RC 20348 (1/22/96)
Computer Science/Mathematics 15 pages

IBM Research Report

A Review of Random Access Algorithms

Chatschik Bisdikian

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York 10598

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

A Review of Random Access Algorithms

Chatschik Bisdikian

IBM Research Division
T. J. Watson Research Center
30 Saw Mill River Rd., M/S H2-C24
Hawthorne, NY 10532, USA
e-mail: bisdik@watson.ibm.com

Abstract: *A common component of all IEEE 802.14 MAC*[1] *proposals is the use of a some form of a* random access *process for accessing the network. The need for random access is dictated by the presence of a large number of stations, most of which are expected to be silent at any single moment, that have no means to coordinate directly their transmissions with each other. In this contribution, we discuss some of the most known random access algorithms and review some of their pros and cons.*

[1]This paper was presented at the January 1996 meeting of the IEEE 802.14 standards committee (contribution no. IEEE802.14-96/019).

1 Introduction

If there is a common element that all 802.14 MAC proposals share is that *random access* is used at some point during the operation of the (802.14) network. Random access is used either during the initialization (power-up) of new stations, or during an "initial" transmission of a station, or in both cases. The use of random access is dictated by the presence of a large number of stations (in the order of a few thousands), most of which are expected to be silent at any single moment, that have no means to coordinate directly with each other their transmission attempts.

The theory of *random access algorithms* (RAAs) for medium access in computer communication networks is maturing for well over a quarter of a century and its roots spread several decades (even centuries) into the past. A number of RAAs have been suggested, studied, analyzed, and compared in hundreds of technical papers in the open literature. Some of them have been also widely deployed in commercial products. In this contribution, we will review some of the most known RAAs, and we will discuss several of their pros and cons. In our presentation, we will focus on the fundamental properties of RAAs, rather than their specific instantiation within a MAC protocol, by, say, using multiple contention minislots, etc. Our presentation will be based mostly on well known information already available in the literature. This information has been scrutinized by myriads of researchers working all over the world on this fascinating topic.

2 The system assumptions

We will be considering only versions of the RAAs that operate in a synchronized environment like the one encountered in a 802.14 environment. As a matter of fact, without lack of generality, we assume that the time axis is slotted, comprised of fixed size *slots* (time slices), with each station transmitting at most one (information) *packet* within the slot boundaries. Following the transmission of packets in a slot, stations learn about the outcome of these transmissions before the following slot. This information will be called *feedback*, and, in the 802.14 network case, it is provided by the *headend* station which monitors all traffic activity on the network.

Note that although many 802.14 MAC proposals do not assume fixed size slots, and that in 802.14 networks feedback information cannot be instantaneous (even the speed of light has a finite value), for the discussion presented here, our system assumptions are sufficient. For all 802.14 MAC proposals, it is assumed that station initialize with a *ranging* process to learn their round-trip distance from the headend. Having acquired the ranging information, stations then delay their subsequent transmissions in such a way that transmissions from various station arrive at the headend in sync. Also, with the concept of RAA engine *interleaving*, we may focus in the operation of only one these RAA engines, for which the concept of feedback being available prior to the next RAA action is clearly valid.

Before proceeding with our presentation, we will like to start with the definition of the stability of a network operating under a particular RAA. Let λ and S denote the (long term) arrival and departure rates, respectively, of packets to[2] and from the network.

Definition: A network operating under an RAA is called *stable* if and only if there exists a *positive* number Λ such that for all arrival rates λ of new packets generated in the network such that $\lambda \leq \Lambda$, the arrival rate λ and the departure rate S are equal.

In other other words, in a stable system $S = \lambda$, i.e., the rates are maintained, for all $\lambda \leq \Lambda$, where $\Lambda > 0$.

The positive number Λ will be called the *throughput* of the RAA, while the range of arrival rates between 0 and Λ specifies the *stability region* of the RAA.

In our presentation, we will first review the two major random access "philoshopies," the *individualism* of the *ALOHA* based algorithms, and the *togetherness* of the *splitting* algorithms. We will then provide some of their performance characteristics.

3 The ALOHA family of algorithms

This is the most popular family of RAAs. It is straightforward to implement, and it has enjoyed wide commercial deployment. Assuming stability, to be discussed later, it is very easy to analyze, thus also explaining the plethora of published technical papers related to the various flavors of ALOHA.

The philosophy of individualism of the ALOHA algorithms dictates its followers (the network stations) to be interested in the outcome, i.e., feedback, of their *own* transmissions *only*, through positive acknowledgements (feedbacks) . Hence, a station does not need to monitor the activity on the network except just before (for carrier-sense in Ethernet networks, not to be discussed any further) and after its transmission. If the transmission is unsuccessful, i.e., the station does not receive any acknowledgement, the station reschedules a future transmission and remains "inactive" until that time.

There is a number of alternatives for selecting the retransmission time.

- uniformly select a time interval over the next K time slot;
- transmit at the next time slot with probability p.

The parameters K or p can be either static or dynamic depending on the number of times that a station has experienced collisions. The later case gives rise to the very popular *binary exponential back-off* algorithm, used in the 802.3 Ethernet LANs:

- A station that has experienced m collisions selects randomly to (re)transmit after 1 to 2^m slots. The variable m starts at 0 and extends up to an upper limit M, which is commonly around 10 although not necessarily.
- If m has reached the upper limit M the station terminates transmission attempts and (for the time) discards the packet.

[2] In the case of the arrival rate, λ accounts only the *newly* generated packets. It does not account for the retransmitted packets.

The idea behind the exponential back-off algorithm is to "spread" farther and farther apart in time the retransmissions of (collided) stations, since repeated collisions may be an indication that the traffic load in the network has increased.

The exponential back-off algorithm gives more transmission opportunities to newer packets, thus resulting in a *quasi-last-in-first-out* (quasi-LIFO) packet transmission ordering.

3.1 The stability region of ALOHA: Is there any?

The ALOHA algorithms are notoriously unstable. It has been proven analytically and verified through simulation studies, that for very large station population[3] purely acknowledgement based versions of ALOHA presented before cannot maintain any arrival rate. In other words, no matter at what rate new traffic is generated (can be as close to 0 as desired), if the system operates for a sufficiently long time it will eventually be overwhelmed by collisions, and nothing will be able to be transmitted. In the case of the truncated ($M < \infty$) binary exponential back-off algorithm, it is guaranteed that many packets will always experience M collisions and hence will be discarded no matter what the arrival rate is; hence, $S < \lambda$, for any λ. Thus, when applied to a very large unspecified (frequently modeled as an infinite) station population, the purely acknowledgement based ALOHA algorithms are unstable for any arrival rate λ of new packets, and they possess no stability region, i.e., $\Lambda = 0$.

The instability problem of ALOHA has long been documented, although it appears to the author that it has not received its due attention. For good discussions on the instability of ALOHA see [3, 8], two excellent textbooks on the analysis of computer networks, [1, part 4] and reference therein. Next, borrowing from [13], we present an intuitive explanation of the instability with the help of figure 1.

Assuming an overall Poisson packet traffic model, figure 1 shows the packet departure rate S as a function of the *total offered load* G, i.e., the load due to the new *and* retransmitted packets, see figure 2; assuming stable operation, we should also have $S = \lambda$. Suppose that the total offered load is G_1 with corresponding departure rate $S = G_1 e^{-G_1}$. These quantities represent (long term) averages and the offered load and the departure rate will fluctuate from time to time around these "nominal" values. If at some point the offered load increases a little above G_1 so will the departure rate as well, say to S_1. Now since $S_1 > S = \lambda$, i.e., the instantaneous packet departure rate is higher than the arrival rate, the number of backlogged stations will decrease and the offered load will return back to its nominal value G_1. Now if the offered load increases beyond G_2, something to happen with probability 1, see also [8], then the new departure rate, say, S_2 will become less than $S\,(= \lambda)$. Hence, the departure rate becomes less than the arrival rate which will cause an increase of backlogged stations and a corresponding increase of the total offered load G, which will cause an even smaller departure rate S_2', and so on. Thus, with time the departure rate from the system drops to 0, i.e., independently of the arrival rate λ, eventually no successful transmissions are possible from the system. As shown in [2], even the more "robust" exponential back-off algorithm suffers from this catastrophic behavior.

In [8] is shown that even finite station population systems can suffer from instability.

[3] Recall that the 802.14 network is to support up to a few thousand stations.

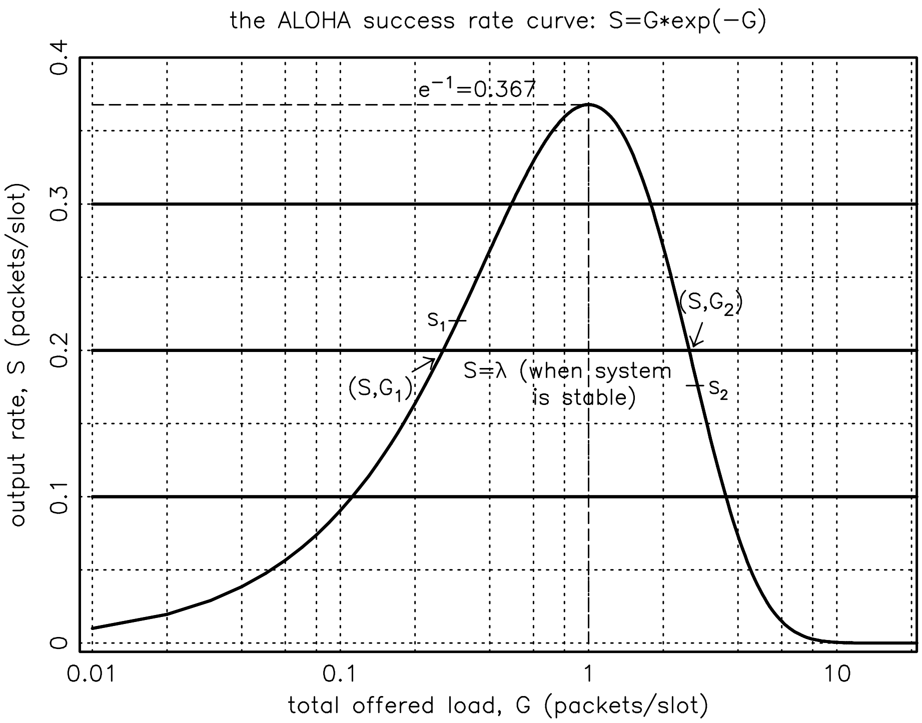

Figure 1: The "throughput" curve of ALOHA.

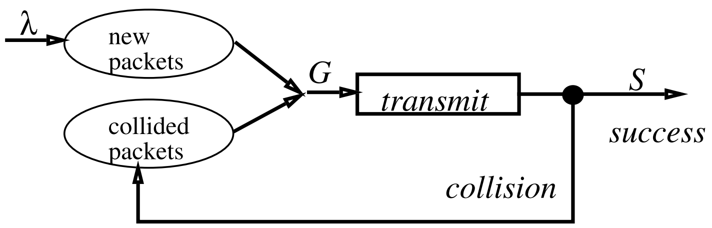

Figure 2: The arrival rate λ, offered load G, and departure rate S in an ALOHA network.

For example, for a system of 200 stations that use ALOHA with $K = 10$ slots to share a 2 Mb/sec channel and that cumulatively generate about 0.30 packets/slot, it will require only about 1.4 minutes of system operation for it to run into the "instability region," when each slot carries 1 Kbit packets, and a mere 34 seconds for 53-byte slots.

Short of total system restart, recovery from the instability region requires the loss of packets in order to reduce the system backlog, and hence reduce the offered load to the system.

3.2 Stabilizing ALOHA

Note that in figure 1, S achieves its maximum $e^{-1} \approx 0.367$ at $G = 1$. This value of S is typically considered as the throughput of ALOHA. As we have just discussed, this is misleading and incorrect.

ALOHA "look-alike" RAAs that achieve a stable e^{-1} throughput could be devised. For example, the retransmission probability could be dynamically change as the number of backlogged stations also chances. Ideally, one should select $p = 1/n$, where n represents the number of backlogged stations. Practically though, n cannot be known, and sophisticated techniques need to be employed for estimating it.

In [3] and references therein, the following slot-to-slot update algorithm for the estimate $\widehat{n}$ of the backlogged stations is presented:

$$\widehat{n}_{k+1} = \begin{cases} \max\{\lambda, \widehat{n}_k + \lambda - 1\}, & \text{for non\_collision slots;} \\ \widehat{n}_k + \lambda + (e-2)^{-1}, & \text{for collision slots.} \end{cases}$$

This algorithm is shown to achieve a stable throughput of e^{-1}, but it requires that all stations use the same retransmission probability from slot to slot given by $p_k = 1/\widehat{n}_k$. Since the arrival rate λ is not known, this too has to be estimated, but unless the estimate of λ is substituted by the constant e^{-1} no proof of stability of the algorithm exists.

This and other [3, 13] stable "ALOHA" algorithms require stations to monitor the traffic activity at each slot and adjust their retransmission decisions based on the latest feedback information received by the system. This feedback can be in the form of binary Collision/Non_Collision (C/NC) feedback information, or ternary Collision/Empty/Success (C/E/S) feedback information, or information given to the station by a monitor station, e.g., the headend in a 802.14 station. Based on this feedback, *all* the stations at each slot set the parameters p, or K, of ALOHA to the same value. Clearly, this kind of operation can be considered heretic to the ALOHA philosophy of individualism.

4 Splitting algorithms

To address the instability of the ALOHA algorithm, splitting (collision resolution) algorithms have been proposed that require stations to take transmissions and retransmission decisions collectively based on continuous observation of the feedback transmissions, just like in the stable "ALOHAs." What is interesting though is that there exist simple splitting algorithms that outperform ALOHA even in its stable versions.

There are two versions of the splitting algorithms.

- *blocked-access* splitting algorithms, usually referred to as *tree-search* algorithms: no new packets are permitted into the system while a collision resolution is under way;

- *free-access*, usually referred to as *stack* algorithms: new packets can "jump" right-in the during an on-going collision resolution.

A hybrid version of blocked and free access algorithms is not going to be further discussed [13].

For random access with splitting algorithms, collided stations split in subgroups by "flipping dices" with n sides (usually, $n = 2$), and stations that flip an i (the i-th subgroup), $i = 0, 1, \ldots, n-1$, transmit after they determine that all stations in subgroups $j < i$ have successfully transmitted. Since, this process can be easily visualized through an n-ary tree (especially the blocked-access ones), or a stack (especially the free-access ones), these algorithms are also identified by these names.

In random access with splitting algorithms, one can identify *collision resolution intervals* (CRIs), the termination of which can be easily identified by a monitor station. At the termination instant of a CRI, it is guaranteed that *all* packets collided at the beginning of and during the CRI have been successfully transmitted. In ALOHA networks, especially when the retransmission probability p is used, no CRIs can be identified.

According to the splitting algorithms, all active stations participate in the collision resolution process at every slot based on feedback information they receive for each slot. Note that although all stations take some actions at each slot, not all of them transmit at every slot, which depends on the "state" of the station that is updated at each slot. This collective effort by all stations guarantees that the system will have a stable throughput.

Splitting algorithms are further separated based on the feedback they require for their operation. Blocked-access tree-search algorithms are also categorized according to the *first transmission rule* that controls when and how the blocked stations will first transmit in the system.

4.1 Binary vs. ternary feedback

An example of the a binary tree-search algorithm with *binary* C/NC *feedback* is shown in figure 3. Following each collision, the collided packets are (randomly) split amongst the n branches of a tree that has as a root the "current transmission slot." In figure 4, we show a free-access stack algorithm (splitting parameter $n = 2$) with binary C/NC feedback. Following each collision, the collided packets are (randomly) split amongst the first n cells of the stack, pushing the rest of the packets already in the stack upwards. Just like the tree in figure 3, the stack in figure 4 is only a visualization aid and it does not need to be realized by any station. Stations require only a counter to implement the stack algorithm.

Search for collided packets in a tree-search algorithm is done along the branches of the tree, and for each collision the tree splits one more time. In a stack algorithm, search is done by looking at the contents of the cells of a stack. The bottom cell, cell number 0, is the transmission cell; it is also the cell at which all the new packets enter the system. In

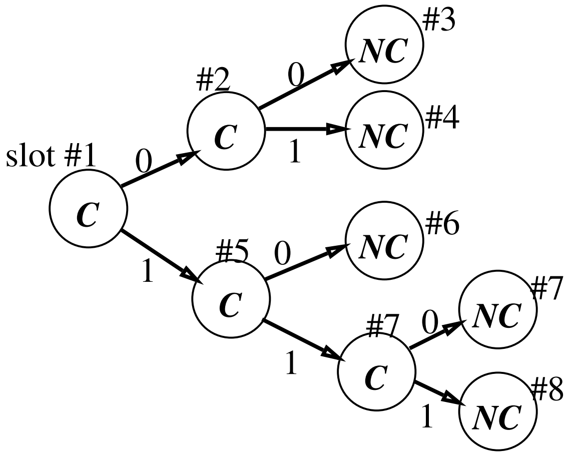

Figure 3: A binary tree-search collision resolution algorithm with binary C/NC feedback information; the CRI is 8 slots long.

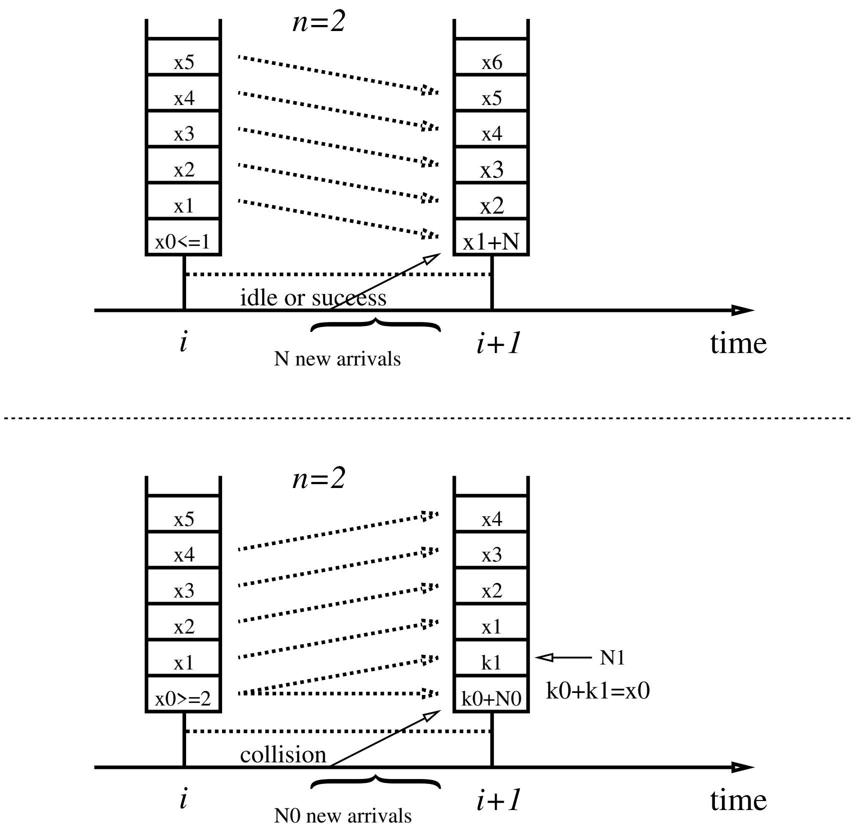

Figure 4: A stack collision resolution algorithm with binary C/NC feedback information and splitting parameter $n = 2$.

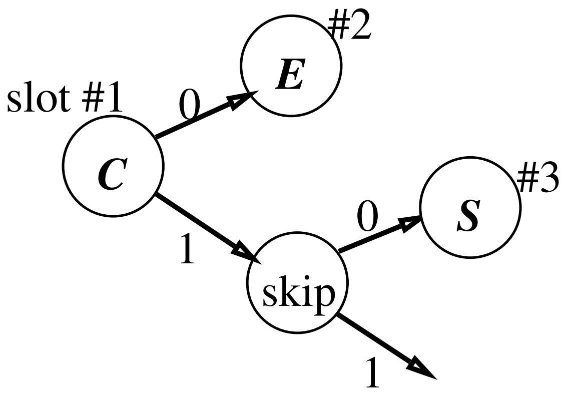

Figure 5: Skipping sure collisions in ternary sc c/e/s feedback tree-search collision resolution algorithms.

both cases, after each splitting that follows a collision, each subgroup waits for its turn to transmit after the previous subgroup finishes its transmission.

If a collision resolution starts with a collision and the next $(n-1)$ slots are empty, it is guaranteed that the following slot (the n-th one) will be a collision. For example, in figure 3 if slot $\# 6$ were actually an empty slot, then one could deduct that slot $\# 7$ would be a collision, before even the transmission in slot $\# 7$ takes place. This sure to occur collision can be avoided if the empty slot $\# 6$ is so identified. Hence, if *ternary* C/E/S *feedback* is used sure to occur collisions can be skipped, see figure 5. Similarly, ternary feedback stack algorithms can be constructed.

Although, the performance advantage of the ternary feedback information is apparent, the ternary feedback implementation is less robust in the presence of feedback errors. For example, figure 6 demonstrates how a CRI in a tree-search algorithm never ends due to a single empty to collision error in the feedback. Measures against such deadlocks can be taken, e.g., allow only a maximum of successive empty slots and then force a CRI termination, but then the advantages of this more elaborate scheme start to become insignificant.

4.2 Obvious vs. non-obvious first transmission rule

Blocked-access (tree-search) RAAs also require a first transmission rule for allowing the blocked packets access to the network. Recall that free-access (stack) RAAs allow new packets into the network immediately, actually at the beginning of the very next slot. So their first transmission rule is trivial and they will not be considered further in this subsection.

The *obvious* first transmission rule simply states that all new arrivals during a CRI attempt their first transmission as soon as the CRI terminates, see figure 7. In a totally distributed system, stations can identify endings of CRIs only if they have observed the sequence of feedback information from the beginning of the operation of the system (even if they never had anything to transmit)! This requirement can be relaxed in the presence of a

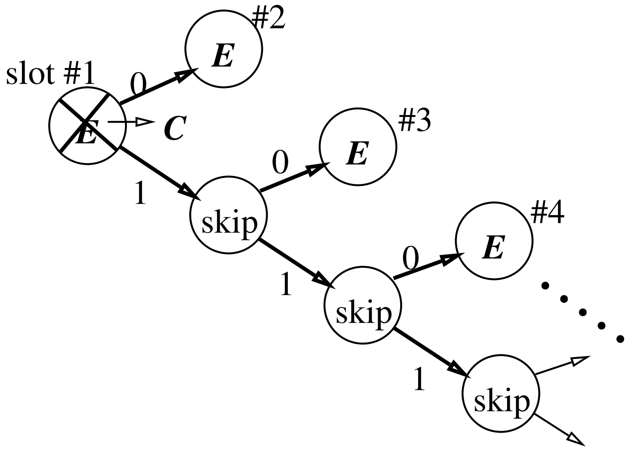

Figure 6: The reaction of the ternary feedback tree-search collision resolution algorithm to an E→C feedback error.

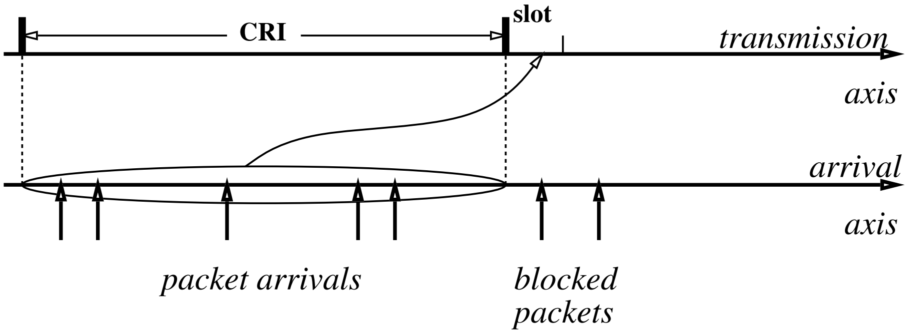

Figure 7: The *obvious* first transmission rule for blocked access tree-search collision resolution algorithms.

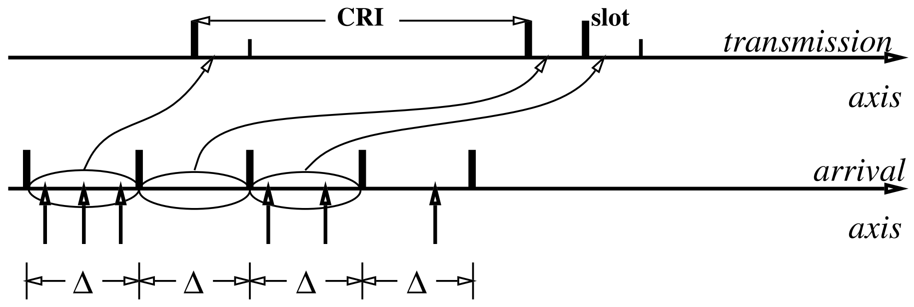

Figure 8: The *non-obvious* first transmission rule for blocked access tree-search collision resolution algorithms.

monitor station, e.g., the headend station in the 802.14 network, that monitors the system from the beginning of its operation. The monitor station can simply signal the stations when a CRI has terminated.

As it is apparent from figure 7, with the obvious first transmission rule, the number of packets transmitting in the first slot of a new CRI, and hence its length, is positively correlated to the length of the previous CRI. Hence, large CRIs are followed by large CRIs as well, a condition that may persist for long time resulting in low system performance. Performance may increase (and it indeed does) if the arrival axis is "decoupled" from the transmissions axis. This is what the *non-obvious* first transmission rule does, as illustrated in figure 8. Only packets that arrived during the next time slice of size Δ are accepted into the system. The parameter Δ can be selected for optimum system performance.

The non-obvious first transmission rule requires (in the case of a monitor station) to notify the stations not only of the end of a CRI, but also of the next arrival interval to be "inspected," the so called *enabled interval.* The latter piece of information may require several bits of overhead depending on (a) the desired granularity in recording time; and (b) the maximum possible time to be recorded, prior to restarting counting time from zero.

In closing this section, we need to note that in tree-search algorithms, the splitting in subgroups can be done not only by flipping dices, but also by appropriately splitting the enabled interval. This much harder to implement version of the tree-search algorithms can achieve *absolute first-in-first-out* (FIFO) packet transmission ordering, in contrast to the *quasi*-FIFO service of the dice splitting, tree-search RAAs, and the *quasi*-LIFO service of the stack RAAs. The most efficient implementable RAA, albeit its complexity, belongs to this class of time-splitting tree-search RAAs. The algorithm is known as the "0.487 algorithm" for its throughput performance [3].

5 Performance comparisons

Here we will summarize some performance results for various RAAs. Most of the results have been already published in the literature, e.g., [1, 3, 5, 6, 7, 9, 12, 13], and references

RAA	capacity	feedback	comments
ALOHA	0	ACK	easiest to implement, unstable
binary tree-search	0.3465	C/NC	obvious first transmission rule
stack ($n = 2$)	0.3601	C/NC	free access
stable "ALOHA"	0.3678	>C/NC	the ALOHA name is a misnomer
tree-search	0.3752	C/E/S	obvious first transmission rule
stack ($n = 3$)	0.4015	C/NC	free access
tree-search	0.4294	C/NC	non-obvious first transmission rule
tree-search	0.4622	C/E/S	non-obvious first transmission rule
tree-search	0.4872	C/E/S	elaborate time splitting algorithm
tree-search	0.5323	k	k: number of packets collided

Table 1: Throughput for various RAAs.

therein. For these results, the following system assumptions are made:

- The time axis is slotted.
- Stations generate (fixed size) packets each one big enough to fit in a slot.
- The system is comprised of a large (unspecified) number of station.
- The cumulative packet generation process constitutes a Poisson process with intensity λ packets per slot.

The above system and traffic model, frequently referred to as the *infinite Poisson population model* (IPP model), has been the basis of analysis and comparisons for almost any analytical study published on this topic; any throughput results presented far in this contribution are also under the IPP model.

The first, most important, and easier to derive analytically (without this implying that it is indeed easy) performance metric is the system throughput (under the IPP model). Table 1 summarizes this result. Clearly, all high efficiency RAA are ternary feedback, tree-search algorithms with the non-obvious first transmission rule. The most efficient RAA with a simple first transmission rule is the very simple stack algorithm with splitting parameter $n = 3$ that operates with the more robust binary C/NC feedback.

Table 2 shows the performance of various RAAs under the following "stress" scenario:

- A large number N of packets are all at once presented in a previously empty system. Calculate the average number of slots it will take to transmit all N packets assuming that no new packets enter the system. In other words, study the average length of a CRI, L_N, as $N \to \infty$.

Note that under this stress scenario, the first transmission rule is of no consequence. The results for the tree-search and stack algorithms have been taken from [10]. As expected,

RAA	$\alpha = L_N/N$	L_{2000} (slots)
binary tree-search (C/E/S)	≈ 2.664	$\approx 5,328$
ideal (stable) "ALOHA"	≈ 2.718	$\approx 5,436$
stack ($n = 3$) (C/NC)	≈ 2.731	$\approx 5,462$
tree-search (C/NC)	≈ 2.885	$\approx 5,770$
implementable stable "ALOHA"	> 3.2	$\approx 6,400$
pure ALOHAs	very big	$> 10,000$

Table 2: Stress performance for various RAAs.

delay (slots)	ideal "ALOHA"			obvious tree-search (C/NC)		
	$\lambda = 0.10$	0.20	0.30	$\lambda = 0.10$	0.20	0.30
1	89.27	75.78	56.08	85.01	66.06	37.02
2	2.28	3.92	3.32	3.28	5.52	5.09
3	1.13	2.00	1.73	3.99	6.72	6.46
4	2.41	3.50	2.37	1.94	3.83	4.35
5	1.42	2.45	1.98	1.73	3.71	4.51
6	0.89	1.78	1.66	1.06	2.59	3.57
7	0.62	1.46	1.45	0.87	2.25	3.41
8	0.45	1.17	1.34	0.54	1.66	2.94
9	0.35	0.94	1.20	0.42	1.46	2.74
10	0.25	0.78	1.10	0.29	1.10	2.46
> 10	0.93	6.22	27.77	0.97	5.10	27.45
μ_d	1.47	2.94	19.15	1.55	2.81	9.01
σ_d	1.89	5.87	50.06	1.80	3.88	13.02

Table 3: Delay percentage histograms of various RAAs.

once again a ternary feedback, tree-search RAA achieves the shortest CRI with the ideal non-implementable "ALOHA" and the binary feedback stack algorithm close behind. The implementable ALOHA algorithms take by far the longest time to resolve the collisions in our stress scenario.

Finally tables 3 and 4 summarize some simulation results regarding delay histograms for few RAAs [4]. The delay of a packet is the time in slot between the slot following its arrival until its transmission ends; μ_d is the average delay observed and σ_d is its standard deviation.

delay (slots)	stack, $n=2$			stack, $n=3$		
	$\lambda = 0.10$	0.20	0.30	$\lambda = 0.10$	0.20	0.30
1	88.34	73.41	46.10	89.17	75.47	56.72
2	2.52	4.84	7.46	2.16	4.12	6.24
3	2.83	5.06	6.97	2.28	4.35	6.27
4	1.44	2.76	4.88	2.39	4.48	6.09
5	1.27	2.72	4.36	0.90	1.90	3.07
6	0.89	1.81	3.11	0.82	1.78	2.89
7	0.74	1.59	2.85	0.68	1.65	2.70
8	0.40	1.09	2.16	0.35	0.88	1.49
9	0.38	1.03	2.03	0.27	0.87	1.44
10	0.25	0.66	1.51	0.26	0.80	1.41
> 10	0.94	5.03	18.57	0.72	3.70	11.68
μ_d	1.47	2.83	10.88	1.42	2.41	5.88
σ_d	1.88	5.65	34.68	1.68	4.32	15.73

Table 4: Delay percentage histograms of various RAAs.

6 Concluding remarks

We have reviewed some of the most known RAAs. Many more (hundreds of them) have been proposed but almost all of them borrow heavily from the class of RAAs presented here.

The most widely commercially available RAA is the exponential back-off flavor of ALOHA which is used in millions of Ethernet LANs. We have shown that under the IPP model, the family of ALOHA algorithms are unstable for any arrival rate. It has been conjectured (although not yet rigorously proven) that there can be no acknowledgement only based algorithm that is stable [2]. We have then presented the family of stable "ALOHA" algorithms that under the IPP model truly achieve a throughput of e^{-1}, but the stations (or at least a monitor station) need to be able to distinguish between collision and non_collision slots. Since these algorithms use a dynamically changing retransmission probability p, like some pure ALOHA algorithms, these stable algorithms are still (erroneously) grouped within the ALOHA family of algorithms. Yet, for the stable "ALOHAs," the update algorithm for p requires much more system information than what is available on the acknowledgement only ALOHA algorithms.

We also reviewed the class of the splitting algorithms and presented some of their flavors. The splitting algorithms are stable and can achieve throughputs much higher than the stable "ALOHA," even with the same feedback information. Implementable tree-search algorithms have been proposed achieving throughputs between 0.346 and 0.487 with increasing degree of complexity. There are blocked-access (tree-search) and free-access (stack) algorithms. In the former, new packets are not allowed into the system, prior to the termination of any on-going CRI, while in the later new packets can jump-in right away. Blocked-access algorithms provide a quasi-FIFO packet transmission ordering, while the free access ones provide a quasi-LIFO packet transmission ordering. Our simulation results have shown that the LIFO service does not cause major performance degradation. In addition in an 802.14 network, not all transmissions are to be made through random access, since many of them can be through reservations under the control of the headend station. Hence, even fewer packets transmissions could experience the effects (if any) of a mostly LIFO service provided by stack algorithms. Also, all RAAs are fair algorithms, i.e., they never give consistent preference to any particular station. So, over time all stations are treated equally well by either the FIFO or LIFO service. Tree-search algorithms require a first-transmission rule for the blocked packet, and tree-search algorithms with non-obvious first transmission rule achieve the highest known throughputs, but they are harder to implement. Also, we have shown that the use of ternary C/E/S feedback may cause deadlocks in the presence of feedback errors.

We cannot ignore the wide deployment of ALOHA related products. Yet, we cannot either ignore their stability problems. In current installations, ALOHA is used either with small Eithernet networks, with no more than few tens of stations, that also employ carrier-sense to probe first for any activity on the transmission medium prior to their own transmission, or with slow speed packet radio and satellite networks. In these networks, the time needed to reach the unstable region of ALOHA is large enough to operate them with acceptable reliability. Unfortunately, in the 802.14 network none of these situations are valid any more. The channel speeds are in the Mb/sec region, as opposed to Kb/sec for the radio networks, while carrier-sense is impossible. Moreover, the network must support a wide range of station populations reaching to a few thousands. Under these conditions, the use of the unstable ALOHA cannot be relied upon any more. Some other inherently stable RAA needs to be used instead.

Of all the RAAs presented, the free-access stack algorithms appear most appealing to the author. They require no signaling information for notifying the end of a CRI; they require no timing information for which portion of the arrival axis needs to be inspected next; they are simple to implement, only a single counter is required which is updated according to feedbacks received at each slot; they operate quite efficiently with the very robust C/NC feedback; and achieve a stable throughput that is higher than any other known RAAs, with the exception of the blocked-access tree-search algorithms with the harder to implement non-obvious first transmission rule.

The IBM MLAP [11] uses a flow-controlled RAA, called START, derived from the class of stack algorithms. START comes with enhancements that allows it to switch between (quasi-)blocked- and free-access modes depending on traffic demand, and to operate over a dynamically changing number of contention slots per fixed size time intervals, referred in MLAP as blocks.

References

[1] N. Abramson (Ed.), *Multiple Access Communications: Foundations for emerging technologies*, IEEE Press, 1993.

[2] D. J. Aldus, "Ultimate instability of exponential back-off protocol for acknowledgement-based transmission control of random access communication channels," *IEEE Trans. on Inform. Theory*, March 1987.

[3] D. Bertsekas and R. Gallager, *Data Networks, (2nd ed.)*, Prentice Hall, 1992.

[4] C. Bisdikian, *Delay analysis of a class of limited sensing algorithms for a multiple access channel*, M.Sc. Thesis, Univ. of Connecticut, Storrs, CT, 1985.

[5] J. I. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inform. Theory*, September 1979.

[6] L. Georgiadis and P. Papantoni-Kazakos, "A collision resolution protocol for random access channels and energy detectors," *IEEE Trans. on Commun.*, November 1982.

[7] *IEEE Transactions on Information Theory*, special issue on "Random-Access Communications," March 1985.

[8] L. Kleinrock, *Queueing Systems, vol 2: Computer applications*, John Willey and Sons, 1976.

[9] J. L. Massey, "Collision-resolution algorithms and random access communications," in *Multi-user Communications*, G. Longo (Ed.), *CISM Courses and Lecture Series*, Springer-Verlag, 1981.

[10] P. Mathys and P. Flajolet, "Q-ary collision resolution algorithms in random access systems with free or blocked channel access," *IEEE Trans. on Inform. Theory*, March 1985.

[11] B. McNeil, R. Norman, and C. Bisdikian, "Formal proposal for 802.14 MAC protocol (part 2 of 2): MLAP (MAC level access protocol)," contribution to the *IEEE802.14 WG* no. IEEE802.14-95/158, November 1995.

[12] L. Merakos and C. Bisdikian, "Delay analysis of the n-ary stack algorithm for a random access broadcast channel," *IEEE Trans. on Inform. Theory*, vol. 34, no. 5, Sept. 1988.

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