network and terminals which become active after having remained silent for a period of time or have sudden increases in their uplink traffic. Therefore, an additional uplink contention channel is provided to allow these MSs to inform the BS about their bandwidth needs. A dynamic portion of each fixed length frame is assigned to this contention channel.

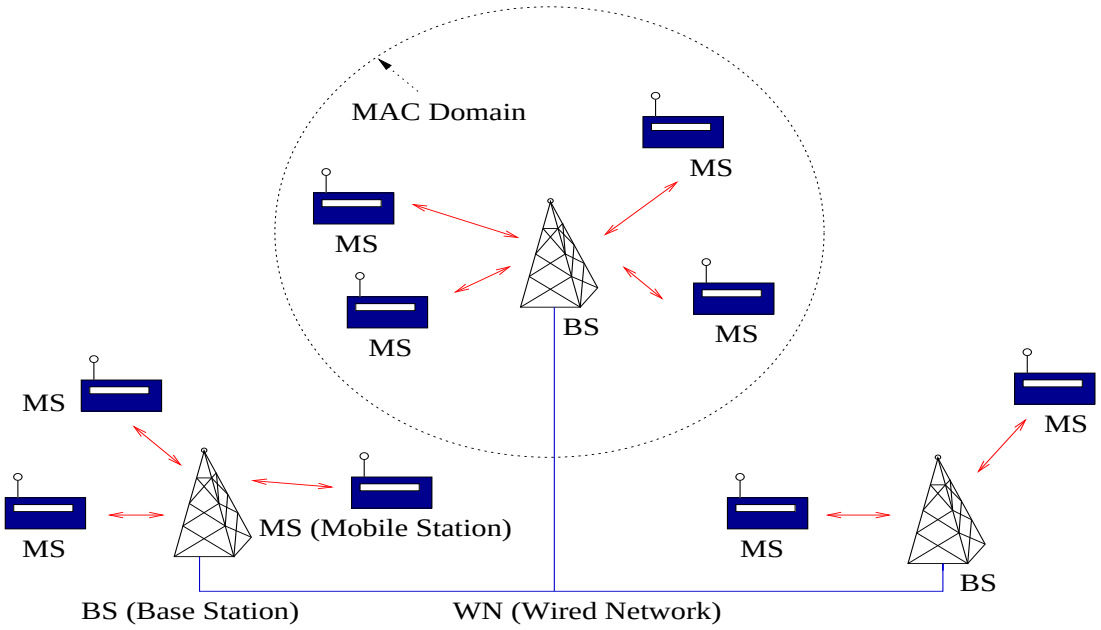

Figure 1: *Reference configuration of the system*

We propose to use a contention resolution scheme based on a tree algorithm, namely the

Identifier Splitting Algorithm, which was developed within the MBS project [11, 10, 12]. In order to improve its performance, the ISA is combined with a polling scheme. The resulting scheme is further enhanced by skipping a few levels in the tree when starting the algorithm, both in a static and dynamic way and by allowing multiple instants simultaneously. The throughput and delay density functions are analytically evaluated. The application of the analysis on numerical examples illustrates the influence of the different system parameters on the delay and throughput characteristics.
The structure of the paper is as follows. Section 2 presents a description of the Identifier Splitting Algorithm (ISA) combined with polling. A scheme where the first levels are skipped is also proposed, and the use of multiple instants of the algorithm is discussed. In section 3, the analytical model to evaluate the performance of the protocol is presented. Using the results of the performance analysis, some numerical examples are given in section 4, and conclusions are drawn in section 5.

2 The Contention Resolution Scheme

2.1 The Identifier Splitting Algorithm (ISA)

The Identifier Splitting Algorithm is based on the well known tree algorithm [2, 1, 16] and was proposed by Petras in [11, 10, 12]. A contention cycle (CC) consists of a number of consecutive upstream frames during which the contention is solved for all requests present in the MSs at the beginning of the cycle. Requests that intend to use the contention resolution scheme generated by the MSs during a CC, have to wait for participation till the start of the next CC.
In the first frame of a cycle, one contention slot is available. Any MS having a request ready, at the start of this CC, makes use of this slot. Next the BS checks whether the transmission was successful and informs the MS(s) that were involved in the scheme accordingly in the next downstream frame using a feedback field. Two situations are possible:

(i) An MS sending in this slot was successful. In this case the MS will eventually be granted a permit to send an upstream cell by the BS.

(ii) The transmission was not successful, i.e. a collision occurred. In this case, the next (second) frame of the CC provides 2 contention slots. Based on the first bit of their MAC addresses, as opposed to the classical coin flip, the MSs that are involved split up into two distinct sets. An MS belonging to the first set uses the first slot to attempt a retransmission, while the second slot is used by the MSs belonging to the second set.

This process of generating two slots in the next frame for each slot in which a collision occurred, is repeated frame after frame, each time using the next bit of the MAC address in case of a collision. Thus during the i-th frame of a CC, two MSs can only collide if their MAC addresses have the same $i-1$ first bits. Therefore, provided that the address that uniquely identify an MS, is n bits long, all collisions are always resolved in $n+1$ frames. Also notice that for every frame the number of contention slots equals twice the number of collisions of the previous frame. To clarify all this, Figure 2 shows an example of a CC with 6 participants. In this figure **CO** refers to a collision, **SU** to a success and **EM** to an empty slot. The MAC addresses of the successful MSs are added to the corresponding slot.

2.2 The Identifier Splitting Algorithm Combined with Polling

One of the attractive features of the Identifier Splitting Algorithm, apart from its dynamic nature, is that as the scheme is being resolved, the BS obtains more and more knowledge about the MSs

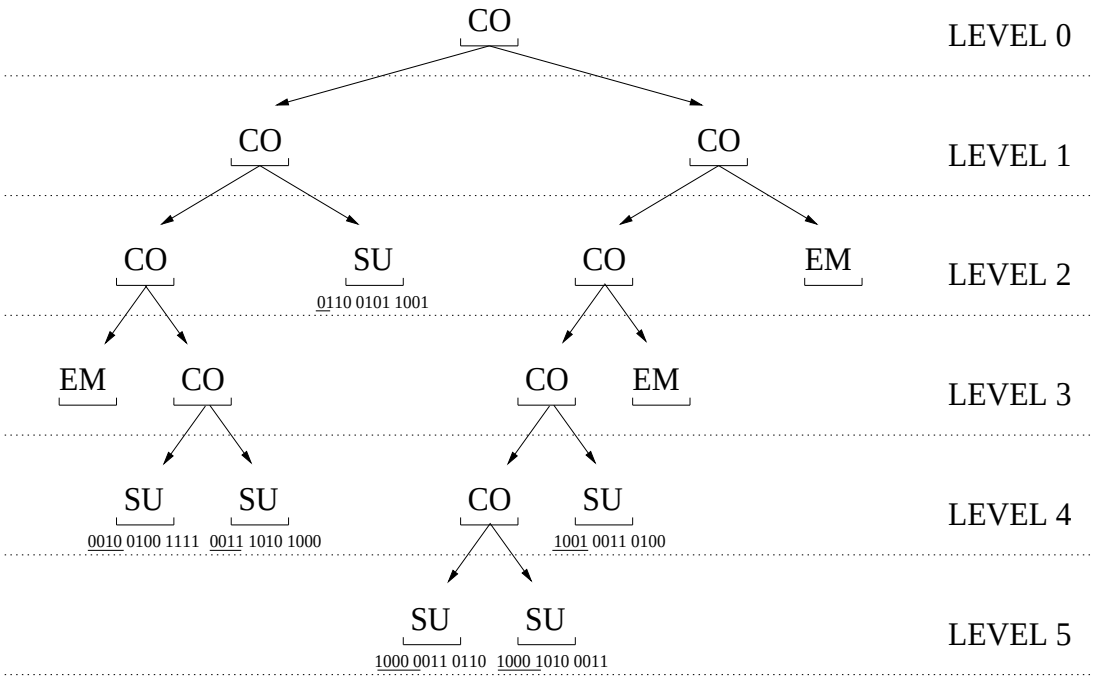

Figure 2: *Demonstrating ISA*

that are still competing. For example, if the BS notices that the tree at level i (the top of the tree is referred to as level 0, see Figure 2) contains k collisions and the MAC-addresses are n bits long then the BS concludes that the remaining competing MSs can only have $k2^{n-i}$ possible addresses. This follows from the fact that each slot at level i corresponds to 2^{n-i} addresses. This information can be used by the BS to take a decision whether to continue to use the ISA protocol or to switch to polling. The basic idea here is that when the size of the remaining MAC address space becomes smaller than some predefined value, say N_p, the protocol switches to polling. Polling, in this context, means that in the next frame, one slot is provided for each address in the remaining address space.
In the next section we indicate how the MSs know which slot to use and when to switch to a polling scheme.

2.3 MS Behaviour

The behaviour of the different MSs, located within the observed cell, is described in Figure 3 by means of a flow chart.

As long as an MS is able to piggyback its requests, it remains in the *piggybacked* state. When an ATM-PDU generated by an MS, finds the queue with ATM PDUs waiting for transmission empty, then a transition occurs to the state *blocked*. There it remains until the current CC is solved, checking the feedback field at the beginning of every frame. The feedback is given by a set of bits, one for each contention slot, where a zero indicates a success (or an empty slot) and a one a failure. Notice that one feedback bit is sufficient as we do not take capture effects into account. Once the current CC has ended, i.e. all feedback bits equal zero, the two parameters 'LEVEL' and 'POS' are initialized. They have the following function:

- LEVEL : indicates the current level of the CC, and therefore its value is incremented by one at the start of each frame during a CC,

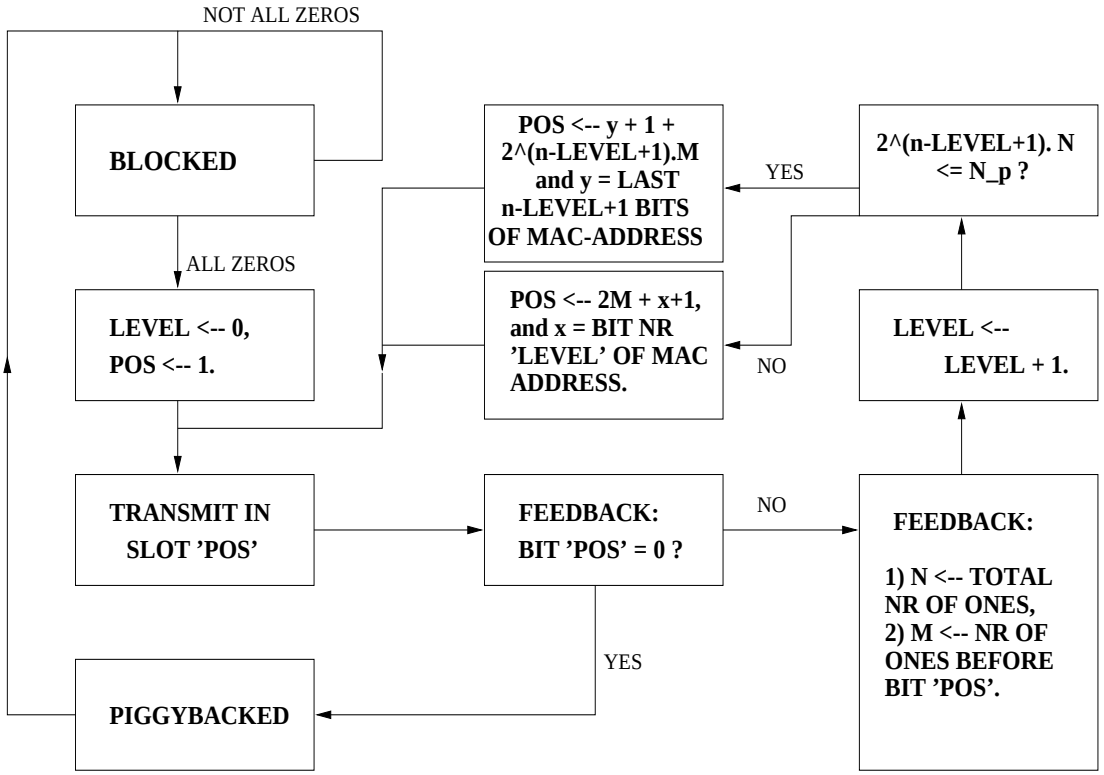

Figure 3: *The Flow Chart of an MS*

- POS : is a variable that holds the number of the contention slots to be used by the MS (the slots are numbered starting from one).

After initialization, the *transmission* state is entered. While in this state, a transmission will take place in slot number 'POS' and the result is found by checking the corresponding feedback bit. If successful we return to the *piggybacked* state, otherwise the MS sets the parameters N and M, and increments 'LEVEL' by one. Next, the MS checks to see whether a switch to the polling scheme is made, and depending on this result assigns a new value to 'POS'. Finally the MS returns to the *transmission* state and this routine is repeated until a successful transmission occurs. Checking to see whether, at level $i+1$, a switch to polling is made simply consists of calculating the size of the remaining address space and comparing the result with N_p. Due to the fact that a slot at level i corresponds with 2^{n-i} addresses, the remaining address space is found by multiplying the number of collisions at that level i, by 2^{n-i}.

2.4 Skipping the First Few Levels

In the previous sections a CC was always started with just one contention slot at level zero of the tree. In order to investigate the impact of this fact on the performance characteristics, we consider other starting levels. Hence, instead of starting with just one contention slot in the first frame, we provide more than one slot during the first frame of a CC.
At first the starting level is fixed at a predefined value S_l. It is expected that this has a positive impact on the delay. Apart from that, the throughput might improve in case of high loads [12]. Unfortunately as will be shown in the numerical results in section 4, this results in some extra throughput losses during low load periods. To solve this we propose a scheme that changes the starting level dynamically, between level S_{min} and S_{max}, depending on the length of the previous

CC. To make this decision, the system load ρ is not taken into account, as this value is hard to measure or predict in real systems.
The starting levels are defined using the following two threshold values B_l and B_m. Suppose that at some point in time the starting level equals S_l and L is the length of this CC. Then the new starting level S_l' obeys the following equation

$$S_l' = \begin{cases} \max(S_l - 1, S_{min}) & L \leq B_l \\ S_l & B_l < L < B_m \\ \min(S_l + 1, S_{max}) & L \geq B_m \end{cases} \quad (1)$$

Clearly all MSs, wanting to access the contention channel, need to be aware of the current starting level. We suggest that this knowledge is broadcasted

by the BS at the start of every CC. Therefore it is not necessary for all MSs, including those that do not use the contention channel, to keep track of the lengths of the CCs.

2.5 Multiple Instances of ISA

In what follows, we demonstrate by means of an example how multiple instances of the ISA protocol with a fixed starting level $S_l \geq 1$ can be introduced. For this example (see Figure 4), the starting level S_l is fixed at one. Here at level 3, all collisions in the right-hand side of the tree are resolved. Suppose that during these 3 frames a number of MSs, not necessarily participating in this CC and some with the first bit of their MAC address equal to one, have generated a new request. Then according to the previous sections they have to wait until the start of the next CC, that is until the end of frame 5.
Alternatively, the BS may initiate a new instance of the ISA protocol, used by all MSs belonging to the second half of the tree, thereby creating a second instance of the ISA protocol. The first instance is used by all MSs with the first bit of their address equal to 0, the second half is devoted to the other MSs, i.e. with a MAC address that starts with 1.
In general, the ISA protocol with starting level S_l can be uncoupled to form 2^{S_l} different instances of ISA, where each instance corresponds with a partitioning of the address space. The MS behaviour is very similar as before, thus no extra complexity is added.

Another advantage of this method is that the contention slots are spread more uniformly over consecutive frames, as the different instances are not necessarily in phase, i.e. the tops of the different trees might occur in different frames. The disadvantage of uncoupling is that we can not decrease any longer the starting level, dynamically, below level S_l.
Although it is possible to combine multiple instances and polling, this is beyond the scope of this paper.

3 Performance Analysis

3.1 The Analytical Model

For this analysis we assume that each MS has at most one active connection. We define n as the size of the MAC-addresses (in bits). The number of MSs located within the reach of the BS is 2^n, i.e. all MAC addresses are utilized.
We assume that the aggregate traffic generated by all MSs, on the uplink contention channel, has a Poisson distribution with a mean of λ requests per frame. As the number of MSs is finite and equals 2^n, the number of requests, generated during a CC, should never exceed 2^n. Therefore we drop, at random, some of the arrivals if this value is exceeded (for $x > 2^n$ arrivals, we drop $x - 2^n$ arrivals). In this way, we assure that the requests arrive in a uniform way during a CC. Hence,

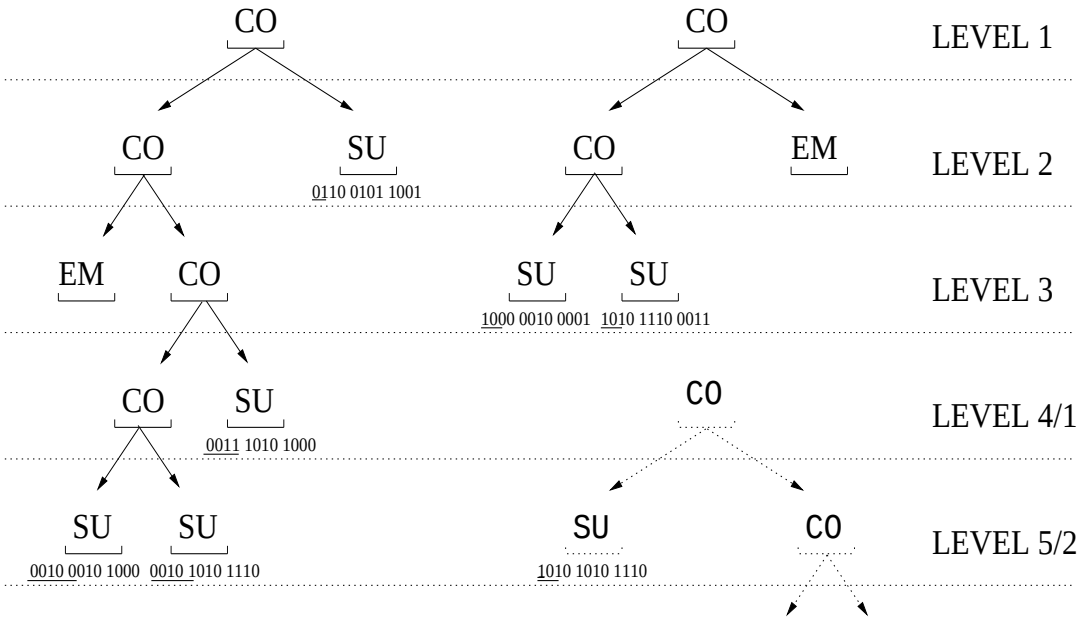

Figure 4: *Creating Multiple Instances of ISA*

define the random variable I_i as the number of requests generated during a CC, consisting of i frames, as

$$\begin{aligned} P[I_i = k] &= \frac{(\lambda i)^k}{k!} e^{-\lambda i}, k < 2^n \\ P[I_i = 2^n] &= \sum_{k \geq 2^n} \frac{(\lambda i)^k}{k!} e^{-\lambda i}. \end{aligned}$$

Notice that we do not need to consider bursty input traffic since we are observing the access channel used by an MS to transmit a first request after a period of silence. In real-life systems the following holds with respect to the number of MSs participating, and their addresses.

- MSs that were successful during the last frame of a CC, will never participate in the next CC.
- Participating MSs, regardless of the frame in which they were successful, are less likely to take part in the next CC as opposed to those that did not participate at all.

To make the system analytically tractable, both these remarks are ignored. Thus the addresses of the MSs taking part in the scheme at the beginning of a CC are uniformly distributed over the complete address space and their number is distributed according to a Poission distribution, where the mean depends on the length of the previous CC.

The following random variables will be used in the sequel of this section.

- X_c, resp. X_a, denotes the number of contenders or participants in a CC for the ISA protocol without resp. with polling.
- R_c, resp. R_a, denotes the level at which the CC is resolved (i.e. the number of frames needed minus one) and this again for the ISA scheme without resp. with polling.

- $C_i^{(c)}$, resp. $C_i^{(a)}$, denotes the number of collisions at level i for both protocols. These variables range from 0 to 2^i.
- P_a denotes the level at which we poll for the ISA scheme with polling. If the scheme is solved without polling we let P_a be equal to $n+1$.

Furthermore we use the symbol C_r^n to denote the number of different possible combinations of r from n different items.

3.2 The Delay Analysis

3.2.1 The Identifier Splitting Algorithm

(A) We start by studying the random variable R_c conditioned on X_c. Notice that at level i the address space is split into 2^i equal parts of size 2^{n-i}. For the scheme to be collision free at level i we can only allow one participating MS in each subspace. This results in

$$P[R_c \leq i \mid X_c = k] = \frac{2^{(n-i)k} C_k^{2^i}}{C_k^{2^n}}. \tag{2}$$

This can be proven by noticing that $P[R_c \leq i \mid X_c = k] = P[R_c \leq i \mid X_c = k-1] * 2^{n-i} * (2^i - (k-1))/(2^n - (k-1))$ using induction on k. An alternative proof is based on the multivariate hypergeometric distribution. By subtraction we obtain $P[R_c = i \mid X_c = k]$, which is denoted as $p_c(k, i+1)$ (we write $i+1$ to indicate the number of frames used).

(B) Let us now focus on X_c. Clearly X_c is the steady-state vector of the Markovian process $(X_n^{(c)})_n$, where $X_n^{(c)}$ denotes the number of contenders during the n-th CC. Due to (A),

$$t_c(k,j) \stackrel{\text{def}}{=} P[X_{n+1}^{(c)} = j \mid X_n^{(c)} = k] = \sum_{t=1}^{n+1} \frac{(\lambda t)^j e^{-\lambda t}}{j!} p_c(k,t), \tag{3}$$

for $0 \leq j \leq 2^n - 1$. When $j = 2^n$ we assign the remaining probability mass. X_c is then found by solving the eigenvector problem. Applying the definition of the expected value gives us the mean number of participants $E[X_c]$ in a CC.

(C) Before we can calculate the delay we still need to make the following observation. Assume a random arrival in a CC, then we need to know the probability that there are k contenders in this CC and that there will be l in the next CC. We denote $X_{cur}^{(c)}$ and $X_{next}^{(c)}$ as the number of participants in these two CCs. Some straightforward reasoning shows that the following relationship between $X_{next}^{(c)}$, $X_{cur}^{(c)}$ and X_c holds

$$P[X_{next}^{(c)} = l] = \frac{P[X_c = l] l}{E[X_c]} \tag{4}$$

and

$$P[X_{cur}^{(c)} = k] = \frac{\sum_{j=1}^{2^n} P[X_c = k] t_c(k,j) j}{E[X_c]} \tag{5}$$

where $t_c(k,j)$ was defined in (3).

Combining (A), (B) and (C) Having done this we can calculate the mean delay. Clearly the delay consists of two parts. The first is the time until the start of the next CC and the second is the number of frames needed until our tagged request is successful. Using expression (5) and knowing that the arrivals are distributed uniformly within a CC (see section 4.1), the expected value for the first part equals

$$E[RCL] \stackrel{\text{def}}{=} E[\text{remaining cycle length}] = \sum_{i=1}^{n+1} \sum_{k} P[X_{cur}^{(c)} = k] \frac{i\ p_c(k,i)}{1 + E[R_c \mid X_c = k]}\ i/2. \tag{6}$$

By definition of the expected value the second part equals

$$E[FNBS] \stackrel{\text{def}}{=} \sum_{i=0}^{n} \sum_{k \geq 1} P[X_{next}^{(c)} = k](i+1)(\mathcal{F}_c(i,k) - \mathcal{F}_c(i-1,k)), \tag{7}$$

where $\mathcal{F}_c(i,k)$ denotes the probability that a tagged request is successful at or before level i given that there where $k-1$ other contenders ($\mathcal{F}_c(-1,k)$ is zero in the expression above). Again we can prove by induction that

$$\mathcal{F}_c(i,k) = \frac{C_{k-1}^{2^n - 2^{n-i}}}{C_{k-1}^{2^n - 1}}. \tag{8}$$

Adding $E[RCL]$ and $E[FNBS]$ results in the mean delay.

The delay density function Using **(A), (B)** and **(C)**, it is easy to find the delay density function $D_c(x)$ (with x between 1 and $2 * (n+1)$). This function is the following step function

$$D_c(x) = \sum_{s=1}^{\lfloor x \rfloor} \sum_{j=\lceil x \rceil - s}^{n+1} \sum_{l=1}^{2^n} \frac{\mathcal{F}_c(s-1,l) - \mathcal{F}_c(s-2,l)}{j}\ \mathcal{G}_j(l) \cdot$$
$$\cdot \left(\sum_{k=0}^{2^n} \frac{j\ P[R_c = j-1 \mid X_c = k]}{1 + E[R_c \mid X_c = k]} P[X_{cur}^{(c)} = k] \right) \tag{9}$$

where $\mathcal{G}_j(l) = \frac{(\lambda j)^{l-1}}{(l-1)!} e^{-\lambda j}$ for $l < 2^n - 1$ and $\mathcal{G}_j(2^n) = \sum_{j \geq 2^n - 1} \frac{(\lambda j)^{l-1}}{(l-1)!} e^{-\lambda j}$. In (9) s denotes the number of transmissions (including the successful transmission) a tagged request needs, j refers to the length (in frames) of the CC in which our tagged request is generated. Finally $l-1$ equals the number of other competitors apart from our tagged one.

3.2.2 The Identifier Splitting Algorithm combined with Polling

In this section we will follow the same lines of reasoning and therefore we start by studying $P[R_a \leq i \mid X_a = k]$.

(A') Two cases can be considered. First the CC might be solved before level i or at level i due to polling, secondly it might be solved at level i without a switch to polling. Thus we get

$$P[R_a \leq i \mid X_a = k] = P[R_a \leq i-1 \cup P_a \leq i \mid X_a = k] +$$
$$P[R_a = i \cap P_a > i \mid X_a = k]. \tag{10}$$

The first probability is discussed in **(A1')**, the second in **(A2')**.

(A1') We calculate the complementary probability mass. Clearly by definition of the polling mechanism we have

$$P[R_a \geq i \cap P_a > i \mid X_a = k] = P[C_{i-1}^{(a)} > \left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor \mid X_a = k]. \tag{11}$$

The right-hand side is found using the following relationship

$$P[C_{i-1}^{(a)} = \left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor + x \mid X_a = k] = P[C_{i-1}^{(c)} = \left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor + x \mid X_c = k] \tag{12}$$

for $x \geq 1$, but not necessarily for $x \leq 0$. To prove this we must show that having $\left\lfloor \frac{N_p}{2^{n-i+1}} \right\rfloor + x$ collisions at level $i-1$ given k contenders for one scheme, implies the same for the other scheme. Clearly if the polling scheme had this number of collisions (at level $i-1$ given k participants), polling did not occur before and thus we have the same effect for the non-polling scheme. On the other hand if the ISA scheme without polling results in that many collisions, polling again is not an issue as the remaining address space is too large and only decreases in size as the level increases. This observation motivates us to study $C_i^{(c)}$ conditioned on X_c in more detail.
Our main objective here is to find the probability that we have exactly l collisions at level i given k participants. Although these values are easy to describe mathematically by an extremely large sum of multivariate hypergeometic probabilities, this is of no practical use due to high computational complexity. A more complicated but appropriate way would be to apply the Inclusion-Exclusion Principle. This method sometimes tends to give numerical problems for large values of n, and therefore we propose the following variation on the Inclusion-Exclusion Principle (where the first equality is a consequence of (2)):

$$s(i, 2^i, k) = \frac{2^{(n-i)k} C_k^{2^i}}{C_k^{2^n}}, \tag{13}$$

$$s(i, l, k) = C_l^{2^i} \sum_{l_1=0}^{l} 2^{(n-i)l_1} \frac{C_{l_1}^{l} C_{k-l_1}^{2^n - l2^{n-i}}}{C_k^{2^n}} - \sum_{x=1}^{2^i - l} C_l^{l+x} s(i, l+x, k), \tag{14}$$

where $s(i, l, k) = P[C_i^{(c)} = 2^i - l \mid X_c = k]$. This concludes **(A1')**

(A2') In this case each collision at level $i-1$ involves only two MSs, otherwise it cannot be solved at level i. The probability that such a collision is solved, at level i, clearly equals $\frac{2^{n-i}}{2^{n-i+1}-1}$. Thus by means of the multivariate hypergeometric distribution we get

$$P[R_a = i \cap P_a > i \mid X_a = k] = \sum_{u=u_{i-1}+1}^{\lfloor \frac{k}{2} \rfloor} 2^{(n-i+1)(k-2u)} \frac{\left(C_2^{2^{n-i+1}}\right)^u C_u^{2^{i-1}} C_{k-2u}^{2^{i-1}-u}}{C_k^{2^n}} \left(\frac{2^{n-i}}{2^{n-i+1}-1} \right)^u, \tag{15}$$

where u_i denotes $\left\lfloor \frac{N_p}{2^{n-i}} \right\rfloor$.

(B',C') As before we define $p_a(k, i+1) = P[R_a = i \mid X_a = k]$. Steps **(B')** and **(C')** are very analogue to **(B)** and **(C)**. As for the calculation of the mean delay, we still need to find $\mathcal{F}_a(i, k)$ i.e. the probability that a tagged request is successful at or before level i given that we had k contenders (for the ISA scheme with polling).
We denote $u_{i-1} + 1$ as v_i, thus $v_i = 1 + \lfloor \frac{N_p}{2^{n-i+1}} \rfloor$. In **(A1')** it was argued that the event $C_{i-1}^{(c)} \geq v_i$

is the same as $C_{i-1}^{(a)} \geq v_i$, when conditioned on X_c resp. X_a, which in its turn coincides with $P_a > i \cap R_a \geq i$. Therefore we have

$$\mathcal{F}_a(i,k) = P[R_a \leq i-1 \cup P_a \leq i \mid X_a = k] + \sum_{s \geq v_i} P[R_t \leq i \cap C_{i-1}^{(c)} = s \mid X_c = k], \tag{16}$$

where R_t denotes the level at which our tagged request is successful. This first probability was found in **(A1')**. The second one is calculated using the equations (13) and (14) as follows. We define $t(i,s,k)$ as $P[R_t \leq i \cap C_{i-1}^{(c)} = 2^{i-1} - s \mid X_c = k]$. Then we get (where the first equation is a consequence of (2))

$$t(i, 2^{i-1}, k) = \frac{2^{(n-i+1)k} C_k^{2^{i-1}}}{C_k^{2^n}} \tag{17}$$

$$\begin{aligned} t(i,s,k) &= C_s^{2^{i-1}} \sum_{l_1=0}^{s} 2^{(n-i+1)l_1} \frac{C_{l_1}^s C_{k-l_1}^{2^n - s2^{n-i+1}}}{C_k^{2^n}} \\ &\quad \times \left(\frac{l_1}{k} + \left(1 - \frac{l_1}{k}\right) \frac{C_{k-l_1-1}^{2^n - s2^{n-i+1} - 2^{n-i}}}{C_{k-l_1-1}^{2^n - s2^{n-i+1} - 1}} \right) - \sum_{x=1}^{2^{i-1}-s} C_s^{s+x} t(i, s+x, k). \end{aligned} \tag{18}$$

With these values it is straightforward to find the second term of expression (16) and thus we have an expression for the mean delay. When we look at the delay density function we can use the same formula as we did in the basic scheme (where the superindex a is used instead of c). This concludes the delay analysis.

3.3 The Throughput Analysis

3.3.1 The Identifier Splitting Algorithm

In this section we determine the throughput for the basic scheme. First we define two more sets of random variables $S_i^{(c)}$ and $S_i^{(a)}$, being the number of slots used at level i by both schemes. From the foregoing we already obtained $P[X_c = k]$ and thus the throughput T_c is found as

$$T_c = \frac{E[X_c]}{\sum_{k=0}^{2^n} P[X_c = k] * E[\sum_i S_i^{(c)} \mid X_c = k]}. \tag{19}$$

We could calculate the expected number of slots in this formula as was done in [12] (using a strong recursive scheme). Still it is possible to get the same results using a more direct approach as follows. First notice that

$$E[\sum_i S_i^{(c)} \mid X_c = k] = 1 + \sum_{i=1}^{n} E[S_i^{(c)} \mid X_c = k]. \tag{20}$$

On the other hand we know that the expected number of slots at level i equals twice the expected number of collisions at level $i-1$. While the expected number of collisions at level i matches

$$E[C_i^{(c)} \mid X_c = k] = 2^i \left(1 - \frac{C_k^{2^n - 2^{n-i}}}{C_k^{2^n}} - 2^{n-i} \frac{C_{k-1}^{2^n - 2^{n-i}}}{C_k^{2^n}} \right). \tag{21}$$

Hence we have found the throughput results without using any form of recursion.

3.3.2 The Identifier Splitting Algorithm combined with Polling

Again since we already know the probabilities $P[X_a = k]$ from the delay analysis, it is sufficient to find $E[\sum_i S_i^{(a)} \mid X_a = k]$. Unfortunately this is not as straightforward as one might expect. We start in a similar manner as above. The expected number of slots used equals the sum of the expected number of slots used at each level. By definition of the adapted scheme we have that

$$E[S_i^{(a)} \mid X_a = k] = P[P_a = i \mid X_a = k]\ E[S_i^{(a)} \mid X_a = k \cap P_a = i] + \ldots$$
$$P[P_a > i \cap R_a \geq i \mid X_a = k]\ E[S_i^{(a)} \mid X_a = k \cap P_a > i \cap R_a \geq i], \tag{22}$$

by observing that the expected number of slots is zero if $R_a \leq i-1$. The second probability was obtained in **(A1')**, the first one is calculated as $P[R_a \leq i-1 \cup P_a \leq i \mid X_a = k]$ minus $P[R_a \leq i-1 \mid X_a = k]$, two results that were also obtained in **(A')**. It remains to compute both expected values (for $i \geq 2$ since $S_0^{(a)}$ and $S_1^{(a)}$ are trivial to obtain). They are discussed in **(D')** and **(E')**.

(D') First consider $E[S_i^{(a)} \mid X_a = k \cap P_a > i \cap R_a \geq i]$. In this case the number of slots used at level i is twice the number of collisions of the previous level. Also in **(A1')** it was shown that the event $P_a > i \cap R_a \geq i$ is the same as $C_{i-1}^{(c)} \geq v_i$, again when conditioned on X_a resp. X_c. Thus it is sufficient to find

$$E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-1}^{(c)} \geq v_i].$$

This is done using the definition of the expected value combined with (12)

$$E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-1}^{(c)} \geq v_i] = v_i + \sum_{s=0}^{2^{i-1}-v_i} s \frac{P[C_{i-1}^{(c)} = s + v_i \mid X_c = k]}{P[C_{i-1}^{(c)} \geq v_i \mid X_c = k]}, \tag{23}$$

where we applied the following proposition. If an event $A \subseteq C$ then $P[A \mid B \cap C]$ equals $P[A \mid B]/P[C \mid B]$. In this case A is equal to $C_{i-1}^{(c)} = s + v_i$ and C is chosen as $C_{i-1}^{(c)} \geq v_i$.

(E') As opposed to the first case the expected number of slots is now 2^{n-i+1} times the expected number of collisions at level $i-1$ given that we do poll on level i and we had k contenders. Also since the event $P_a = i$ is the same as $R_a \geq i \cap C_{i-1}^{(a)} < v_i$ we are actually looking for

$$E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i \cap C_{i-1}^{(a)} < v_i]. \tag{24}$$

We start with the following observation

$$E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i] =$$
$$P[C_{i-1}^{(a)} \geq v_i \mid X_a = k \cap R_a \geq i]\ E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i \cap C_{i-1}^{(a)} \geq v_i] + \ldots$$
$$P[C_{i-1}^{(a)} < v_i \mid X_a = k \cap R_a \geq i]\ E[C_{i-1}^{(a)} \mid X_a = k \cap R_a \geq i \cap C_{i-1}^{(a)} < v_i], \tag{25}$$

where the expression of interest is part of the right-hand side. Both probabilities are clearly each others complement and thus it is sufficient to calculate the first. To do this remark again that if an event $A \subseteq C$ then $P[A \mid B \cap C]$ equals $P[A \mid B]/P[C \mid B]$. Application of this result with A equal to $C_{i-1}^{(a)} \geq v_i$ and with C as $R_a \geq i$, (A is a part of C because $v_i > 0$) yields the following expression for the first probability

$$P[C_{i-1}^{(a)} \geq v_i \mid X_a = k]/P[R_a \geq i \mid X_a = k]. \tag{26}$$

Both these values were obtained in section **(A')**. Again two expected values remain unknown, **(E1')** and **(E2')** are devoted to them.

(E1') We start with the one in the right-hand side. Notice that event $C_{i-1}^{(a)} \geq v_i$ is a part of the event $R_a \geq i$ (as mentioned above) and this first event is the same as $C_{i-1}^{(c)} \geq v_i$ when conditioned on $X_a = k$ or $X_c = k$ respectively. Thus the expression we are looking for is reduced to (23). It remains to find the left-hand side of expression (25).

(E2') Remark that the event $R_a \geq i$ coincides with $C_{i-1}^{(a)} > 0$. As the event $C_{i-2}^{(a)} \geq v_{i-1}$ contains this last event, we can also write it as $C_{i-2}^{(a)} \geq v_{i-1} \cap C_{i-1}^{(a)} > 0$. So, we want to find

$$E[C_{i-1}^{(a)} \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1} \cap C_{i-1}^{(a)} > 0]. \tag{27}$$

Some straightforward reasoning based on the definition of the expected value yields

$$E[C_{i-1}^{(a)} \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1} \cap C_{i-1}^{(a)} > 0] = \frac{E[C_{i-1}^{(a)} \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1}]}{1 - P[C_{i-1}^{(a)} = 0 \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1}]}. \tag{28}$$

Applying $P[A \mid B \cap C] = P[A \cap C \mid B]/P[C \mid B]$ we find the probability in the denominator. Thus

$$P[C_{i-1}^{(a)} = 0 \mid X_a = k \cap C_{i-2}^{(a)} \geq v_{i-1}] = \frac{P[C_{i-1}^{(a)} = 0 \cap C_{i-2}^{(a)} \geq v_{i-1} \mid X_a = k]}{P[C_{i-2}^{(a)} \geq v_{i-1} \mid X_a = k]}, \tag{29}$$

due to our discussion in **(A1')** we can substitute the super- and subscripts a for c in both probabilities without altering their values. Having done this we use (13) and (14) for the computation of the denominator, while the nominator is obtain based on a similar argument as in **(A2')**

$$P[C_{i-1}^{(c)} = 0 \cap C_{i-2}^{(c)} \geq v_{i-1} \mid X_c = k] = \sum_{u=v_{i-1}}^{\lfloor \frac{k}{2} \rfloor} 2^{(n-i+2)(k-2u)} \frac{\left(C_2^{2^{n-i+2}}\right)^u C_u^{2^{i-2}} C_{k-2u}^{2^{i-2}-u}}{C_k^{2^n}} \left(\frac{2^{n-i+1}}{2^{n-i+2}-1}\right)^u. \tag{30}$$

The expression is the same as in **(A2')**, but with $i-1$ substituted for $i-2$ (remember that $v_i = 1 + u_{i-1}$).
We end with the determination of the expected value in the right-hand side of (28). Again we can substitute the sub- and superscripts a for c. Then using the definition of the expected value we get

$$E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-2}^{(c)} \geq v_{i-1}] = \sum_{l \geq v_{i-1}} E[C_{i-1}^{(c)} \mid X_c = k \cap C_{i-2}^{(c)} = l] \frac{P[C_{i-2}^{(c)} = l \mid X_c = k]}{P[C_{i-2}^{(c)} \geq v_{i-1} \mid X_c = k]}. \tag{31}$$

Finally we calculate the nominator of this sum using the same methodology as in (13) and (14), where we define $e(i-1,s,k)$ as $E[2^{i-1} - C_{i-1}^{(c)} \mid X_c = k \cap C_{i-2}^{(c)} = 2^{i-2} - s] * P[C_{i-2}^{(c)} = 2^{i-2} - s \mid X_c = k]$. This results in the following equations (again the first equation is a consequence of (2))

$$e(i-1, 2^{i-2}, k) = 2^{i-1} \frac{2^{(n-i+2)k} C_k^{2^{i-2}}}{C_k^{2^n}} \tag{32}$$

$$e(i-1,s,k) = C_s^{2^{i-2}} \sum_{l_1=0}^{s} 2^{(n-i+2)l_1} \left(2s + (2^{i-1} - 2s) \frac{C_{k-l_1}^{m_i} + 2^{n-i+1} C_{k-l_1-1}^{m_i}}{C_{k-l_1}^{2^n - s2^{n-i+2}}}\right) \cdot \frac{C_{l_1}^s C_{k-l_1}^{2^n - s2^{n-i+2}}}{C_k^{2^n}} - \sum_{x=1}^{2^{i-2}-s} C_s^{s+x} e(i-1, s+x, k) \tag{33}$$

with m_i equal to $2^n - s2^{n-i+2} - 2^{n-i+1}$.
We end this section by remarking that the expected number of slots used in this scheme given that we had k contenders is independent of the way the slots (polling and contention slots) are incorporated into the frame structure. Only the probability of having k contenders depends on the frame structure.

3.4 Delay and Throughput when the First Levels are Skipped (STATIC)

In this section we describe the necessary adaptations to the evaluation methods above such that it allows us to evaluate the scheme when some of the first levels of the tree are skipped. We restrict to the ISA protocol with polling as setting N_p equal to zero leads to the performance measures of the other scheme.
The starting level is denoted by S_l. The following random variables are defined:

- X_a^+ : the number of participants in the scheme, this variable ranges from 0 to 2^n.
- R_a^+ : the level at which the scheme is resolved, this variable ranges from S_l to n.
- $S_i^{(a+)}$: the number of slots used at level i.
- P_a^+ : the level at which we poll, if the scheme is solved without polling, the variable obtains the value $n+1$.

We start with the delay analysis.

3.4.1 The Delay when the First Levels are Skipped (STATIC)

To solve this problem we follow the same lines of reasoning as before. In this section we will address the most significant differences with the previous evaluation (of ISA combined with polling). Before going into the mathematical details, let us summarize the two major differences regarding the behaviour of the protocol. First the scheme can no longer be solved before level S_l as these levels no longer exist. Secondly polling during level S_l is no longer possible as level $S_l - 1$ is skipped.

(A+) Let us start with R_a^+. Notice that if the scheme was resolved at the first level S_l then it is also solved at or before level S_l with the basic scheme and vice versa. Secondly the events $R_a^+ \leq x$ and $R_a \leq x$ coincide if $x > S_l$. Thus we have (due to (2))

$$\begin{aligned} P[R_a^+ = S_l \mid X_a^+ = k] &= \frac{2^{(n-S_l)*k} C_k^{2^{S_l}}}{C_k^{2^n}}, \\ P[R_a^+ \leq S_l + x \mid X_a^+ = k] &= P[R_a \leq S_l + x \mid X_a = k], \end{aligned}$$

for every value $x > 0$. This means that the probability of resolving the scheme before or at level S_l might decrease a bit, compared to the scheme that starts at level zero. If so the probability that it is solved at level $S_l + 1$ increases together with the probability of polling at this level. Remark that the probability of solving the scheme at level $S_l + 1$ without polling remains the same. For the other levels everything remains exactly the same as before. Finally we define $p_a^+(k,i)$ as $P[R_a^+ = S_l + i - 1 \mid X_a^+ = k]$.

$\mathcal{F}_a^+(i,k)$ is defined as the probability that a tagged request is successful at or before level i. With similar arguments as used for R_a^+ we get

$$\begin{aligned}\mathcal{F}_a^+(S_l,k) &= \frac{C_{k-1}^{2^n-2^{n-S_l}}}{C_{k-1}^{2^n-1}},\\ \mathcal{F}_a^+(S_l+x,k) &= \mathcal{F}_a(S_l+x,k),\end{aligned}$$

for every positive value x. The remainder of the analysis is analogue to the one with starting level zero.

3.4.2 The Throughput when the First Levels are Skipped (STATIC)

The main objective of this section is to find the expected number of slots used at each level. Once we have these values, the distribution of X_a^+ allows us to calculate the throughput.
This new scheme clearly never polls at level S_l (or before since these levels don not exist), thereby the probability of polling at level S_l+1 is increased. This causes the expected number of slots during level S_l and S_l+1 to be different from the ones we had before. All the other expected values stay the same. The expected number of slots at level S_l matches 2^{S_l} because we start at this level.
The situation for level S_l+1 is a bit more complicated. We start with equation (22) (where we add a '+' to all random variables and set i equal to S_l+1). In view of the discussion in **(A+)**, adding a '+' only changes the first two values (of the right-hand side) in this expression. Based on the fact that the events at level S_l are similar to those of the ISA scheme without polling and with the starting level at zero the product of these two values is given by

$$\begin{aligned}&P[P_a^+ = S_l+1 \mid X_a^+ = k]E[S_{S_l+1}^{(a+)} \mid X_a^+ = k \cap P_a^+ = S_l+1] =\\ &\quad 2^{n-S_l} * \sum_{i=1}^{u_{S_l}} iP[C_{S_l}^{(c)} = i \mid X_c = k] \qquad (34)\end{aligned}$$

This concludes the throughput analysis.

3.5 Delay and Throughput Analysis when the First Levels are Skipped (DYNAMIC)

Having done the analysis for the static starting level it is easy to extend these results to the proposed dynamic model. We use the same random variables as above but substitute the '+' sign for a '$*$' to indicate the dynamic nature of the scheme. We also introduce a new random variable B^* as the starting level.

3.5.1 The Delay when the First Levels are Skipped (DYNAMIC)

As always we start with the search for R_a^* but now when conditioned on X_a^* and B^*. Assuming that the starting level equals S_l we have the following (due to the STATIC part)

$$P[R_a^* = S_l \mid X_a^* = k \cap B^* = S_l] = \frac{2^{(n-S_l)*k}C_k^{2^{S_l}}}{C_k^{2^n}}, \qquad (35)$$

$$P[R_a^* \le S_l + x \mid X_a^* = k \cap B^* = S_l] = P[R_a \le S_l + x \mid X_a = k], \qquad (36)$$

with x a positive number. Having found this we define $p_a^*(k,S_l,x+1)$ as $P[R_a^* = S_l+x \mid X_a^* = k \cap B^* = S_l]$.

To find the joint distribution of (X_a^*, B^*) it is sufficient to construct the following transition matrix and do the matrix inversion

$$t_a^*(k; S_b, j; S_a) = P[X_{n+1}^{(a*)} = j \cap B_{n+1}^* = S_a \mid X_n^{(a*)} = k \cap B_n^* = S_b] =$$
$$\sum_{t=1}^{n+1-S_b} \frac{(\lambda t)^j e^{-\lambda t}}{j!} p_a^*(k, S_b, t) 1_{\{(t \le B_l \wedge S_a = S_b - 1) \vee (B_l < t < B_m \wedge S_a = S_b) \vee (t \ge B_m \wedge S_a = S_b + 1)\}}.$$

Suppose that we observe the system at an arrival time, then the probability that there are k contenders in the current CC and that this CC started at level S_l is needed. We also need this probability for the next CC (after the one containing the random arrival). These values are the natural extensions of (4) and (5)

$$P[X_{next}^{(a*)} = k \cap B_{next}^* = S_l] = \frac{P[X_a^* = k \cap B^* = S_l]k}{E[X_a^*]} \tag{37}$$

and

$$c_a^*(k, S_l) = P[X_{cur}^{(a*)} = k \cap B_{cur}^* = S_l] =$$
$$\sum_{j=1}^{2^n} \sum_{S_n = S_{min}}^{S_{max}} \frac{P[X_a^* = k \cap B^* = S_l] t_a^*(k; S_l, j; S_n) j}{E[X_a^*]}. \tag{38}$$

Finally we need to find $\mathcal{F}_a^*(i, k, S_l)$, being the probability that a tagged arrival is successful at or before level i, knowing that there were $k-1$ other participants and the CC started at level S_l. Again, using the results of the previous section (see STATIC) we obtain

$$\begin{aligned} \mathcal{F}_a^*(S_l, k, S_l) &= \frac{C_{k-1}^{2^n - 2^{n-S_l}}}{C_{k-1}^{2^n - 1}}, \\ \mathcal{F}_a^*(S_l + x, k, S_l) &= \mathcal{F}_a(S_l + x, k). \end{aligned}$$

As before we can combine these results to obtain the average delay of the system. Let us now focus on the delay density function. As in (9), s is the number of frames that the tagged element competes, j is the length of the CC (in frames) in which the tagged request was generated and S_b the level at which this CC started. While $l-1$ is the number of other competitors next to the tagged element,

$$D_c(x) = \sum_{s=1}^{\lfloor x \rfloor} \sum_{S_b = S_{min}}^{S_{max}} \sum_{j = \lceil x \rceil - s}^{n+1} \sum_{l=1}^{2^n} \frac{\mathbf{\Delta}_1 \mathcal{F}_a^*(f(S_b, j) + s - 1, l, f(S_b, j))}{j} \cdot$$
$$\cdot \mathcal{G}_j(l) \left(\sum_{k=0}^{2^n} \frac{j \; P[R_a^* = S_b + j - 1 \mid X_a^* = k \cap B^* = S_b]}{1 + E[R_a^* - S_b \mid X_a^* = k \; \cap \; B^* = S_b]} c_a^*(k, S_b) \right), \tag{39}$$

where the function $f(S_b, j)$ is given by

$$f(S_b, j) = \begin{cases} \max(S_{min}, S_b - 1) & j \le B_l \\ S_b & B_l < j < B_m \\ \min(S_{max}, S_b + 1) & j \ge B_m \end{cases}, . \tag{40}$$

and $\mathbf{\Delta}_1 \mathcal{F}_a^*(x, y, z)$ equals $\mathcal{F}_a^*(x, y, z) - \mathcal{F}_a^*(x - 1, y, z)$.

3.5.2 The Throughput when the First Levels are Skipped (DYNAMIC)

In the section above we obtained the joint distribution of (X_a^*, B^*). This is used to derive an expression for

$$T_a^* = \frac{E[X_a^*]}{\sum_{k=0}^{2^n} \sum_{S_l=S_{min}}^{S_{max}} P[X_a^* = k \cap B^* = S_l] * E[\sum_i S_i^{(a*)} \mid X_a^* = k \cap B^* = S_l]}, \tag{41}$$

where the expected values were obtained in the evaluation of the static model.

3.6 Delay and Throughput for Multiple Instances

The advantage of the analysis, as demonstrated above, is that we can use it in a direct manner to obtain results for the scenario with multiple instances. This is due to the fact that the delay experienced by a tagged request is independent of the instance it belongs to.

4 Numerical Results

In this section, we use the analytical model to investigate the impact of the arrival rate λ, the trigger value N_p and the starting level S_l on the mean delay, the delay density function and the throughput. The system parameters are set as follows:

- The number of mobiles is 128, hence $n = 7$.
- The arrival rate λ (requests per frame) varies between 0.05 and 3.5.
- The three values studied for N_p (i.e. the size of the address space determining the instant at which a switch from contention resolution using splitting to polling occurs) are 0, 20 and 40, where the first case corresponds with ISA without polling.
- The starting level S_l will vary from level 0 to 2.
- When studying a system with a dynamic starting level, B_l and B_m are set to 1 and 4 respectively. Therefore the level is decreased by one, if the CC is solved in one frame and is increased by one, if the CC consist of four or more frames. The boundary values are set as follows: $S_{min} = 0$ and $S_{max} = 2$.
- The number of instances varies between 1 and 4.

We study four different scenarios. First we investigate the impact of the polling threshold N_p. In that case the starting level is fixed at 0. Next the influence of the starting level S_l is discussed. Then the impact of the use of a dynamic scheme for the starting level is considered. Finally we look at the effect of using multiple instances of ISA. More numerical results for higher values of λ and S_l are found in [17].

4.1 The Influence of the Polling Threshold on the System Performance

Figures 5 and 6 show the influence switching to polling has on the mean delay and the throughput. As expected we get a tradeoff between the delay and throughput characteristics: the sooner the ISA protocol switches to polling, the shorter the mean delay, but the lower the throughput.

From the figures we observe that the protocol behaves very similar for different N_p values when the arrival rate λ is small (below 0.25). A similar result is obtained for large values of λ (beyond

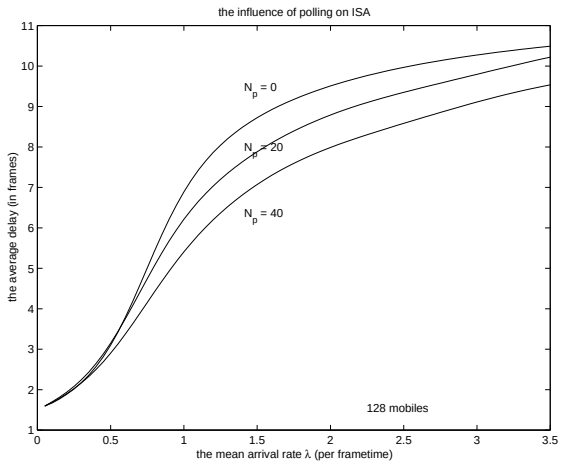

Figure 5: *The impact of polling on the mean delay*

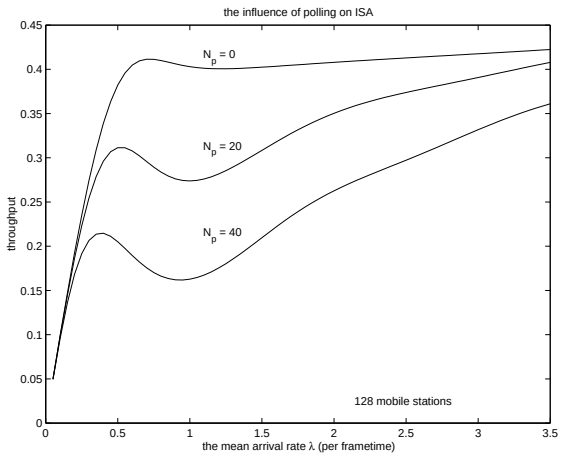

Figure 6: *The impact of polling on the throughput*

5). Both these results are intuitively clear. Polling is not an issue in these cases: for λ very small, collisions rarely occur and hence ISA solves the collisions; if λ is very large, the remaining size of the address space is too large to switch to polling. For the used system parameters, the asymptotic values for the mean delay and throughput are 12 and $\frac{128}{255}$.
Let us now consider moderate values for λ. Recall that for the polling threshold $N_p = 40$, resp. $N_p = 20$, the protocol will never start polling until level 3, resp. level 4. Thus the impact of N_p on the performance measures is low for small values of λ. If the arrival rate increases (look at the range 0.5 till 1), the probability that collisions at level 2, resp. 3 are introduced increases. In most cases these collisions contain very few participants, and hence occasionally 32, resp. 16 polling slots are provided at level 3, resp. 4 to poll very few competitors. Therefore the throughput decreases with increasing value of N_p. If λ is increased even more, beyond one, polling is postponed in most cases to a later level (as the expected number of collisions at level 2, resp. 3 becomes larger than one) and will contain more participants. This results in higher throughput values for a fixed value of N_p.

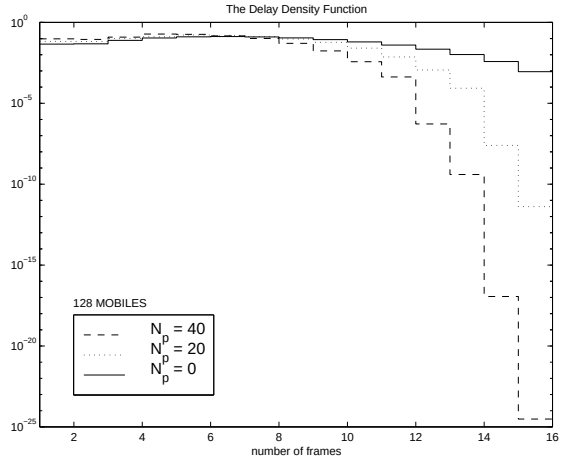

Figure 7: *The impact of polling on the delay density function with $\lambda = 1$*

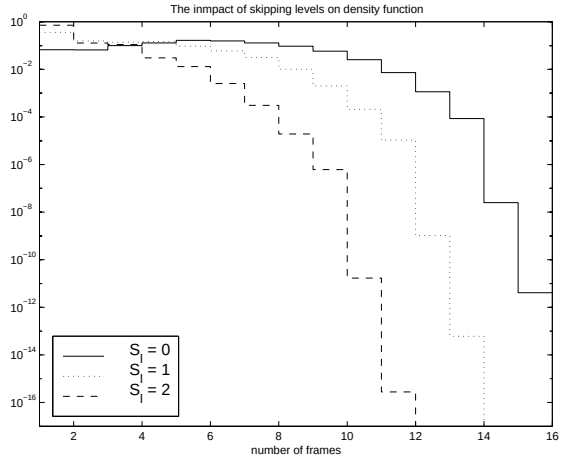

Figure 8: *The impact of Skipping with $\lambda = 1$ and $N_p = 20$*

Now we investigate whether the conclusion that the mean waiting time decreases due to polling still holds for the tail of the delay distribution. Figure 7 shows the impact of polling on the delay distribution function for $\lambda = 1$. It illustrates the fact that the main improvement of the delay is located in the tail of the distribution.

4.2 The Influence of Skipping Levels (STATIC) on the System Performance

Figures 9 and 10 illustrate the impact of S_l on the average delay and the throughput. In these figures we have three different types of curves, full, dotted and dashed, corresponding to $S_l = 0, 1$ and 2 respectively. Moreover for each value of S_l the results for $N_p = 0, 20$ and 40 are depicted. For a fixed value of S_l, the upper curve corresponds to $N_p = 0$, the middle curve to $N_p = 20$ and the lowest to $N_p = 40$.

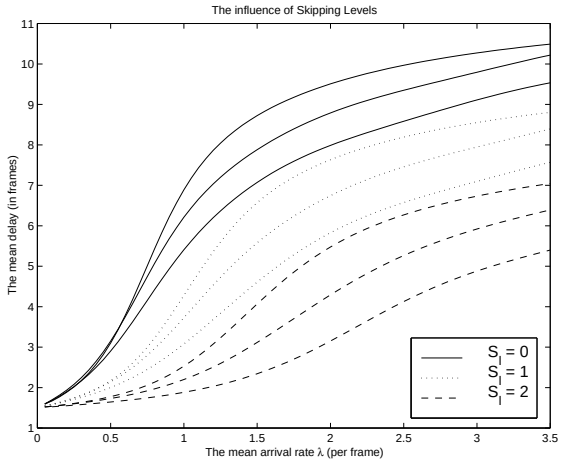

Figure 9: *The influence of skipping on the mean delay.*

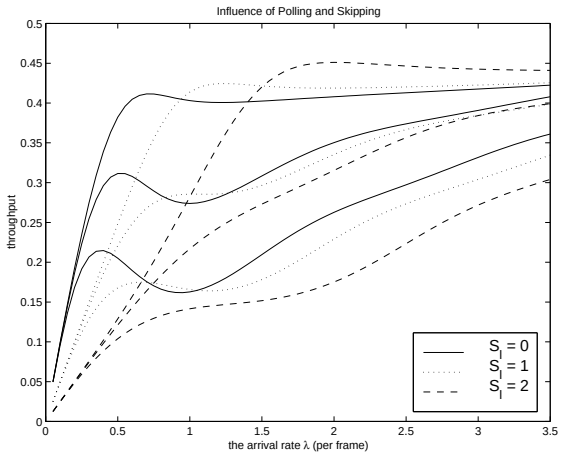

Figure 10: *The influence on the throughput for N_p = 0, 20 and 40.*

Clearly skipping the first levels leads to a decrease of the mean waiting time. The asymptotic value now is $12 - 1.5 \times S_l$. Let us now focus on the impact of polling for variable values of S_l. First Figure 9 shows a faster decrease of the delay due to polling, when the starting level is larger. This can be seen by observing the area between the curves for $N_p = 0, 20$ and $N_p = 40$. Secondly, notice that the curves converge slower for increasing value of S_l (observe the differences for $\lambda = 3.5$ in Figure 9). Figure 10 represents the throughput results for $N_p = 0, 20$ and 40. We see that for low values of λ skipping levels results in a lower throughput (as most of the S_l slots are wasted). If λ becomes larger this loss is converted in a small gain, due to the fact that the majority of the slots before level S_l contain collisions.
The influence of skipping levels on the delay density function is shown in Figure 8.

4.3 The Influence of Skipping Levels (DYNAMIC) on the System Performance

From the previous sections we may conclude that a higher starting level has a positive impact on the delay and even on the throughput, especially for larger values of λ. Unfortunately a high price is paid for this in terms of throughput if λ is small. The aim of this section is to show that the dynamic scheme as proposed in section 2.4 solves this problem. That is, if λ is small the result should tend to the results for $S_l = S_{min}$, while for λ large the behavior should be similar to the one corresponding to $S_l = S_{max}$.

Figures 11 and 12 show that this is the case (for $N_p = 20$), meaning that a system where the levels are skipped dynamically, is able to limit the maximum delay while keeping the throughput high.

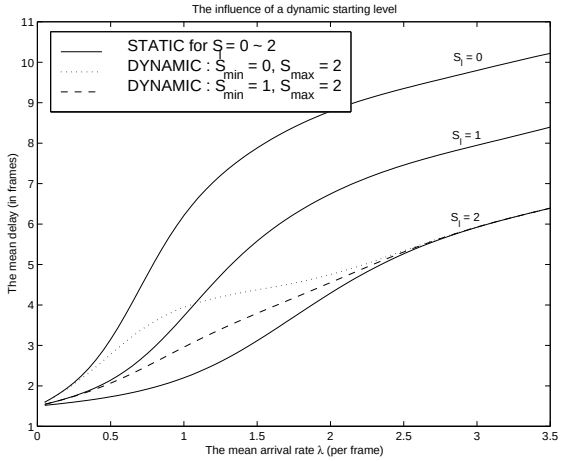

Figure 11: *The influence of dynamic skipping on the delay ($N_p = 20$).*

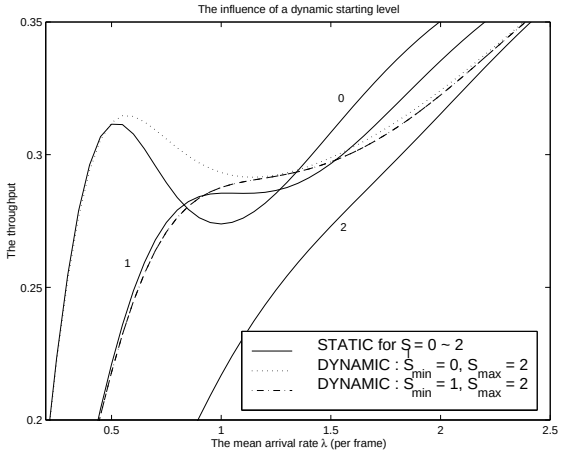

Figure 12: *The influence of dynamic skipping on the throughput for $N_p = 20$.*

4.4 The influence of Multiple Instances of ISA on the System Performance

Notice that in this final scenario λ varies between 0 and 6. Figures 13 and 14 show the delay and throughput results for three configurations. In the first we have one instance and the starting level S_l is fixed at 2. In the second we have two instances, with $S_l = 1$. Finally we have four instances, with $S_l = 0$.
Clearly the more instances we use, the better the average delay. Except for very small and very large values of λ, were all scenarios perform alike. Indeed, based on intuition, it is clear that the behaviour in both halves (or quarters) in the tree is very similar in both these cases. Thereby uncoupling has little influence.

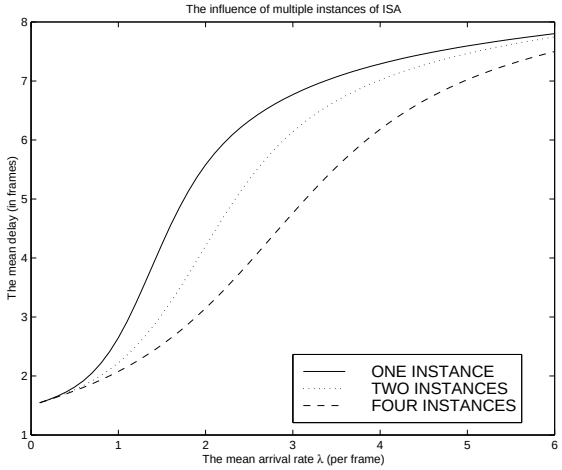

Figure 13: *The influence of multiple instances on the delay.*

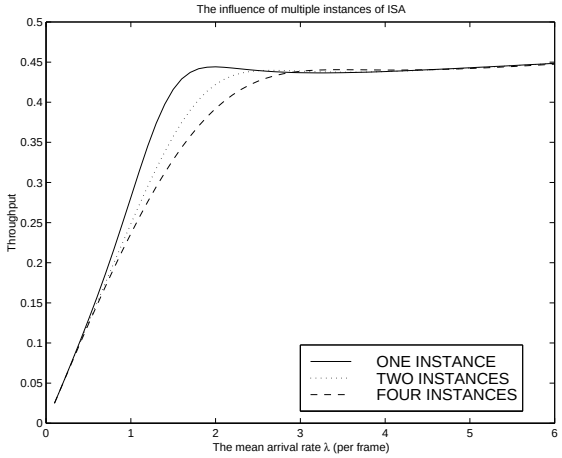

Figure 14: *The influence of multiple instances on the throughput.*

As before, for more moderate values of λ, there exists a tradeoff between the delay and throughput, thus the more instances we use, the smaller the delay and the lower the throughput is. Still this time the decrease in throughput is considerably smaller, thereby making the use of multiple instances attractive.

5 Conclusions

In this paper the performance analysis of the Identifier Splitting Algorithm combined with polling, when employed on an uplink contention channel used in a Wireless ATM environment, was presented. Both the delay density function and the throughput characteristics were obtained analytically. Multiple numerical results have shown that the Identifier Splitting Algorithm combined with polling, enhanced by a mechanism of skipping in a dynamic way the first levels, leads to a good trade off between low delay and high throughput results. Finally the use of multiple instances was shown to have a good impact on the delay, with little throughput losses.

References

[1] D. Bertsekas and R. Gallager. Data networks. *Englewood Cliffs, New Jersey, Prentice-Hall*, 1987.

[2] J.I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Trans. Inform. Theory, Vol 25, No 5, pp. 319 - 329*, 1979.

[3] C.S. Chang, K.C. Cheng, M.Y. You, and J.F. Chang. Guaranteed quality-of-service wireless access control for wireless ATM environments. *IEEE J. on Selected Areas on Comm., Vol. 15, No 1, pp. 106-118, Jan.*, 1997.

[4] Jean-Pierre Ebert, Ralf Holtkamp, Adam Wolisz, and Louis Ramel. A distributed media access control for wireless ATM environments. *6th WINLAB Workshop on Third Generation Wireless Systems, New Brunswick, NJ, USA, March*, 1997.

[5] D.J. Goodman, R.A. Valenzuela, K.T. Gayliard, and B. Bamamurthi. Packet reservation multiple access for local wireless communications. *IEEE Transactions on Communications, vol. COM-37, pp. 885-890, Aug.*, 1989.

[6] Mark J. Karol, Zhao Liu, and Kai Y. Eng. Distributed-queueing request update multiple access (DQRUMA) for wireless packet (ATM) networks. *Wireless Networks, Baltzer Science Publishers, Vol. 1, no III*, 1995.

[7] Jouni Mikkonen, James Aldis, Geert Awater, Andrew Lunn, and David Hutchison. The MAGIC WAND-functional overview. *IEEE Journal on Selected Areas in Communications, Vol. 16, No. 6, August*, 1998.

[8] Nikos Passas, Sarantis Paskalis, Dimitra Vali, and Lazaros Merakos. Quality-of-service-oriented medium access control for wireless ATM networks. *IEEE Communications Magazine, November*, 1997.

[9] Dietmar Petras. Medium access control protocol for wireless, transparent ATM access in MBS. *RACE Mobile Telecommunications Summit, (Cascais, Portugal)*, 1995.

[10] Dietmar Petras and A. Kramling. MAC protocol with fast collision resolution for an ATM air interface. *IEEE ATM Workshop, San Francisco, Aug.*, 1996.

[11] Dietmar Petras, A. Kramling, and A. Hettich. MAC protocol for wireless ATM: contention free versus contention based transmission of reservation requests. *7th IEEE PIMRC*, 1996.

[12] Dietmar Petras and Andreas Kramling. Fast collision resolution in wireless ATM networks. *2nd MATHCOM, Vienna, Austria, Feb*, 1997.

[13] John Porter and Andy Hopper. An ATM based protocol for wireless LANs. *ORL technical report*, 1994.

[14] F.D. Priscolli. Medium access control protocol for the median systems. *ACTS Mobile Summit, Granada, Nov.*, 1996.

[15] P.F.M. Smulders and C. Blondia. Application of the asynchronous transfer mode in indoor radio networks. *Proc. of Joint PIMRC and WCN conference 94, The Hague, pp. 839, netherlands*, 1994.

[16] B. Tsybakov and N. Vvedenskaya. Random multiple access stack algorithms. *Prob. Inform. Trans., Vol. 16, pp 230 - 243, Jul-Sept.*, 1980.

[17] B. Van Houdt, C. Blondia, O. Casals, J. Garcia, and D. Vazquez-Cortizo. A MAC protocol for wireless ATM systems: Supporting the ATM service categories. *Proceedings of 16th International Teletraffic Conference (ITC)*, 1999.