Analysis of Robust Collision Resolution Algorithm with Successive Interference Cancellation and Bursty Arrivals

Sergey Andreev,
Tampere University of Technology,
Tampere, FINLAND
E-mail: sergey.andreev@tut.fi

Eugeny Pustovalov, and Andrey Turlikov,
State University of Aerospace Instrumentation,
St. Petersburg, RUSSIA
E-mails: eugeny@vu.spb.ru, turlikov@vu.spb.ru

***Abstract*—A novel tree multiple access algorithm is discussed, which is robust to imperfect successive interference cancellation. The algorithm exploits a single memory location to store the captured packets and thus has feasible complexity. For this algorithm, the performance evaluation is conducted, which focuses on its transmission rate and mean packet delay. Both virtual and actual delays are established analytically. Additionally to Poisson arrivals, bursty arrival model is considered and the analysis is extended respectively. The results are new and important to conclude upon the efficiency of the proposed algorithm.**

***Index Terms*—multiple access, collision resolution, tree algorithm, successive interference cancellation, performance analysis.**

I. Introduction and Background

Contemporary wireless networks exploit various Medium Access Control (MAC) approaches suggesting efficient solutions to arbitrate access of multiple users to the shared communication channel. Among these, Random Multiple Access (RMA) algorithms are preferred under bursty load and allow reaching considerably low mean packet delay values even when user population is considerably high.

We remind that each RMA algorithm comprises a Channel Access Algorithm (CAA) and a Collision Resolution Algorithm (CRA). The former arbitrates user channel access, whereas the latter handles collisions, whenever two or more packets are transmitted over the channel simultaneously. Currently, the family of ALOHA-based algorithms is widely exploited and represented by diversity slotted ALOHA, binary exponential backoff, and many other schemes. Importantly, as these approaches are very simple, they concentrate more on CAA rather than on CRA and thus are less efficient. Generally, in case of a collision, the packet retransmission is deferred for a random time interval.

By contrast, tree RMA algorithms focus on CRA and thus demonstrate better performance comparing to ALOHA-based solutions. Historically, Standard Tree Algorithm (STA) and Modified Tree Algorithm (MTA) were the first ones to be proposed in [1] and [2] independently. We refer to these approaches as to *conventional* tree algorithms in what follows.

The previous analysis of conventional tree algorithms typically assumed that whenever packets collide (or, interfere wirelessly) the receiver extracts no meaningful information. Recent advances at the physical layer enable the application of Successive Interference Cancellation (SIC) techniques [3], [4] to improve performance. SIC may be naturally used in the uplink channel of contemporary cellular networks [5], as common receiver (e.g. base station) facilitates the operation of SIC.

A novel approach that tailors SIC to a tree algorithm (SICTA) was proposed in [6]. Summarizing, SIC processes the previously stored collision packets (signals) and takes advantage of unbounded signal memory. Whereas infeasible, the availability of infinite memory allows the original SICTA to double the performance of STA. This promising idea was taken further and a variety of SICTA modifications were proposed in [7], [8], [9], [10].

All the existing SICTA-based solutions may be classified into two categories. Firstly, there are algorithms that assume perfect SIC operation and therefore are susceptible to cancellation errors falling into a deadlock. Secondly, there are algorithms that are robust to imperfect SIC operation, but at the same time are unstable when the number of users grows unboundedly. In our previous work [11], we proposed a robust SICTA (R-SICTA) algorithm that tolerates cancellation errors and demonstrates nonzero performance when the user population is infinite. In this paper, we conduct the analysis of the extended R-SICTA and establish its transmission rate and mean packet delay. In particular, we consider both virtual and actual delay definitions from [12] and conclude upon the efficiency of the proposed algorithm.

II. System Model

A. Proposed Algorithm

We consider *classical* system model widely used to compare and contrast various RMA algorithms and described as long ago as in [13], [14]. We further extend it to account for the SIC operation. Figure 1 demonstrates a simple example of how SIC may improve the time of collision resolution.

We assume that the infinite population of users communicates data packets to a common receiver. In each time slot, the receiver acquires packets (signals) from its wireless

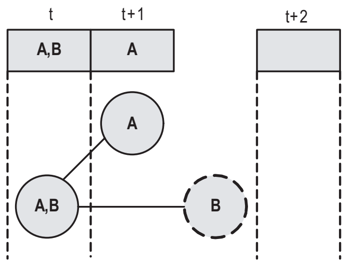


Figure 1: A simple example of SIC operation

link and processes them. Consequently, the receiver reports the information about the channel events (empty, success, or collision) and the SIC output to the users by the end of the current slot. The feedback is error-free and is available immediately.

Consider two data packets from users A and B colliding in slot t. Denote by y_t the signal at the receiver by the end of slot t and by x_A and x_B the transmitted signals corresponding to packets from A and B respectively. Therefore, the signal y_t is a combination of signals x_A and x_B. Assuming an error-free link, we may write $y_t = x_A + x_B$. Processing the signal y_t, the receiver detects a collision and stores y_t in its signal memory. The collided users then decide if they retransmit in slot $t + 1$ with probability $\frac{1}{2}$. According to our example, only one user accesses the channel in slot $t + 1$. The receiver thus acquires the signal $y_{t+1} = x_A$ and extracts the signal x_A. As such, the packet from A is received successfully.

Further, the SIC procedure at the receiver processes the stored collision signal y_t and cancels the successfully extracted signal x_A. We denote the cancellation operation by $y_t - x_A$ and obtain $\tilde{y}_t = y_t - x_A$. The SIC receiver then extracts the signal $x_B = \tilde{y}_t$ also by the end of slot $t + 1$ and thus the packet from B is also received successfully. As such, the subsequent collision resolution is not necessary. In the considered example, the collision resolution process lasts one slot less taking advantage of SIC capability. We say that this slot may be *skipped*. Otherwise, a retransmission of the packet from B would be attempted according to a conventional CRA.

Most works on SICTA operation assume perfect interference cancellation. For the example in Figure 1, it means that x_B is extracted from y_t and x_A with probability 1. We however extend the model for the more realistic case of imperfect SIC operation. Consequently, the packet from B is restored successfully with some probability less than 1. Otherwise, a cancellation error occurs and the signal $\tilde{y}_t = y_t - x_A$ is perceived as a collision with the complementary probability. The cancellation success probability depends on the SIC receiver implementation and on the channel events. In our previous work [11], we proposed a novel R-SICTA algorithm that is robust to cancellation errors and takes advantage of a single signal memory location.

The main purpose of the considered R-SICTA algorithm exemplified in Figure 2 is to avoid the deadlock effect of the original SICTA algorithm. In the example, the best case time diagram corresponds to two successful SIC operations, whereas the worst case diagram illustrates two cancellation errors. For more details on the algorithm operation, including its formal description the reader is referred to [11]. In Figure 3, we address important channel events as follows.

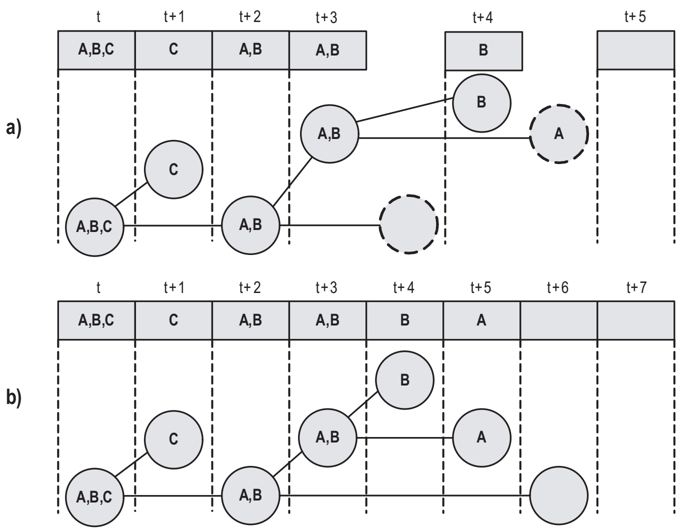


Figure 2: Examples of proposed R-SICTA operation

- Figure 3, a, when a collision slot is followed by an empty slot. The MTA rules allow to skip a collision slot with probability 1.
- Figure 3, b, when a collision slot is followed by a *collision* slot. The SIC operation allows to skip an *empty* slot with probability $1 - q_{ce}$.
- Figure 3, c, when a collision slot is again followed by a *collision* slot. The SIC operation allows to skip a *success* slot with probability $1 - q_{cs}$.
- Figure 3, d, when a collision slot is followed by a *success* slot. The SIC operation allows to skip a *success* slot with probability $1 - q_{ss}$.

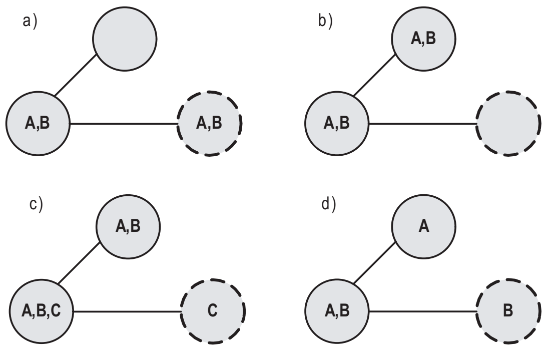


Figure 3: Some important channel events

B. Main Definitions

Following [12], we formally define an RMA *algorithm* Ω as a rule according to which a user with a pending packet decides whether it should attempt its transmission in slot $t \in \{1, 2, ...\}$ or to defer channel access.

We assume that if a packet arrives in slot t then the arrival time is a random variable x distributed uniformly over the slot duration $[t-1, t)$. As such, an RMA algorithm is a function of three arguments:

- Packet arrival time x.
- Sequence $\boldsymbol{\theta}(t) = (\theta_1, \dots, \theta_t)$ of channel events. Here $\theta_i = E$ if slot i is empty, $\theta_i = S$ if there is a success in slot i, and $\theta_i = C$ if there is a collision in slot i.
- Sequence $\boldsymbol{\nu}^{(x)}(t) = (\nu_1^{(x)}, \dots, \nu_t^{(x)})$ of user decisions regarding the packet arrived at time x. Here $\nu_i^{(x)} = 0$ if the user decided not to transmit the packet in slot i and $\nu_i^{(x)} = 1$ otherwise.

Analyzing the performance of an RMA algorithm Ω, its transmission rate and mean packet delay are typically of primary interest [15]. These important metrics established in the framework of the discussed classical model are commonly used to compare and contrast the performance of various RMA algorithms.

Packet *delay* $\delta_\Omega(\lambda)$ is a random time interval from the moment x a packet arrives into the system to the moment when the successful transmission of this packet ends. Here λ is the overall mean arrival rate of new packets.

Considering now the mean packet delay, there are two popular definitions depending on how the delay is actually measured [12]. Let us inject an extra packet into the system at time t and denote its delay by $\delta_\Omega^{(t)}(\lambda)$. Mean *virtual* packet delay is:

$$D_\Omega(\lambda) \triangleq \limsup_{t \to \infty} E[\delta_\Omega^{(t)}(\lambda)]. \tag{1}$$

Alternatively, we enumerate all the packets in the order of their arrival into the system. Mean *actual* packet delay is:

$$D'_\Omega(\lambda) \triangleq \limsup_{n \to \infty} E[\delta_\Omega^{(n)}(\lambda)], \tag{2}$$

where $\delta_\Omega^{(n)}(\lambda)$ is the delay of the tagged packet number n.

In the classical case of Poisson arrivals, the mean virtual delay and the mean actual delay are equal for known tree algorithms [15], that is $D'_\Omega(\lambda) = D_\Omega(\lambda)$. However, if arrivals are bursty, the mean delays may diverge depending on the arrival flow parameters. We elaborate more on this fact in what follows.

Finally, the transmission rate (or, maximum stable throughput) of an RMA algorithm is the highest possible (Poisson) arrival rate λ that still yields finite packet delay with probability 1, that is:

$$R_\Omega \triangleq \sup\{\lambda : D_\Omega(\lambda) < \infty\}. \tag{3}$$

Note that conventional STA and MTA have the transmission rates of 0.346 and 0.375 respectively for binary and fair tree splitting.

III. Transmission Rate Calculation

A. Standard Tree Algorithm

In the rest of the text, we concentrate exclusively on the tree multiple access algorithms. As such, the function $\Omega(x, \boldsymbol{\theta}(t), \boldsymbol{\nu}^{(x)}(t))$ may take only two different values: 0 and 1. As such, a user either transmits its packet in a slot or defers transmission with probability 1. The function Ω is determined by both CAA and CRA. Accordingly, the end nodes (leaves) of a collision resolution tree are labeled by events E or S, whereas remaining nodes are labeled by event C.

Collision Resolution Interval (CRI) length is defined as the number of slots required to build a complete collision resolution tree. For conventional STA, the number of nodes in its collision resolution tree is equal to the CRI length. Denote random CRI length by τ. Its conditional mean gives the average collision resolution time:

$$T_k = E[\tau | \text{collision of size } k \text{ is being resolved}]. \tag{4}$$

For the transmission rate R_Ω of any known tree algorithm, the following bounds are proved [1]:

$$\liminf_{k \to \infty} \frac{k}{T_k} < R_\Omega < \limsup_{k \to \infty} \frac{k}{T_k}. \tag{5}$$

Asymptotically, for the conventional STA it holds [16]:

$$\frac{T_k}{k} = \frac{2}{\ln 2} + H \sin(2\pi \log_2 k + \phi) + O\left(\frac{1}{k}\right), \tag{6}$$

where $H = 3.127 \cdot 10^{-6}$ and $\phi = 0.9826$.

From (5) and (6), it readily follows:

$$\left(\frac{2}{\ln 2} + H\right)^{-1} < R < \left(\frac{2}{\ln 2} - H\right)^{-1}, \tag{7}$$

where R is the transmission rate of the conventional STA. Therefore, we repeat the classical result:

$$R \approx \frac{2}{\ln 2}. \tag{8}$$

B. SICTA-based Algorithms

Below we propose a simple but general method to calculate the transmission rate of existing SICTA-based algorithms. The key idea is that the operation of any known tree algorithm may be regarded as the collision resolution tree of STA, where the time required to label some tree nodes is zero (as the respective time slots are skipped). Our method is composed of the following steps:

- A rule to match the collision resolution tree nodes with the CRI length is formulated for the studied algorithm.
- Contrasting this rule against the STA rule, the mean CRI length of the studied algorithm is expressed as a function of the CRI length of STA.
- Accounting for (6), the bounds on the transmission rate of the studied algorithm are established.

For the original SICTA algorithm, each tree node labeled by event C is put into correspondence to a value of a collision signal. This value is either extracted directly from the link or calculated by SIC. The rule of correspondence between the tree nodes and the time slots, as well as collision signal extraction rule are the following.

Consider slot t which corresponds to the *upper* tree node. Let $V(P)$ be the signal corresponding to node P, P_{cur} be the node corresponding to the current slot t, P_{tmp} be the *lower* tree node adjacent to node P_{cur}, and P_{prev} be the parent node for both P_{cur} and P_{tmp}. We describe the possible situations in slot t as follows.

1) If in slot t node P_{cur} receives label C, then in this slot node P_{tmp} is not labeled. The upper child of node P_{cur} becomes the current node in slot $t+1$.
2) If in slot t node P_{cur} receives label E (see Figure 4, a), then in this slot node P_{tmp} is labeled by C and $V(P_{tmp}) = V(P_{prev})$. The upper child of node P_{cur} becomes the current node in slot $t+1$.
3) If in slot t node P_{cur} receives label S (see Figure 4, b), then in this slot for node P_{prev} the following steps apply:
   - $V_{tmp} = V(P_{prev}) - V(P_{cur})$ is calculated.
   - V_{tmp} value is analyzed by SIC. If V_{tmp} corresponds to a successful signal or to a collision signal, then in this slot node P_{tmp} is labeled by S or C respectively.
     As such, the lower child of node P_{prev} is labeled.
   - For nodes P_{prev} and P_{tmp} the signal values are established: $V(P_{prev}) = V_{tmp}$, $V(P_{tmp}) = V_{tmp}$.
   - If V_{tmp} corresponds to a successful packet, then node P_{prev} is temporarily set as the current. For this node, its adjacent node, and its parent node the above three steps are repeated. That is, the collision resolution tree is analyzed backwards toward the root and as the result some lower nodes may become labeled. If V_{tmp} corresponds to a collision, then the upper child of node P_{tmp} becomes the current node in slot $t+1$.

From the above description of the original SICTA algorithm, it follows that all the lower nodes are labeled in no time (and the respective time slots may be skipped). As such, the following equation holds:

$$T_k^S = \frac{T_k - 1}{2} + 1, \tag{9}$$

where T_k^S is the mean CRI length for the original SICTA algorithm.

Using the above together with (6), we establish the following bounds on the transmission rate R_S of SICTA:

$$\left(\frac{1}{\ln 2} + \frac{H}{2}\right)^{-1} < R_S < \left(\frac{1}{\ln 2} - \frac{H}{2}\right)^{-1}. \tag{10}$$

As the bounds are close enough, the R_S of SICTA is approximately 0.693. This value was established earlier in [6] with a far more complicated analytical approach.

However, as mentioned above, the unbounded signal memory requirement of the original SICTA algorithm and its vulnerability to cancellation errors prohibit its practical implementation. To mitigate these limitations, we refrain from

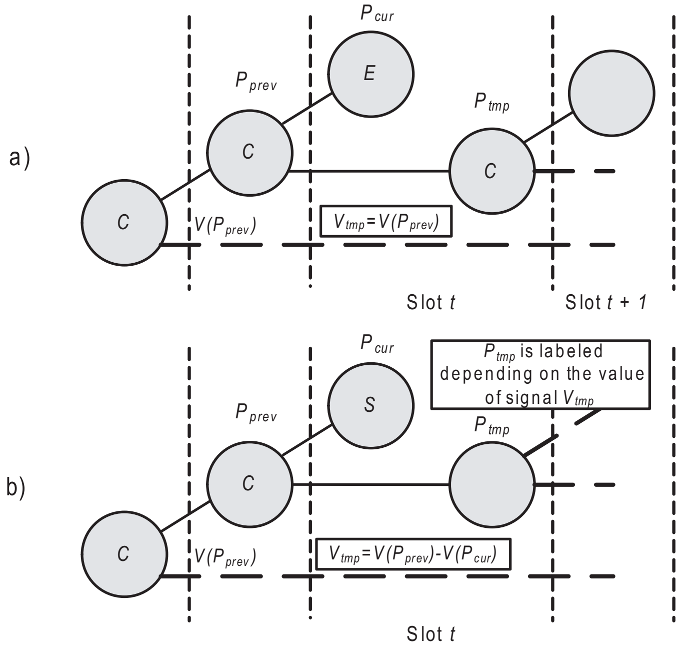


Figure 4: Tree node labeling for SICTA

skipping some lower tree node and thus propose R-SICTA algorithm. R-SICTA is more robust for the cost of some reduction in the transmission rate. Its formal description from [11] requires respective modifications of situations 1 and 3 (see Figure 5). Therefore, and also using relationships from [16], the mean CRI length T_k^{RS} of R-SICTA is given by:

$$T_k^{RS} = \left(\frac{1}{2} + \frac{q_{ce}}{4}\right) T_k - \frac{1}{2} + \frac{q_{ce}}{4} + \frac{k}{2}(1 + q_{cs} - q_{ce}) + \frac{N_k}{2}(q_{ss} - q_{cs}), \tag{11}$$

where N_k is the mean number of collisions of size two in the STA collision resolution tree of initial size k, whereas probabilities q_{ce}, q_{ss}, and q_{cs} were defined above.

The derivation of N_k is a separate problem that we solved in [11]. To summarize, with the precision of at least three decimal digits, $\limsup\limits_{k\to\infty} \frac{N_k}{k} = \liminf\limits_{k\to\infty} \frac{N_k}{k} = \gamma = 0.721$. We note that as the bounds on the transmission rate R_{RS} of R-SICTA are again close enough, the following approximate value holds:

$$R_{RS} \approx \frac{2 \ln 2}{2 + q_{ce} + 2 \ln 2 (1 + q_{cs} - q_{ce} + (q_{ss} - q_{cs})\gamma)}. \tag{12}$$

In particular, in case of perfect interference cancellation ($q_{ce} = q_{ss} = q_{cs} = 0$), the R_{RS} of R-SICTA is approximately 0.515.

The proposed transmission rate calculation technique is novel and important for the performance evaluation of tree RMA algorithms, as it allows for the analysis of known and future algorithms using the existing knowledge behind STA.

IV. Mean Packet Delay

A. Virtual Delay Analysis

We consider an embedded Markov chain, where chain state is the CRI length. As such, a semi-infinite transition probability matrix may be composed and its stationary distribution π_n

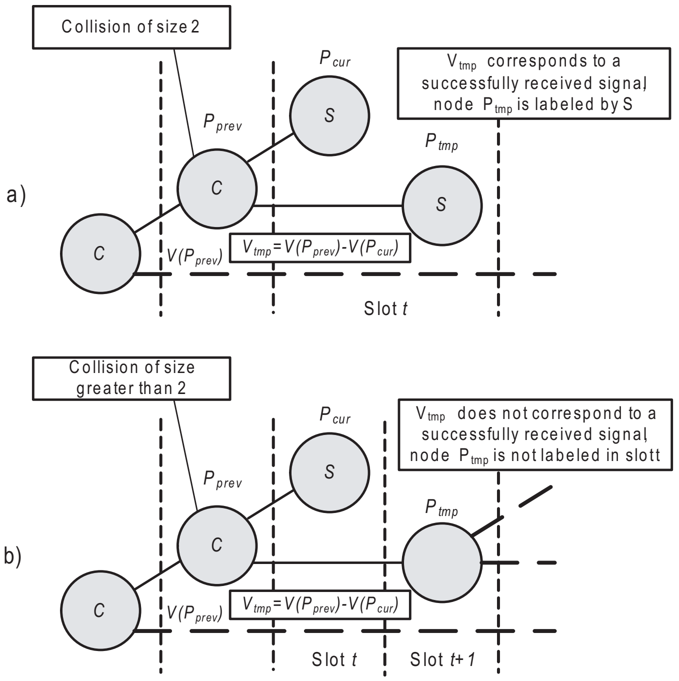

Collision of size 2
P_{cur}
V_{tmp} corresponds to a successfully received signal, node P_{tmp} is labeled by S
P_{prev}
S
P_{tmp}
a)
C
S
C
$V(P_{prev})$
$V_{tmp}=V(P_{prev})-V(P_{cur})$
Slot t
Collision of size greater than 2
P_{cur}
V_{tmp} does not correspond to a successfully received signal, node P_{tmp} is not labeled in slott
P_{prev}
S
P_{tmp}
b)
C
C
$V(P_{prev})$
$V_{tmp}=V(P_{prev})-V(P_{cur})$
Slot t
Slot $t+1$

Figure 5: Tree node labeling for R-SICTA

may be established numerically. For the sake of simplicity, we change some of the previous notations in this section. Using Markov chain theory, we calculate the probability that a tagged packet arrives during a CRI of length n as:

$$\widetilde{\pi}_n = \frac{n\pi_n}{N}, \tag{13}$$

where N is the mean CRI length.

For the considered R-SICTA algorithm, the virtual packet delay δ is composed of two components $\delta = \delta_1 + \delta_2$, where δ_1 is the time interval from the packet arrival time to the end of the current CRI and δ_2 is the time interval from the start of the following CRI to the end of the successful transmission of the packet.

Coming to the expected values, the mean virtual delay of R-SICTA is given by $D_{RS} = E[\delta_1] + E[\delta_2]$. We introduce auxiliary notations: u_n is the mean value of δ_1 conditioning on the fact that the tagged packet arrives during the CRI of length n, whereas $d(n)$ is the mean value of δ_2 conditioning on the same fact. Therefore, we have:

$$D_{RS} = \sum_{n=1}^{\infty} \widetilde{\pi}_n \cdot [u_n + d(n)]. \tag{14}$$

We also introduce d_k as the mean CRI length conditioning on the fact that the size of the current collision without the tagged packet is k. The value of d_k is determined by the properties of CRA. As the size of the current collision depends on the number of new arrivals during the previous CRI, the values $d(n)$ and d_k are related as:

$$d(n) = \sum_{k=0}^{\infty} d_k \cdot Pr\left\{\xi_u = k | \tau_{u-1} = n\right\}. \tag{15}$$

Here the probability $Pr\left\{\xi_u = k | \tau_{u-1} = n\right\}$ is determined by the arrival flow type, where τ_i is the duration of the i-th collision and ξ_i is the size of the i-th collision.

Additionally to Poisson arrival flow, we consider a more realistic bursty arrival model. The state of the packet source is set randomly and memorylessly to either ON or OFF at the beginning of each slot with probabilities p_{on} and $p_{off} = 1 - p_{on}$ respectively. In state ON, the packets arrive according to Poisson distribution with the mean arrival rate of λ_{on}. Similarly, the mean Poisson arrival rate in the OFF state is λ_{off}. The resulting overall mean arrival rate is thus $\lambda = p_{on}\lambda_{on} + (1-p_{on})\lambda_{off}$. Clearly, when $p_{on} = 1$ the arrival flow becomes Poisson with the mean arrival rate of λ_{on}.

For both Poisson and bursty arrival flows it holds $u_n = \frac{n}{2}$. Further, for Poisson arrival flow only we establish:

$$Pr\left\{\xi_u = k | \tau_{u-1} = i\right\} = \frac{(\lambda i)^k}{k!} \cdot e^{-\lambda i}, \tag{16}$$

whereas for bursty arrival flow it holds:

$$\begin{aligned} &Pr\left\{\xi_u = k | \tau_{u-1} = i\right\} = \\ &= \sum_{s=0}^{t} \sum_{i=0}^{k} \tfrac{(\lambda_{on} s)^i}{i!} e^{-\lambda_{on} s} \tfrac{[\lambda_{off} \cdot (t-s)]^{k-i}}{(k-i)!} e^{-\lambda_{off}(t-s)} \beta, \end{aligned} \tag{17}$$

where $\beta = C_t^s p_{on}^s (1-p_{on})^{t-s}$.

B. Actual Delay Analysis

We now consider an embedded Markov chain, where chain state is the size of a collision. Again, a semi-infinite transition probability matrix may be composed and its stationary distribution π_k may be established numerically. Using Markov chain theory, we calculate the probability that a tagged packet is transmitted during a CRI of size k as:

$$\widetilde{\pi}_k = \frac{k\pi_k}{K}, \tag{18}$$

where K is the mean CRI size.

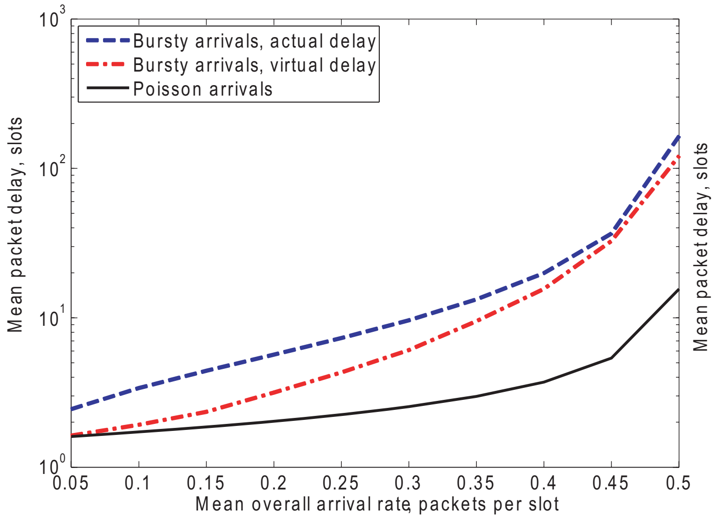
For the considered R-SICTA algorithm, the actual packet delay δ' is again composed of two components $\delta' = \delta'_1 + \delta'_2$, where δ'_1 is the time interval from the packet arrival time to the end of the current CRI and δ'_2 is the time interval from the start of the following CRI to the end of the successful transmission of the packet.

Analogously to the above, the mean actual delay of R-SICTA is given by $D'_{RS} = E[\delta'_1] + E[\delta'_2]$. Its first component is calculated as:

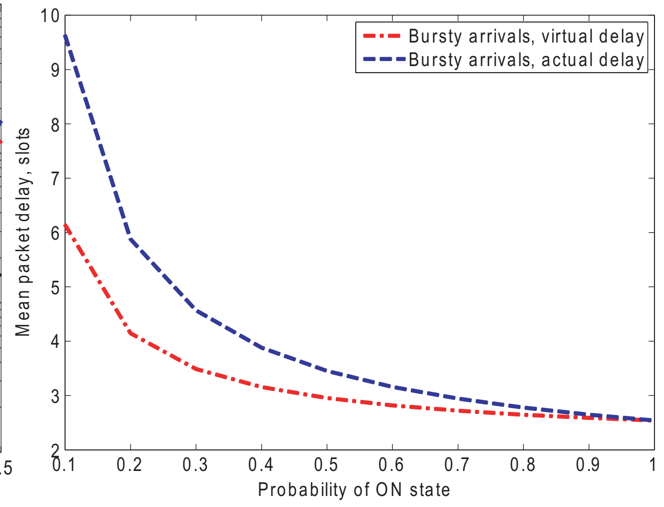
$$D'_1 = \sum_{s=1}^{\infty} \widetilde{\pi}(s) \cdot u_s, \tag{19}$$

where $u_s = \frac{s}{2}$ and $\widetilde{\pi}(s)$ is the probability that a packet arrives during CRI of length s. The second component is calculated as:

$$D'_2 = \sum_{k=1}^{\infty} \widetilde{\pi}_k \cdot d_{k-1}. \tag{20}$$



(a) Delay vs. arrival rate



(b) Delay vs. probability of ON state

Figure 6: Mean packet delay results

V. Numerical Results and Conclusions

In this section, we exemplify our main result of mean packet delay calculation for the proposed R-SICTA algorithm. For simplicity, here we assume perfect interference cancellation ($q_{ce} = q_{ss} = q_{cs} = 0$) and $\lambda_{off} = 0$. We remind that when there are no cancellation errors, the transmission rate of R-SICTA is approximately 0.515.

Figure 6 plots mean packet delay values for two scenarios. In particular, Figure 6, a contrasts Poisson and bursty arrivals with equal overall mean arrival rate λ. Mean virtual and actual delays are also shown, of which the former is generally lower as the tagged packet has a high probability to arrive during the OFF state. In Figure 6, b we fix $\lambda = 0.3$ and vary p_{on}. As expected, when $p_{on} = 1$ both virtual and actual packet delays converge.

In this paper, we considered a family of multiple access algorithms using successive interference cancellation. A simple but robust tree algorithm with a single signal memory location was proposed and analyzed. We established its transmission rate and mean packet delay, whereas we differentiated between virtual and actual delay definitions. Additionally, both Poisson and bursty packet arrival flows are taken into account. We conclude that the proposed algorithm is an attractive and feasible solution to be exploited in contemporary cellular networks with common receiver, e.g. at the bandwidth requesting stage in [5], to increase channel utilization and decrease mean delay.

Acknowledgments

This work is supported by the Russian Foundation for Basic Research (projects #10-08-01071-a and #11-08-01016-a) as well as by the Branch of Nano- and Information Technologies of Russian Academy of Sciences (project 2.3).

References

[1] B. S. Tsybakov and V. A. Mikhailov, “Free synchronous packet access in a broadcast channel with feedback,” *Problems of Information Transmission*, vol. 14, no. 4, pp. 32–59, 1978.

[2] J. I. Capetanakis, “Tree algorithms for packet broadcast channels,” *IEEE Transactions on Information Theory*, vol. 25, no. 4, pp. 505–515, 1979.

[3] A. Agrawal, J. Andrews, J. Cioffi, and T. Meng, “Iterative power pontrol for imperfect successive interference cancellation,” *IEEE Transactions on Wireless Communications*, vol. 4, no. 3, pp. 878–884, 2005.

[4] S. Weber, J. Andrews, X. Yang, and G. Veciana, “Transmission capacity of wireless ad hoc networks with successive interference cancellation,” *IEEE Transactions on Information Theory*, vol. 53, no. 8, pp. 2799–2814, 2007.

[5] *IEEE Std 802.16m (D9), Amendment to IEEE Standard for Local and metropolitan area networks – Advanced Air Interface*.

[6] Y. Yu and G. B. Giannakis, “SICTA: A 0.693 contention tree algorithm using successive interference cancellation,” *Proceedings of IEEE INFOCOM*, vol. 3, pp. 1908–1916, 2005.

[7] Y. Yu and G. B. Giannakis, “High-throughput random access using successive interference cancellation in a tree algorithm,” *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4628–4639, 2007.

[8] X. Wang, Y. Yu, and G. B. Giannakis, “A robust high-throughput tree algorithm using successive interference cancellation,” *IEEE Transactions on Communications*, vol. 55, no. 12, pp. 2253–2256, 2007.

[9] G. T. Peeters, B. V. Houdt, and C. Blondia, “A multiaccess tree algorithm with free access, interference cancellation and single signal memory requirements,” *Performance Evaluation*, vol. 64, no. 9-12, pp. 1041–1052, 2007.

[10] G. T. Peeters and B. V. Houdt, “Interference cancellation tree algorithms with k-signal memory locations,” *IEEE Transactions on Communications*, vol. 58, no. 11, pp. 3056–3061, 2010.

[11] S. Andreev, E. Pustovalov, and A. Turlikov, “SICTA modifications with single memory location and resistant to cancellation errors,” *Proceedings of NEW2AN*, vol. 5174/2008, pp. 13–24, 2008.

[12] B. S. Tsybakov, “Survey of USSR contributions to random multiple-access communications,” *IEEE Transactions on Information Theory*, vol. 31, no. 2, pp. 143–165, 1985.

[13] R. Rom and M. Sidi, *Multiple Access Protocols: Performance and Analysis*. Springer-Verlag, 1990.

[14] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, 1992.

[15] N. D. Vvedenskaya and B. S. Tsybakov, “Packet delay in the case of a multiple-access stack algorithm,” *Problems of Information Transmission*, vol. 20, no. 2, p. 137–153, 1984.

[16] G. S. Evseev and A. M. Turlikov, “A connection between characteristics of blocked stack algorithms for random multiple access system,” *Problems of Information Transmission*, vol. 43, no. 4, pp. 345–279, 2007.