Conflict Resolution Algorithms and their Performance Analysis

Mart L. Molle *George C. Polyzos*

Computer Systems Research Institute
University of Toronto
Toronto, Canada M5S 1A4

Dept. of Computer Science and Eng.
University of California, San Diego
La Jolla, California 92093-0114

Technical Report Number CS93-300

July 1993

Abstract

Multiple Access protocols are distributed algorithms that enable a set of geographically dispersed stations to communicate using a single, common, broadcast channel. We concentrate on the class of Conflict Resolution Algorithms. This class exhibits very good performance characteristics for ''bursty'' computer communications traffic, including high capacity, low delay under light traffic conditions, and inherent stability. One algorithm in this class achieves the highest capacity among all known multiple-access protocols for the infinite population Poisson model. Indeed, this capacity is not far from a theoretical upper bound. After surveying the most important and influential Conflict Resolution Algorithms, the emphasis in our presentation is shifted to methods for their analysis and results of their performance evaluation. We also discuss some extensions of the basic protocols and performance results for non-standard environments, such as Local Area Networks, satellite channels, channels with errors, etc., providing a comprehensive bibliography.

1. Conflict Resolution Based Random Access Protocols

The ALOHA protocols were a breakthrough in the area of multiple access communications.[1] They delivered, more or less, what they advertized, i.e., low delay for bursty, computer generated traffic. They suffer, however, from stability problems and low capacity.[2] The next major breakthrough in the area of multiple access communications was the development of random access protocols that resolve conflicts algorithmically. The invention of Conflict Resolution Algorithms (CRAs) is usually attributed to Capetanakis [Capet78, Capet79, Capet79b], and, independently, to Tsybakov and Mikhailov [Tsyba78]. The same idea, but in a slightly different context, was also presented, earlier, by Hayes [Hayes78]. Later, it was recognized [Berge84, Wolf85] that the underlying idea had been known for a long time in the context of *Group Testing* [Dorfm43, Sobel59, Ungar60].

Group Testing was developed during World War II to speed up processing of syphilis blood tests. Since the administered test had high sensitivity, it was suggested [Dorfm43] that many blood samples could be pooled together. The result of the test would then be positive if, and only if, there was at least one diseased sample in the pool, in which case individual tests were administered to isolate the diseased samples. Later, it was suggested that, after the first diseased sample was isolated, the remaining samples could again be pooled for further testing. The beginning of a general theory of Group Testing can be found in [Sobel59], where, as pointed out in [Wolf85], a tree search algorithm is suggested, similar to the ones we present in section 1.2.

The first application of Group Testing to communications arose when Hayes proposed a new, and more efficient, polling algorithm that he named *probing* [Hayes78]. Standard polling schemes are unacceptable for large sets of bursty stations because the overhead is proportional to the number of stations in the system, and independent from the amount of traffic. Hayes' main idea was to shorten the polling cycle by having the central controller query subsets of the total population to discover if these subsets contain stations with waiting packets. If the response is negative, the total subset is ''eliminated'' in a single query. If the response is positive, the group is split into two subgroups and the process is continued, recursively, until a single active station is polled. This station is then allowed to transmit some data, which does not have to be in the form of constant size packets. Clearly, this is a reservation protocol. In subsequent papers Hayes has also considered direct transmission systems. Notice that the controller receives feedback in the form *something — nothing* (at least one station, or no station with waiting packets).

1.1. Basic Assumptions

The protocols that will be presented in this section have been developed, and analyzed, on the basis of a set of common assumptions[3] that describe a standard environment that is usually called an *ALOHA-type channel.*

1. *Synchronous (slotted) operation:* The common-receiver model of a broadcast channel is usually implicitly assumed. Furthermore, messages are split into packets of fixed size. All transmitters are (and remain) synchronized, and may initiate transmissions only at predetermined times, one packet transmission time apart. The time between two successive allowable packet transmission times is called a *slot* and is usually taken as the time unit. Thus, if more than one packet is transmitted during the same slot, they are ''seen'' by the receiver simultaneously, and therefore, overlap completely.

2. *Errorless channel:* If a given slot contains a single packet transmission, then the packet will be received correctly (by the common receiver).

---

[1] For an introduction to the area of multiple access communications see the books by Bertsekas and Gallager [Berts92, chapter 4] and [Rom90]. Actually, chapter 4 of [Berts92] and chapter 5 of [Rom90] also present good expositions of Conflict Resolution Algorithms.

[2] If no special control is exercised to stabilize the protocols, the term capacity must be taken in the ''broader'' sense of maximum throughput maintained during considerable periods of time, since the true capacity is zero [Fergu75, Fayol77, Aldou87]. However, having to stabilize the protocols detracts from their initial appeal that was mainly due to their simplicity.

[3] Some of the protocols can operate with some of the assumptions weakened. When this is the case we point it out during their presentation. In section 6 we discuss protocols and analyses techniques that weaken or modify some of these assumptions.

3. *Destructive interference among simultaneous transmissions:* If two or more packets are transmitted during the same slot, the receiver is unable to correctly receive any of the packets. We call this event a *conflict* or *collision.*[1] Thus, the state of the channel in any given slot is one of:
    i. *idle:* when no packet transmission is taking place,
    ii. *success:* when a single packet transmission is taking place (which, therefore, is successful), or
    iii. *conflict:* when two or more packets are being transmitted simultaneously (and none is received correctly).
4. *Immediate feedback:* Stations are informed of the state of the channel at the end of the current slot (before they need to make decisions for their actions during the following time slot).
5. *Binary or ternary feedback:* The feedback that stations receive at the end of each slot, informing them about the state of the channel during the slot that just ended, is *ternary*, denoted by, *i*, *s*, and *c* for idle, success, and conflict, respectively. Some of the algorithms make use only of *binary* feedback of the type ''conflict—no-conflict'' (CNC).[2]
6. *Errorless feedback:* It is assumed that the feedback obtained by all stations represents the true state of the channel.
7. *Availability of arbitrary resolution time-stamps:* Some of the protocols require stations to time-stamp packets with their arrival or generation time at infinite or arbitrary resolution so that any two packets can be distinguished on the basis of this characteristic alone.[3]
8. *Full sensing:* The complete past history of the channel is available to all stations. This is usually taken to imply that all stations must start the protocol simultaneously (monitoring the channel continuously, or just keeping track of the channel history from the feedback they receive) and no new stations may join while the system is in operation. However, we do present various *limited sensing* algorithms that do not have this requirement, which is obviously a big advantage.

Note that we have not yet made any assumptions about the number of stations in the system. There are two directions that have been followed here. The first alternative is to consider specific finite populations, of size *M*. The second is to consider an idealized infinite population model.

In *finite population* models it is usually assumed that each station, *s*, generates new packets according to a Bernoulli process with probability q_s — per cycle [Berge84], or per slot, but with only the first packet in a cycle being accepted [Capet79], or per window [Polyz87]. Furthermore, usually, only the homogeneous case, where $q_s = q$, for all *s*, is considered, and the population size is assumed to be a power of two. In [Polyz87], however, the general non-homogeneous case is treated (and therefore, no restrictions on the population size are imposed). The above Bernoulli models can also be thought of as blocked Poisson processes, i.e., Poisson processes from which only the first, if any, packet generated in a time period (cycle, or window) is accepted. It is generally assumed that the other packets that might have been generated are discarded. In [Polyz87] an approximate delay analysis is provided for the case that these packets are buffered; a more comprehensive discussion of these alternatives is provided in that paper.

---

[1] The most widely used term is probably *collision. Contention*, is also used as a synonym. Interestingly, all these terms leave the acronym Conflict Resolution Algorithms (CRAs) invariant. There is even a fourth ''interpretation'' of the acronym 'CRA' in [Berge83]: *Confusion* Resolution Algorithms!

[2] There are two more combinations: ''something—nothing'' feedback that we have seen used in probing, and ''success—failure,'' where a station is notified of its success, but cannot distinguish between idle and conflict. We discuss protocols based on these two types of feedback in section 6.

[3] This capability can be simulated through the use of a finite resolution time-stamp facility and a pseudorandom number generator (that provides the higher resolution).

If we let $M \to \infty$, and $q \to 0$, so that $Mq = x$ (a constant), all the above models converge to the *infinite population* Poisson model. With this model we can assume that each packet arrives to a ''new'' station that never generates another packet. Thus, all the ''buffering'' problems and the peculiarities of finite population models ''magically'' disappear. The infinite population model permits a uniform characterization of the CRAs themselves, instead of the combination of the CRA and the specific traffic environment in which it operates. Furthermore, results obtained through the infinite population model are pessimistic estimates of performance metrics for finite population systems, since any system can simulate the infinite population model by treating the packets that arrive at the same station independently from one-another, causing some incestuous collisions.

We denote by $F(x)$, $\Phi(x)$, and $\Psi(x)$ the probabilities of no-conflict, idle, and a single successful packet transmission, respectively. For the infinite population Poisson model with parameter x we have

$$F(x) = (1+x)\,e^{-x}\,, \quad \Phi(x) = e^{-x}\,, \quad \text{and} \quad \Psi(x) = x\,e^{-x}\,.$$

We also use the notation $\bar{f} \stackrel{\Delta}{=} 1 - f$, for any f, throughout this paper.

1.2. Conflict Resolution Algorithms

The term CRA refers, broadly, to any protocol that handles conflicts by resolving them algorithmically, and, in a stricter sense, to that part of the protocol that specifically resolves a conflict after it arises. To form a complete random-access protocol a CRA (in the strict sense) must be combined with a Channel Access Algorithm (CAA), which specifies when new packets may join a CRA. It is not always easy to isolate the conflict resolution from the channel access algorithm; however, for most widely known protocols it is possible. When this is the case, both the description of the protocols, and their analysis is greatly simplified. We say that a CRA protocol is *separable* if no packet given to the CRA by the CAA is ever returned undelivered for rescheduling.

1.2.1. The Standard Tree Algorithm

The first CRA we present is usually referred to as the standard, or basic, or the Capetanakis, or the Capetanakis-Tsybakov-Mikhailov tree algorithm [Capet78, Tsyba78, Capet79, Capet79b]. It is not identical to the one initially proposed by Capetanakis, it seems, however, more natural than Capetanakis' since it is based on a *preorder* traversal of the tree [Knuth68]. Therefore, it has been adopted in most subsequent works. We refer to it as the Standard Tree Algorithm (STA) and present the original Capetanakis algorithm later.

The algorithm can easily be described recursively with the help of an active population set — the set of stations that are allowed to access the channel at a particular time. At the beginning of a Conflict Resolution Interval (CRI), or epoch, the active set is determined by the channel access algorithm. This active set is enabled at the first slot of the CRI and all packets belonging to the active set get transmitted. If the outcome is idle or success, the active set is said to have been satisfied and the CRI ends at this point. However, if there is a conflict, the algorithm splits the active set into two subsets, and we say that the conflict is resolved when both of these subsets have been satisfied, one after the other. To visualize the procedure (see Fig. 1), one can use a binary tree (hence the name Tree Algorithm), with the root being the first slot of the CRI. If this slot is idle, or results in a successful transmission (no-conflict), the root is a terminal node. Otherwise, the root is a non-terminal node with two subtrees emanating from the root. The participants in the initial conflict are split into two groups and reassigned to one of these subtrees. The question of how the active set is split into subsets, in a distributed manner, is addressed later. For the moment, let us adopt one such splitting scheme: assume that, when required, each station ''tosses a two-sided coin'' and joins the first (left) subset with probability p or the second (right) one with probability $\bar{p}$.

In Fig. 2, we present a stack interpretation of the STA. Each station maintains two stack pointers, for the local and the system stack, respectively. The former is used by the CRA to schedule packet transmissions from the given station, while the latter is used by the CAA to determine CRI boundaries. At the beginning of the CRI all stations start with their packets at the top of each stack. Then, depending on the feedback (and, in the case of the local stack, the splitting scheme that gets invoked after a conflict), the stacks might get pushed or popped. A station transmits only when its packet is at the top of the local stack. For the STA, the system stack gets pushed one level after every conflict while the local stack does so only if the station flips a ''1'' (meaning that it decided to join the second subtree after the conflict) or did not participate in the last conflict. Both stacks get popped after a non-conflict. Note that the feedback used by the STA is only binary, of the type *conflict — no-conflict*. The depths of these stacks (and

that is the only thing that matters) are used in the implementation of the protocol. Note that these stacks might get infinitely deep for the STA (and an infinite population). Other algorithms described later limit the depth of these stacks to a few cells.

The original Capetanakis algorithm [Capet78] did not traverse the tree in the same way as the algorithm we have described. Instead, after a collision, both successors were visited together (say, left and then right) using a pair of slots; then, if a collision had occurred at the left node, the successors of that node were visited in another pair of slots, etc. Capetanakis chose this unusual traversal algorithm because he was mainly interested in the satellite channel, where the propagation delay is very large compared to the transmission time of a packet (for usual packet sizes and channel transmission speeds). The algorithm that Capetanakis proposed had the advantage of parallelism. In general, the next algorithmic step in a CRA depends on the outcome of the current step — which, in a satellite channel, is unavailable for a long time. Capetanakis recognized that (with the STA) *both* successors to a conflict node will be visited eventually, regardless of the outcome of either one. Thus, by visiting both nodes at the same ''algorithmic step,'' as Capetanakis put it, the algorithm can advance at the rate of two nodes per round trip propagation time instead of one that the preorder traversal algorithm achieves. Apart from this, the STA and Capetanakis' version have essentially the same characteristics and thus are seldom distinguished in the literature.

1.2.2. Addressing Schemes

For finite population systems there are at least three ways to split the active set after a conflict. First, one can assume that all stations have binary addresses preassigned to them. Each time there is a conflict, the station uses the corresponding bit of its address to decide which sub-tree to join (see Fig. 1). We call this scheme *deterministic addressing.* Second, the station may join one of the sub-trees at random. The station must in this case generate a (pseudo-)random bit each time it is involved in a conflict. The sequence of these bits can be thought of as (and plays the role of) an address, obtained at random. We call this scheme *random addressing.* A third approach is to use the arrival time of the packet to decide which sub-tree to join. The packets must, therefore, have been time-stamped with their arrival time. The channel access algorithm specifies an initial time interval at the beginning of the CRI, and all packets with time-stamps in the specified interval get transmitted. After a conflict, the station joins the left sub-tree if its packet arrived during the first half of the last tried interval and transmits in the next slot; otherwise, it joins the right interval and defers transmission. We refer to this addressing method as *arrival-time addressing.* Finally, it is easy to extend these addressing schemes to biased and d-ary splitting with $d > 2$.

For finite population systems deterministic addressing leads to finite conflict resolution trees, which is not the case with random and arrival-time addressing, since there is a small probability that two stations might pick the same finite random sequence or have their packets arrive arbitrarily close in time. It has been shown by Capetanakis [Capet78] that in the finite population case, deterministic addressing is more efficient. Furthermore, with binary deterministic addressing and gated channel access, the maximum throughput of Tree Algorithms is always > 0.5, since the number of leaves of a binary tree (which correspond to successful transmissions) are one more than the internal nodes, and thus more than half of the total.[1]

For the infinite population model, deterministic and random addressing cannot be distinguished. Also, arrival-time addressing leads to the same CRI length statistics, but the order of successful packet transmissions is FCFS, which decreases the variance of the packet delay as compared to random addressing.

1.2.3. The Modified Tree Algorithm

When a conflict is followed by an idle slot, the outcome of the next slot in the STA is predetermined to be another conflict. Therefore, it serves no purpose to let these transmissions proceed and waste this slot. Instead, one can modify the standard algorithm to skip this step and proceed directly to the next level of the tree. As a result, this algorithm is referred to as the Modified Tree Algorithm (MTA) or the ''level skipping'' algorithm and it is due to Tsybakov and Mikhailov [Tsyba78] and, independently, to Massey [Masse81]. Notice that ''level skipping'' is not possible with Capetanakis' original pairwise traversal algorithm, since the feedback arrives too late — after the

[1] For d-ary splitting maximum throughput is greater than $1 - 1/d$. As $d \to M$ the system resembles more and more Time Division Multiple Access (TDMA). This is also what happens with the ''Dynamic'' Tree Algorithm described in section 1.3.2.

transmissions for the node that is to be ‘‘skipped’’ have taken place.

This simple modification increased the capacity of the algorithm with gated channel access (to be described later) from 0.346 — which is below the capacity of (stabilized) Slotted ALOHA — to 0.375. However, the modification has an unexpected side effect: the new algorithm can deadlock in the presence of feedback errors. Although we have explicitly excluded feedback errors in our assumptions, it is interesting to explore the situation. If a single idle slot is, incorrectly, interpreted as a conflict, the active set, though empty, will be split in two. The next slot will likewise be empty and thus, according to the modified algorithm, we will skip the ‘‘predictable conflict’’ and split the active set again, and again... The result is deadlock. What is very interesting is that the STA does not suffer from this problem. If the same scenario is presented to the STA the third node will not be skipped, and since it will be empty, the CRI will end there. The STA is therefore more robust than the MTA.

Of course, once the problem was identified, the remedy was easy: ‘‘level skipping’’ should be attempted only a finite, predetermined number of times in succession (this limit should, of course, be a constant of the protocol so that stations remain synchronized) [Ryter80]. Moreover, if a finite population system with deterministic addressing is considered, all conflict resolution trees should be finite and the problem can easily be corrected.

1.2.4. Ternary or *d*-ary Splitting

Although Tsybakov and Mikhailov [Tsyba78] described CRAs through *d*-ary trees, they did not analyze them. Mathys [Mathy84] reintroduced and analyzed them. The generalization is easy, and interesting for two reasons. First, splitting into more than two subsets increases the capacity of some protocols for the standard environment. For example, it is shown in [Mathy85], that *ternary* splitting is optimal for the STA with gated or free channel access (defined below), and for the MTA with free access. And second, it can increase the capacity for all protocols if carrier sensing and/or collision detection techniques are available. Note that with the MTA, a node is ‘‘skipped’’ only if all $d-1$ previous slots at the same level were empty.

1.2.5. Tree Pruning

A second important observation about Tree Algorithms was made by various researchers [Galla78, Tsyba80, Ruget81] and led essentially to the best known protocol to date. Note that if the first (left) subset after a conflict also results in a conflict, we have no information whatsoever about the number of packets in the second (right) subset. Furthermore, if the initial active set had about the ‘‘correct’’ number of packets, then the second subset cannot (since it only contains ‘‘half’’ that number). Therefore, if a sub-CRI is initiated, it will be resolved in sub-optimal time.

Instead, it is more efficient to drop this set from further consideration in the current CRI and to combine it with some other yet unexamined traffic in order to obtain the ‘‘correct’’ expected number of packets in the set before initiating the CRA. However, this combination can easily be achieved only with arrival-time addressing and a form of window channel access, leading to what is known as the FCFS 0.487 Algorithm. We describe this protocol in more detail in section 1.4 after we discuss channel access algorithms in section 1.3. Note that with ‘‘tree pruning’’ the depth of the stack is bounded.

1.2.6. The Three-Cell Algorithm

The Three-Cell Algorithm (3CA), which was mentioned in [Tsyba85] and extensively studied in [Polyz93], incorporates both the ‘‘level skipping’’ and ‘‘tree pruning’’ rules from the FCFS 0.487 Algorithm but is easier to analyze. The key difference from the 0.487 Algorithm is that all ‘‘pruned’’ right sub-sets (if any) accumulate in the third cell of the stack — hence the name 3CA — instead of being dropped from the current CRI (as in the 0.487 Algorithm) or stored individually in the stack (as in the MTA). Thus, all packets in a given initial set must be transmitted successfully before the end of the CRI. Remarkably, this simplification only reduces the capacity of the 3CA by 1.6 % in comparison to the 0.487 Algorithm.

1.2.7. The Two-Cell Algorithm

An even simpler limited stack depth algorithm inspired by the ''tree pruning'' rule is the ''Two-Cell'' Algorithm (2CA), mentioned in [Tsyba85] and extensively studied by Paterakis and Papantoni-Kazakos [Pater89]. It is similar to the 3CA we have just described, but does not differentiate between idle slots and successful transmissions. Thus, only binary feedback of the type *conflict—no-conflict* is required (similarly to the the STA case).[1] This makes it simpler but similar to the STA and also limits its capacity in the standard environment to the level of the STA. However, this is an important algorithm because it is very robust in the presence of channel errors (attaining higher capacities than the STA for channels with errors). It can also be modified to operate with limited sensing.

1.3. Channel Access Algorithms

Channel access algorithms specify when packets may join a CRA — in most cases, after the current CRI has ended. Since it is relatively easy to obtain statistics for various performance metrics related to CRIs, such as their length, and the delay that packets experience in a particular CRI, the complexity of the channel access algorithm is usually what determines the level of difficulty of the analysis. Unfortunately, the best algorithm known is also a complex one.

1.3.1. Gated Channel Access

This is the first channel access algorithm proposed by Capetanakis [Capet78] and Tsybakov and Mikhailov [Tsyba78]. Massey [Masse81] calls it the *obvious* channel access algorithm. According to this algorithm, during the resolution of a conflict no new packets, i.e., packets that did not participate in the initial conflict, may join the CRI. Instead, packets are buffered and all packets that arrived during the current CRI are transmitted in the next CRI. The following analogue can be drawn: To join the CRA packets have to go through a gate. The gate opens only at the beginning of a new CRI allowing all waiting packets in. Then, before the first transmission takes place, the gate closes until the end of the CRI. Hence, the name Gated Access (GA).

In the finite population case with deterministic addressing, there can be at most one packet per station (more specifically, per address) per CRI — otherwise it is not possible to resolve the conflict. Furthermore, to avoid the problems of ''full'' buffering, Capetanakis considered a *two buffer* model. That is, each station has only two buffers. One is reserved for the packet (if any) that participates in the current CRI, while the other is reserved for a new packet that might be generated (or arrive) as the CRI proceeds.

With gated access there is correlation between CRIs, since all packets generated during one CRI are transmitted in the following one, and thus, a long CRI is likely to be followed by another long CRI and vice versa. This correlation results in considerable difficulty in the analysis and, more importantly, bad performance. For example the STA with GA, though stable, offers lower capacity than (stabilized) Slotted ALOHA. The main problem with GA, from the performance point of view, is that it encourages many packets to be transmitted at the same time in the first slot after the end of a long CRI, even though the channel can only support one packet at a time. This causes extra work for the CRA, which must split the initial set many times to get any successful transmissions.

1.3.2. The ''Dynamic'' Tree Algorithm

Capetanakis investigated the efficiency of the STA in isolation and found that it attained a maximum when the average number of packets in the active set was about 1.1. This observation led him to develop the ''Dynamic'' Tree Algorithm, which tries to split the active set into subsets containing approximately 1.1 packets as quickly as possible. To achieve this goal, the channel access algorithm initiates a sequence of ''smaller'' sub-CRIs instead of one ''big'' CRI. The number of packets per sub-CRI is adjusted under dynamic control, based on the length of the previous CRI and an estimate of the current transmission rate.

The specifics of this control scheme are beyond the scope of this work and can be found in [Capet79]. However, the implementation of the splitting into more than two sub-CRIs is interesting. If the approximation of splitting into a number of sub-CRIs that is a power of two that Capetanakis proposes is accepted, then this splitting can be performed through the same mechanism that resolves conflicts. Instead of starting at the root of the tree, one

[1] A similar algorithm, has been proposed by Molloy [Mollo85] for the LAN environment, and analyzed in [Liu87].

''skips'' the initial part and starts at some lower level of the tree. Notice that, if the population is finite (assume a power of two), and the lowest level is chosen, each station gets its own slot, and the scheme is none other than TDMA. However, through the dynamic estimation of the traffic intensity, the ''Dynamic'' Tree Algorithm adapts to the current traffic conditions from the STA/GA to TDMA. Hence, the name ''Generalized TDMA'' that Capetanakis used for it [Capet79]. With a finite population, this can lead to throughputs that approach one. In this sense, it is superior to all other channel access algorithms. However, this type of traffic cannot be considered bursty.

1.3.3. Window Channel Access

A different approach to obtain the ''correct'' number of packets at the beginning of a CRI was suggested by Massey [Masse81] (and is essentially the channel access algorithm of the FCFS 0.487 Algorithm, which we describe later). This method resembles Gated Access, but when the gate opens, only packets that arrived in a specified time period — a window on the arrival-time axis — are allowed to join the new CRI. Therefore, windows are laid on the arrival-time axis sequentially with the left boundary of the current window coinciding with the right boundary of the previous window.[1] We refer to gated and window access collectively as blocked access.

There is a maximum window size specified by the algorithm, which is always selected when the backlog is large and the position of the current window is far from the actual time that transmissions take place in the CRI (see Fig. 1). The maximum window size must be at least as large as the expected CRI length to guarantee that the window will not be ''left behind.'' Therefore, the maximum window size, which we denote by Δ, is always greater than one. Thus, at light loads, where CRIs are usually one slot long, for the algorithm to be causal, smaller window sizes have to be chosen. If we define the *lag* of the algorithm as the time from the end of the window (right boundary), until the end of the conflict resolution CRI it generates, denoted by the random variable t_1 (see section 5), then the Window Algorithm (WA) specifies that the next window size should be the $\min\{ t_1, \Delta \}$.

There is, however, another alternative, also suggested by Massey [Masse81]. Instead of having a variable window size, which complicates the analysis of the protocols, a constant window size can be maintained if stations delay the start of the next CRI, if necessary, until a ''full'' window can be formed. We call these delays ''rest'' periods and the algorithm Simplified Window Algorithm (SWA). Although the SWA exhibits higher delay than the WA, especially in light load conditions, it is not difficult to see that they have the same capacity, since at data rates near capacity, ''small'' windows are rarely chosen. Thus, although ''unnatural,'' the SWA is useful, particularly since the extra delays happen only when the backlog is small.

1.3.4. Free Channel Access

Free Access (FA) algorithms are more easily described with the stack interpretation of Tree Algorithms and hence also called Stack Algorithms [Tsyba80b]. For reasons that will be clear after their description, Stack Algorithms are considered only in combination with CRAs that use random addressing. With blocked access, all stations start the CRI at the same slot, at the top of their stacks. Stations that do not have packets to transmit must update the state of their system stack after every slot, to determine the end of each CRI in order to know when to join a CRI when they finally do get a packet. The observation that led to the discovery of FAAs must have been the following. With the STA, nothing essential changes if at some point, a packet that was not part of the initial conflict, decides to assume the position of the top of the stack and transmit. If no other station transmitted in this slot, then the packet is successful and did not interfere with other packets. Moreover, this does not change the actions of the other stations since the STA does not really distinguish between idle and success. If two or more other stations transmitted, then again, the difference because of the new station transmitting is minimal, and cannot be sensed by any other station. Finally, if one other station transmitted, then the outcome of the slot has been changed by this decision of the new station to transmit. However, no station can distinguish whether this is a new station that caused the collision or one of the initial contenders. Therefore, the ''correctness'' of the algorithm can be deduced from that of the blocked access CRAs with random addressing. Performance analysis, of course, is a different question. Actually, FA increases the ''randomness'' of the whole process, and therefore, is easier to analyze.

---

[1] Note that the use of packet arrival-time information by Window Algorithms for channel access, does not necessarily imply the use of arrival-time addressing.

FA has some advantages over blocked access. Apart from the fact that the problem of performance analysis of Stack Algorithms can be considered solved, they also possess a property that is very important for implementation. Since stations can join the CRI at any time, and not necessarily at the beginning of a CRI, they do not need to be aware of the full channel history, which is needed by blocked access algorithms in order to keep track of CRI boundaries. Therefore, the network need not be ''brought-up'' simultaneously; instead, stations can drop in and out at will. This property is usually referred to as *limited sensing.* Furthermore, the deadlock condition of the MTA is not a problem any more, since, eventually, a new packet will be transmitted in one of the empty slots and force the algorithm to stop ''level skipping.''

On the other hand, the main disadvantage of FA is worse performance than other CAAs. FA does not attain as high capacities[1] as other CAAs, mean delay (for the same throughput) is higher, and worse, the variance of the delay is much higher because the global scheduling discipline is essentially LCFS instead of FCFS.

1.4. The FCFS 0.487 Algorithm

This algorithm was independently discovered by Gallager [Galla78], Tsybakov and Mikhailov [Tsyba80], and others (G. Ruget and/or J. Pellaumail [Ruget81]). In the literature it is usually referred to as Gallager's algorithm, the Gallager-Tsybakov-Mikhailov algorithm, the 0.487 Algorithm, the FCFS 0.487 Algorithm, or as the Part-and-Try algorithm. A modification of this algorithm using optimal dynamic biased splitting [Mosel85] is the most efficient multiple-access algorithm known for the infinite population Poisson model. Therefore, it serves as a lower bound for the capacity of the ALOHA-type channel.

This algorithm is based on the ''tree pruning'' rule and uses window access and arrival-time addressing to combine the ''dropped'' sub-sets with a ''fresh'' interval into a new initial set of the ''correct'' packet intensity to apply the CRA on.

The name of the algorithm, comes from its capacity that has been determined to be ≈ 0.4871 when fair splitting is considered, but rises to ≈ 0.48776 after individual optimization of the chosen sub-intervals, through dynamic programming [Mosel85]. It is also easy to see that the algorithm delivers packets in a FCFS order, since the subsets are visited in that order. This feature is particularly interesting, not only because of fairness considerations, but also because it minimizes the variance of the delay [Kingm62].

1.5. Other Limited Sensing Protocols

The Free Access Stack Algorithms are not the only ones that exhibit the limited sensing property. In this section, we describe two other classes of limited sensing protocols. They are best thought of as transformations of an existing blocked access protocol. The advantage of these protocols is that the necessary transformations for supporting limited sensing can be done without a (significant) loss of maximum attainable throughput compared to full sensing algorithms [Humbl86]. However, there is still a significant effect on delay because of an initial synchronization period and their almost-LCFS packet delivery.

A limited sensing version of the 0.487 Algorithm is presented by Humblet in [Humbl86] and independently by Georgiadis and Papantoni-Kazakos in [Georg87]. This protocol differs from the well-known FCFS 0.487 Algorithm in two important ways. First, the newly-active stations must find a way to synchronize with the current state of the CRI. (This is necessary since the protocol is still blocked access, and hence they cannot participate in the CAA until they observe the end of a CRI.) Fortunately, the 0.487 Algorithm has a maximum stack depth of two cells, so the authors recognized that either a conflict (indicating that the stack depth is now 2) or a success followed by a non-conflict (indicating that the stack depth is now 0) is sufficient for synchronization. The only problem is that a sequence of consecutive idle slots might indicate a sequence of empty CRIs (each leaving the stack at depth 0) or many consecutive applications of level skipping within a single CRI (each leaving the stack unchanged at depth 2). This ambiguity is solved by limiting the number of consecutive level skippings[2] to some fixed value, R. The choice

---

[1] Limited sensing algorithms with high capacity have been designed [Humbl86], [Georg87]. However, these are not of the Free Access type, even though somewhat similar.

[2] Note that this change also improves the robustness of the protocol by eliminating the level skipping deadlock problem.

of R represents a tradeoff between synchronization delay (which is at most R slots, with the maximum occurring under light traffic) and capacity (which rises from 0.449 at $R = 2$ to 0.487 at $R \geq 6$ — see Table 6). The other major change involves letting the window run *backwards* along the arrival time axis, starting from the most recently synchronized group, skipping any previously resolved intervals until a total length of Δ has been accumulated. Although this window manipulation gives us almost LCFS delivery of the packets, it means that each station's CAA need only be aware of the time axis *after* its own packet arrival to determine when to join a CRI.

The SWA suggests another approach to limited sensing [Tsyba85, Woodr92]. Since each station — newly active or otherwise — must be able to determine the slot boundaries, it should not be much harder for them to count slots to determine the *window boundaries*, assuming they are constant-size and integer-valued. If this information is known to all stations, then there is no need for a newly-active station to synchronize with the state of any ongoing CRI. Instead, it just waits for the next window boundary and joins the associated CRI, which is given preemptive priority over the ongoing CRI (if any). When the interruption initiated by this new CRI is completely finished (which only depends on the channel history *after* this point), execution of the previous CRI resumes. Note that although the CRIs associated with each window are initiated in FCFS order, the preemptions still give us almost LCFS delivery of the packets. The LCFS-Preemptive Window CAA may be combined with any CRA that does not terminate without transmitting all packets in the initial set to form a limited sensing protocol. Furthermore, since the operation of the CRA is unchanged (except that the slots for different CRIs may become interleaved because of the preempt/resume scheduling of CRIs), the transformation from full-sensing to limited sensing has no effect on capacity.

2. CRI Length Statistics

One of the most important performance metrics for CRAs is the CRI length, which we can broadly define as the time needed to resolve a conflict. Unfortunately, the meaning of the CRI is not identical for all algorithms. For Separable Blocked Access type protocols the definition is rather straightforward since all packets that were part of the initial conflict, and only those, are transmitted successfully in the CRI. However, with Free Access the initial contenders are not the only packets transmitted in the CRI. Instead, all packets that arrived during the CRI are also transmitted. For the FCFS 0.487 Algorithm on the other hand, the situation is reversed; only a subset of the packets participating in the initial conflict need be transmitted successfully in the CRI. Because of these differences, CRI length statistics are not directly comparable. Instead, these statistics are used to determine the capacity and delay statistics of the protocols, which can be compared directly.

We denote the length of a CRI by the random variable l, with Probability Generating Function (PGF) $Q(z)$ and mean and second moment L and H, respectively. Even though most studies of CRAs concentrated on capacity results (for which typically only the mean CRI length is required), or mean delay (for which usually the first two moments suffice), we will present distribution results (actually, PGFs), which are usually not more difficult to obtain than moments and are necessary for delay distribution results. In this section we present techniques to obtain the statistics of the conditional CRI length using two different methodologies: (i) conditioning on the multiplicity of a conflict, n, i.e., the number of packets transmitted at the beginning of the CRI, and (ii) conditioning on the Poisson intensity of the active set at the beginning of a CRI, x.

We denote the time spent by a 'tagged' packet in the CRI by the random variable t_2 (since it is the third of the components of the packet delay denoted t_0, t_1, and t_2, and fully defined and discussed in section 5). The techniques used to compute the statistics for t_2 are identical to those for the CRI length. This is not surprising since both the end of a CRI and the times for a station to transmit within the CRI are tracked by counters that perform essentially the same functions. As we have seen in section 1.2.1, one is focussed on the individual station in order to determine the station transmission instants, including the final successful transmission, thus determining t_2, and the other is tracking the full set of contenders in order to determine the end of the CRI, and thus l.

2.1. Conditioning on Collision Multiplicity

We first consider conditioning on the multiplicity of the conflict. This has been the traditional approach to obtain the mean and higher moments of the CRI length, (see, for example, [Tsyba78, Capet79b, Masse81, Rom90]). We define the PGF for the CRI length conditioned on n packets at the beginning of the CRI by

$$Q_n(z) \triangleq \sum_{i=0}^{n} \Pr\{\, l = i \mid n \text{ packets at the beginning of the CRI} \,\}\ z^i \,.$$

We present below results for various CRAs and discuss proof techniques or provide intuitive justifications for the results.

2.1.1. The Standard Tree Algorithm

We first consider the binary STA. From the specification of the algorithm we see that when there is no conflict to begin with the CRI length is equal to one; thus,

$$Q_0(z) = Q_1(z) = z \,. \tag{1}$$

When the CRI starts with a collision, i.e., for $n > 1$, two sub-CRIs are generated after the initial collision slot. If the first sub-CRI contains i packets, then the second must contain the remaining packets, i.e., $n-i$. Each of these sub-CRIs is statistically indistinguishable from a CRI with the same number of packets, therefore, we obtain the following recursion on the CRI length

$$Q_n(z) = \sum_{i=0}^{n} B_{n,i}\ z\, Q_i(z)\, Q_{n-i}(z) \,, \qquad n = 2,\, 3,\, \cdots \,. \tag{2}$$

where

$$B_{n,i} \triangleq \Pr\{\, i \text{ packets in the first subset} \mid n \text{ packets total} \,\} \,.$$

For fair splitting we have

$$B_{n,i} = \binom{n}{i} 2^{-n} \tag{3}$$

i.e., the binomial probability of obtaining a $(i,\, n-i)$ split. Note that Eq. (2) contains $Q_n(z)$ on both sides. However, we can easily solve algebraically for $Q_n(z)$, leaving only terms with $Q_i(z)$ for $i < n$ on the other side. For brevity we do not show the resulting expression here or later in other similar cases. The first few conditional PGFs obtained by this recursion, defined by Eqs. (1) and (2), are shown in Table 1.

The mean CRI length for conflicts of multiplicity n, L_n, can be obtained by differentiating the PGFs and setting $z = 1$. Alternatively we can either differentiate Eqs. (1) and (2) to obtain the following recursions on the mean, or obtain them directly with the same reasoning used above for the PGFs (see for example [Masse81])

$$L_0 = L_1 = 1 \,, \tag{4}$$

$$L_n = \sum_{i=0}^{n} B_{n,i}\, [\, 1 + L_i + L_{n-i} \,] = 1 + 2 \sum_{i=0}^{n} B_{n,i}\, L_i \,. \tag{5}$$

Evaluating L_n numerically, Massey saw that $L_n \simeq 2.886\, n$ for large n. Therefore, he conjectured that

$$\lim_{n \to \infty} L_n / n \ = \ 2 / \ln 2 \simeq 2.886 \,.$$

This conjecture proved to be false, even though the result is valid for practical purposes. Instead, it was found that this limit does not exist, since L_n/n fluctuates around $\simeq 2.88$ (with very small amplitude — see [Mathy85]). Nevertheless, Massey did find tight bounds for L_n and the second moment of the conditional CRI length that depend only on n. We'll return to this discussion later, in section 3 on the capacity of CRAs.

We now turn our attention to another metric, related to the CRI: the time a 'tagged' packet spends in the CRI, from the slot initiating the CRI to the slot the 'tagged' packet is successfully transmitted (inclusive). This is the duration of a CRI from the perspective of the 'tagged' packet (which, of course, stops participating in the CRI when it is successfully transmitted). We denote this time by the random variable t_2. Analogously to previous definitions, we define

$$G_{n+1}(z) \stackrel{\Delta}{=} \Pr\{\, t_2 = i \mid n \text{ other packets at the beginning of the CRI} \,\}\ z^i ,$$

and

$$G^{(s)}_{n+1}(z) \stackrel{\Delta}{=} \Pr\{\, t_2 = i \mid n \text{ other packets in the CRI and tagged packet chooses sub-CRI } s \,\}\ z^i ,$$

where s is 0 or 1, for the first and second (or left and right) sub-CRI, respectively. For the STA it should be clear that

$$G_1(z) = z \tag{6}$$

and for all $n \in \mathbb{N}$,

$$G^{(0)}_{n+1} = z \sum_{i=0}^{n} B_{n,i}\, G_{i+1}(z) \quad \text{and} \quad G^{(1)}_{n+1} = z \sum_{i=0}^{n} B_{n,i}\, Q_i(z)\, G_{n-i+1}(z) .$$

The last two equations simply state that if the 'tagged' packet joins the first subset, its t_2 value will be given by the statistics of t_2 with i other packets joining the first subset, plus one for the initial conflict, and averaged over all possible values of i. If the 'tagged' packet joins the second subset, then in addition it will have to wait for the sub-CRI generated by the i packets that joined the first subset to complete. Finally, for fair splitting the 'tagged' packet can be found in either subset with equal probability, thus we have

$$G_{n+1}(z) = \frac{1}{2}\, G^{(0)}_{n+1}(z) + \frac{1}{2}\, G^{(1)}_{n+1}(z) , \quad n = 2, 3, \cdots . \tag{7}$$

Tables 1 and 2 show the first few conditional PGFs and moments for l and t_2.

2.1.2. Level Skipping

It is not difficult to modify the above expressions to account for the 'level skipping' rule of the MTA. Therefore, Eq. (2.1.1:2) becomes[1]

$$Q_n(z) = B_{n,0}\, z\, Q_n(z) + \sum_{i=1}^{n} B_{n,i}\, z\, Q_i(z)\, Q_{n-i}(z) , \quad n = 2, 3, \cdots . \tag{1}$$

That is, instead of the term $z\, Q_0(z)\, Q_n(z) = z^2\, Q_n(z)$ used in the case of a $(0, n)$ split for the STA, we use the term $z\, Q_n(z)$ for the MTA, i.e., one slot less. Note that for the PGF of t_2 for the MTA, Eqs. (2.1.1:6) and (2.1.1:7) can be applied with no structural change, however, the conditional CRI lengths used, $Q_n(z)$ for $n > 1$, must be obtained from Eq. (1) rather than Eq. (2.1.1:2).

Finally, even though in the case of the STA the symmetry of the algorithm does not provide any incentive for biased splitting, this is not the case with the MTA. It is easy to consider biased splitting, with packets joining the first or second subset after a collision at random, with probability p and $\bar{p} \stackrel{\Delta}{=} 1 - p$, respectively. Actually, there is no modification to Eq. (1) that is required to model this case, except that $B_{n,i}$ should be computed through the more general form of Eq. (2.1.1:3), i.e.,

$$B_{n,i} = \binom{n}{i} p^i\, \bar{p}^{n-i} . \tag{2}$$

Conditional CRI lengths for the MTA (both unbiased and optimally biased) are shown in Table 3.

---

[1] We number equations in each section independently and therefore refer to non-local equations by both section number and equation number.

2.1.3. Tree Pruning

As we have seen, when we use 'tree pruning' only a subset of the original contenders is successfully transmitted during the CRI. In particular, when the first subset visited after a collision leads to another collision, the second subset is dropped. I.e., for n, $i > 1$, instead of the conditional CRI having length $z\,Q_i(z)\,Q_{n-i}(z)$ the 'tree pruning' rule leads to a length of only $z\,Q_i(z)$ (and the second subset being dropped, to be dealt with later on). Therefore, the conditional CRI length for the 0.487 Algorithm which uses both the 'level skipping' and the 'tree pruning' rules is given by

$$Q_n(z) = B_{n,0}\, z\, Q_n(z) + B_{n,1}\, z^2\, Q_{n-1}(z) + \sum_{i=2}^{n} B_{n,i}\, z\, Q_i(z)\,, \quad n = 2, 3, \cdots . \tag{1}$$

The 'tree pruning' rule was originally presented in combination with a version of the (Simplified) Window Algorithm for channel access and arrival-time addressing [Galla78]. It is still not easy to separate these components. Therefore, we continue the discussion of 'tree pruning' in section 2.2.3, after we discuss channel access algorithms. There is, however, a way to use the 'tree pruning' rule and transmit all packets in the initially enabled set by using the 3CA. We discuss this next.

2.1.4. The Two-Cell and Three-Cell Algorithms

The 2CA and 3CA have limited stack depth, exactly like the FCFS 0.487 Algorithm. We define the following two families of conditional PGFs. First, we denote by $Q_n(z)$ the PGF of the CRI length of a ''fresh'' interval containing exactly n packets. Second, we denote by $Q_{n,m}(z)$ the PGF of a sub-CRI with a primary interval containing exactly n packets, but also associated with a secondary (right) interval including exactly m packets ($n, m = 0, 1, 2, \cdots$). The sub-CRI is assumed complete after both the primary and secondary intervals are resolved completely.

From the definition of the algorithm we obtain the following recursions for the PGF of the CRI length:

$$Q_0(z) = Q_1(z) = z \tag{1}$$

$$Q_n(z) = B_{n,0}\, z\, Q_n(z) + B_{n,1}\, z^2\, Q_{n-1}(z) + \sum_{j=2}^{n} B_{n,j}\, z\, Q_{j,n-j}(z)\,, \quad n = 2,3, \cdots \tag{2}$$

$$Q_{n,m-n}(z) = z\, Q_{m-n}(z)\,, \quad n \le 1 \tag{3}$$

$$Q_{n,m-n}(z) = B_{n,0}\, z\, Q_{n,m-n}(z) + B_{n,1}\, z^2\, Q_{n-1,m-n}(z) + \sum_{j=2}^{n} B_{n,j}\, z\, Q_{j,m-j}(z)\,, \quad n > 1\,. \tag{4}$$

Note the similarity of Eqs. (2) and (4) with Eq. (2.1.3:1) for the combined 'level skipping' and 'tree pruning' modifications. The only difference is that the 3CA keeps the second subset around and eventually resolves it, while with 'tree pruning' this subset is dropped. Some results for the mean conditional CRI length are given in Table 3. Note that the mean conditional CRI length for the 3CA is shorter than that for the MTA with biased splitting, but only for CRIs containing 6 packets or less.

Similarly, we denote by $G_n(z)$ and $G_{n,m}(z)$, the conditional PGFs for t_2. Then, we can obtain the following recursions:

$$G_1(z) = z$$

$$G_n(z) = z\, G_n(z)\, B_{n,0} + \left[\frac{1}{n} z^2 + \frac{n-1}{n} z^2\, G_{n-1}(z) \right] B_{n,1} + z \sum_{j=2}^{n} G_{j,n-j}(z)\, B_{n,j}\,, \quad n>1$$

$$G_{1,m-1}(z) = \frac{1}{m} z + \frac{m-1}{m} z\, G_{m-1}(z)\,,$$

$$G_{n,m-n}(z) = z\, G_{n,m-n}(z)\, B_{n,0} + \left[\frac{1}{m} z^2 + \frac{m-1}{m} z^2\, G_{n-1,m-n}(z) \right] B_{n,1} + z \sum_{j=2}^{n} G_{j,m-j}(z)\, B_{n,j}\,, \qquad n > 1$$

The results for the 2CA are very similar. For example, Eqs. (2) and (4) are modified as follows in order to eliminate the 'level skipping' modification that the 3CA algorithm uses, but not the 2CA.

$$Q_n(z) = B_{n,0}\, z^2\, Q_n(z) + B_{n,1}\, z^2\, Q_{n-1}(z) + \sum_{j=2}^{n} B_{n,j}\, z\, Q_{j,n-j}(z)\,, \qquad n = 2,3,\cdots \tag{5}$$

$$Q_{n,m-n}(z) = B_{n,0}\, z^2\, Q_{n,m-n}(z) + B_{n,1}\, z^2\, Q_{n-1,m-n}(z) + \sum_{j=2}^{n} B_{n,j}\, z\, Q_{j,m-j}(z)\,, \qquad n > 1\,. \tag{6}$$

2.2. Conditioning on Traffic Intensity

We proceed now to obtain the distribution of CRI lengths for CRIs with a number of packets that is distributed according to a Poisson distribution of known mean. The PGF of CRI lengths for CRIs with a Poisson packet intensity x, denoted by $Q(x,z)$, is defined as

$$Q(x,z) \stackrel{\Delta}{=} \sum_{n=0}^{\infty} z^n \Pr\{\, l = n \mid x \,\}\,.$$

Functional Equations (FEs) on $Q(x,z)$ and other CRI length statistics for various CRAs have been obtained and solved in many previous papers (for example, see [Mathy85, Fayol85, Huang85, Tsyba85, Polyz87b], and others).

2.2.1. The Standard Tree Algorithm

For the STA, $Q(x,z)$ can be obtained by solving the following FE

$$Q(x,z) = z\, Q^2(x/2,z) + (z - z^3)\, F(x)\,. \tag{1}$$

It is easy to intuitively justify the truth of this FE. The CRI length in the case of a conflict is the sum of the duration of the initial (conflict) slot and the lengths of the two sub-CRIs that are generated by the splitting. Since the packets join one of the two subtrees at random, the sub-CRIs are statistically indistinguishable between themselves and indistinguishable from a CRI with, on the average, half the number of initial contenders, i.e., their PGF is $Q(x/2,z)$. The sum (convolution, actually) in the time domain becomes a product in the transform domain and corresponds to the first term of Eq. (1). In the case of no-conflict, however, the CRI reduces to one slot, in which case the first term overestimates the length by two slots. Therefore, the second term must be introduced as an adjustment. I.e., from the true transform of the CRI length in this case, z, we must subtract the result implied by the first term of the equation, i.e., z^3, weighted by the probability of no-conflict, $F(x)$.

We can extend this result to the d-ary splitting case ($d \geq 2$) as follows

$$Q(x,z) = z\, Q^d(x/d,z) + (z - z^{d+1})\, F(x)\,. \tag{2}$$

and get a FE for t_2:

$$G(x,z) = z\, G(x/d,z)\, \frac{1}{d} \sum_{k=0}^{d-1} Q^k(x/d,z) + \left[1 - z\, \frac{1}{d} \sum_{k=0}^{d-1} z^k \right] z\, \Phi(x)\,. \tag{3}$$

Proofs of Eqs. (1), (2), and (3) are given in [Polyz89].

2.2.2. Level Skipping — the Modified Tree Algorithm

We can easily extend these results to the MTA with biased splitting; for example we have

$$Q(x,z) = z\, Q(px,z)\, Q(\bar{p}x,z) + (z - z^3)\, F(x) + (z - z^2)\, [\, Q(\bar{p}x,z) - z\, F(\bar{p}x)\,]\, \Phi(px)\,. \tag{1}$$

The new term is due to the 'level skipping,' and can be intuitively justified as follows. In the case that after a conflict the first subset is empty (this happens with probability $\Phi(px)\, F(\bar{p}x)$), with the MTA we spend two, instead of three, slots before we split the second subset. However, in this case we must take into consideration that it is

known that the second sub-CRI cannot contain less than two packets. For a proof of this FE see [Polyz89]. In Fig. 3 we compare the efficiency of the STA, the unbiased MTA, and the optimally biased MTA (MTA*) for binary and ternary splitting. Notice that the curves for the binary and ternary STA cross, indicating that ternary splitting is more efficient when the expected number of packets in a CRI is large. We discuss this further in section 3.

2.2.3. Tree Pruning — the FCFS 0.487 Algorithm

We now turn our attention to the celebrated FCFS 0.487 Algorithm. We denote by $Q(w,z)$ the PGF of CRI lengths,[1] and the conditional PGFs for CRIs starting with a conflict and non-empty CRIs by $Q^{(c)}(w,z)$ and $Q^{(\bar{i})}(w,z)$, respectively. Considering the two subsets in a possible splitting, the set of all possible events is $\{i,\ s,\ c\} \times \{i,\ s,\ c\}$. By inspection we obtain the following recursion

$$\begin{aligned} Q(w,z) &= \Pr\{(i,i) \text{ or } (i,s) \text{ or } (s,i)\}\ z + \Pr\{(i,c)\}\ z\ Q^{(c)}(\bar{p}w,z) \\ &\quad + \Pr\{(s,s) \text{ or } (s,c)\}\ z^2\ Q^{(\bar{i})}(\bar{p}w,z) + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\}\ z\ Q^{(c)}(pw,z) \\ &= F(x)\ z + \Phi(px)\ \bar{F}(\bar{p}x)\ z\ Q^{(c)}(\bar{p}w,z) + \Psi(px)\ \bar{\Phi}(\bar{p}x)\ z^2\ Q^{(\bar{i})}(\bar{p}w,z) + \bar{F}(px)\ z\ Q^{(c)}(pw,z)\ . \end{aligned}$$

We can obtain A FE on $Q(w,z)$ by using the identities

$$\bar{F}(x)\ Q^{(c)}(w,z) = Q(w,z) - z\,F(x) \qquad \text{and} \qquad \bar{\Phi}(x)\ Q^{(\bar{i})}(w,z) = Q(w,z) - z\,\Phi(x)\ .$$

After substitutions, we obtain the following FE

$$Q(w,z) = z\,Q(pw,z) + [\,z\,\Phi(px) + z^2\,\Psi(px)\,]\,Q(\bar{p}w,z) + (\,z - z^2\,)\,F(x) + (\,z - z^3\,)\,\Psi(px)\,\Phi(\bar{p}x) - z^2\,F(px)\,. \quad (1)$$

Because tree pruning might leave part of the window unexamined, another useful metric is the augmented CRI length, $\tilde{l}$, i.e., the length of a CRI generated by a window of (initial) size w, augmented by the length of the part of the window that remains unexamined at the end of the CRI, if any. Notice that the augmented CRI length is not necessarily an integer anymore, and thus, a Laplace transform is the appropriate transform:

$$\tilde{Q}^*(w,s) \stackrel{\Delta}{=} \int_{-\infty}^{+\infty} e^{-st}\ \frac{d}{dt}\ \Pr\{\,\tilde{l} \le t\,\}\ dt\ .$$

However, it is convenient to define an intermediate form, $\tilde{Q}(w,z)$, very similar to a PGF, so that

$$\tilde{Q}^*(w,s) = \tilde{Q}(w,e^{-s})\ .$$

By inspection we obtain the following FE

$$\begin{aligned} \tilde{Q}(w,z) &= \Pr\{(i,i) \text{ or } (i,s) \text{ or } (s,i)\}\ z + \Pr\{(i,c)\}\ z\ \tilde{Q}^{(c)}(\bar{p}w,z) \\ &\quad + \Pr\{(s,s) \text{ or } (s,c)\}\ z^2\ \tilde{Q}^{(\bar{i})}(\bar{p}w,z) + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\}\ z\ \tilde{Q}^{(c)}(pw,z)\ z^{\bar{p}w} \\ &= F(x)\ z + \Phi(px)\ \bar{F}(\bar{p}x)\ z\ \tilde{Q}^{(c)}(\bar{p}w,z) \\ &\quad + \Psi(px)\ \bar{\Phi}(\bar{p}x)\ z^2\ \tilde{Q}^{(\bar{i})}(\bar{p}w,z) + \bar{F}(px)\ \tilde{Q}^{(c)}(pw,z)\ z^{1+\bar{p}w}\ . \\ &= z^{1+\bar{p}w}\ \tilde{Q}(pw,z) + [\,z\,\Phi(px) + z^2\,\Psi(px)\,]\,\tilde{Q}(\bar{p}w,z) \\ &\quad + (\,z - z^2\,)\,F(x) + (\,z^2 - z^3\,)\,\Psi(px)\,\Phi(\bar{p}x) - z^{2+\bar{p}w}\,F(px)\ . \end{aligned} \quad (2)$$

---

[1] Because we will need w as a parameter separate from λ, we use w instead of x as the first argument of $Q(.,z)$, and the other transforms used in this section, with λ implied.

Another related metric useful for the evaluation of the capacity and delay of the FCFS 0.487 Algorithm is the mean number of successful transmissions per CRI. We denote the PGF and the mean number of packets transmitted during a CRI by $M(x,z)$ and $N(x)$, respectively (assuming an initial interval with Poisson packet intensity x).[1] We have

$$M(x,z) = \Pr\{(i,i)\}\, z^0 + \Pr\{(i,s) \text{ or } (s,i)\}\, z + \Pr\{(s,s)\}\, z^2 + \Pr\{(i,c)\}\, M^{(c)}(qx,z)$$

$$+ \Pr\{(s,c)\}\, z\, M^{(c)}(qx,z) + \Pr\{(c,i) \text{ or } (c,s) \text{ or } (c,c)\}\, M^{(c)}(px,z)$$

which after some manipulations takes the form

$$M(x,z) = M(px,z) + (1+px\,z)\,[\,M(\bar{p}x,z) - 1\,] .$$

Differentiating, we get the FE for the mean

$$N(x) = N(px) + (1+px)\,e^{-px}\,N(\bar{p}x) .$$

With similar techniques we can also obtain a FE for the product $N(x)\,J^*(w,s)$, the PGF of the number of packets transmitted in the CRI multiplied by the Laplace Transform of $t_0 + t_2$. This is of interest because the FCFS 0.487 Algorithm uses arrival-time addressing, which introduces a dependence between t_2 and t_0 (and thus, they need to be treated jointly). For convenience, we use the z variable as the transform of one unit of time, instead of e^{-s}. Since, we concentrate on a ''tagged'' packet, there is always at least one transmission in the sets considered here.

$$N(x)\,J^*(w,s) = \Psi(x)\,(z - z^2)\,U^*(w,s)$$

$$+ [\,\Phi(px)\,z + \Psi(px)\,z^2\,]\,N(\bar{p}x)\,J^*(\bar{p}w,s) + z^{1+\bar{p}w}\,N(px)\,J^*(pw,s) \ .$$

2.3. Free Access

One of the first exact analyses of the performance of a CRA (including packet delay) is due to Fayolle, *et al.* [Fayol85]. The protocol they considered is a Stack Algorithm, consisting of the binary STA with biased splitting combined with free channel access.[2] The main feature of their analysis was to determine FEs for the mean and PGF of the CRI length. The arrival process is assumed Poisson with λ new packets arriving in each slot. Below, we illustrate their approach by providing a simplified derivation of these equations.

Let the random variable $l(x)$ represent the length of an ordinary CRI starting from a Poisson traffic intensity[3] of x. Observe that if a collision occurs in the first slot, then the initial set will be split into two components using a biased ''coin toss'' with probabilities p and $\bar{p}$, and — because we have free access — each component will be joined by a new group of packets chosen independently from a Poisson process with parameter λ. Now consider an auxiliary CRI in which we unconditionally ''skip'' the root of the conflict resolution tree and begin the traversal from the pair of second level nodes. Obviously, if there is a conflict at the root, then the two CRIs are identical except for the missing root in the auxiliary CRI. However, if there is an idle slot at the root node, then the ordinary system is finished but the auxiliary system continues with two (independent) ordinary CRIs each with Poisson intensity λ. Similarly, if there is a success at the root node, then the ordinary system is finished but the auxiliary system continues with two (independent) CRIs such that both of them include an independent Poisson sample with parameter λ and one of them includes a single ''extra'' packet. Defining the new random variable with PGF $Q^{(+1)}(x,z)$ to represent the length of a CRI including the ''extra'' packet, we obtain the following FE:

---

[1] Recall that the algorithm, after a collision, ''drops'' the right sub-interval, and therefore not all packets of the initial interval get transmitted successfully.

[2] The first analysis of CRAs with free channel access is due to Vvedenskaya and Tsybakov [Vvede84].

[3] Since free access is assumed, the expected number of packets transmitted successfully by the end of the CRI is at least x, but possibly more. Remember that in this case the CRI is defined as the time required to return to an empty stack.

$$Q(x,z)/z = Q(px+\lambda,z)\,Q(\bar{p}x+\lambda,z) + \Phi(x)\left(1-Q^2(\lambda,z)\right) + F(x)\left(1-Q(\lambda,z)\,Q^{(+1)}(\lambda,z)\right). \qquad (1)$$

Recognizing that

$$Q'(x,z) \triangleq \frac{\partial}{\partial x}\,Q(x,z) = \frac{\partial}{\partial x}\,\sum_{n=0}^{\infty} Q_n(z)\,\frac{x^n}{n!}e^{-x} = \sum_{n=0}^{\infty} Q_n(z)\left[\frac{nx^{n-1}}{n!}e^{-x} - \frac{x^n}{n!}e^{-x}\right] = Q^{(+1)}(x) - Q(x,z),$$

we obtain the following FE for the CRI length PGF:

$$Q(x,z)/z = Q(px+\lambda,z)\,Q(\bar{p}x+\lambda,z) + \Phi(x)\left(1-Q^2(\lambda,z)\right) + F(x)\left(1-Q(\lambda,z)\,[\,Q'(\lambda,z)+Q(\lambda,z)\,]\right). \qquad (2)$$

By differentiating Eq. (2), we obtain a FE for the mean CRI length:

$$L(x)-1 \;=\; L(px+\lambda)+L(\bar{p}x+\lambda)-F(x)\left[2\,L(\lambda)+\frac{x}{1+x}\,L'(\lambda)\right].$$

Note that Eq. (2) is similar to Eq. (2.2.1:2). This is not unexpected since the basic equation for the STA (without allowing free access) coincides with Eq. (2) with $\lambda = 0$ (but $x \neq 0$, of course).

2.4. Solution Techniques for Functional Equations

Let us now address the issue of solving FEs of the types obtained in sections 2.2 and 2.3. One approach is to represent the functions as power series in x or z, substitute them into the FEs, and then to solve for each coefficient in turn by equating the coefficients of the same powers of x (or z) on both sides of the equation. For example, in the case of the binary STA we assume a power series representation for $Q(x,z)$ of the following form

$$Q(x,z) \triangleq \sum_{n=0}^{\infty} q_n(x)\,z^n$$

and solving Eq. (2.2.1:1) we obtain:

$$q_{2n}(x) = 0 \quad , \quad n = 0,\,1,\,\cdots\,, \qquad (1)$$

$$q_1(x) = (1+x)\,e^{-x}\,, \quad q_3(x) = \frac{x^2}{4}\,e^{-x}\,, \text{ and } \quad q_{2n+1}(x) = \sum_{m=0}^{2n} q_m(x/2)\,q_{2n-m}(x/2) \quad , \quad n = 2,\,3,\,\cdots\,.$$

Similarly, solving Eq. (2.2.1:3) we obtain

$$G(x,z) = \sum_{n=0}^{\infty} g_n(x)\,z^n$$

with

$$g_0(x) = 0\,,\; g_1(x) = e^{-x}\,,\; g_2(x) = \frac{1}{2}\,[\,e^{-x/2} - e^{-x}\,]\,,\; g_3(x) = \frac{1}{4}\,[\,e^{-x/4} - e^{-x/2} + x\,e^{-x}\,]\,,$$

and

$$g_{n+1}(x) = \frac{1}{2}\left[g_n(x/2) + \sum_{m=0}^{n} q_{n-m}(x/2)\,g_m(x/2)\right],\; n \geq 3\,.$$

For moments, the power series solution[1] takes a simpler form. For example, the mean CRI length for the blocked access STA is given by:

$$L(x) = \sum_{n=0}^{\infty} \alpha_n\,x^n\,, \quad \alpha_0 = 1\,, \quad \alpha_1 = 0\,, \quad \alpha_n = (-1)^n\,\frac{2\,(n-1)}{n!\,(1-2^{1-n})}\,, \quad n = 2,\,3,\,\cdots\,.$$

---

[1] The coefficient of x, namely α_1, is not determined by the power series method. However, it is easy to see that $\alpha_1 = 0$, either by recognizing that $\lim_{x\to 0} L(x) = 1$ or considering the equation for the second moment.

Recursion can also be used to solve many of the FEs we have seen. For efficiency reasons, recursion is best suited for FEs arising from the moments since these are typically single variable equations. If we have a FE in which $F(x_1)$, say, is defined in terms of $F(x_2)$ for some $x_2<x_1$, together with a non-recursive solution we can use when some threshold $x_i<x^*$ is reached, then we can evaluate $F(\cdot)$ either ''top down'' using recursion or by building a table of values ''bottom up''. For example, we can use the fact that $\lim_{x\to 0} L(x) = 1+O(x^2)$ in the blocked access STA to avoid the power series altogether. Alternately, we can combine recursion and the power series method to improve numerical stability when evaluating $L(x)$ for $x\gg 1$, which arises in the delay analysis for gated access systems [Molle92].

Many FEs can also be solved numerically using a fixed-point iteration formula. Starting with an initial estimate of the solution, we generate a sequence of new estimates by applying the FE until some metric of the difference between two successive estimates is below a given threshold. Fixed-point iteration can even handle FEs with a non-recursive structure, such as the mean CRI length for free access. Here $L(x)$ is a function of $L(px+\lambda)$ and $L(\bar{p}x+\lambda)$, so that after many levels of recursion the arguments in the FE approach 2λ (where no non-recursive solution is known) rather than zero. Although fixed-point iteration schemes are not always guaranteed to converge, the ones we have attempted have always converged (and to the same solution as obtained using other techniques). For example, the mean CRI length for the unbiased STA under free access can be solved as a countable system of linear equations of the form:

$$L_k \;=\; 1+2\,L_{k+3}-(1+x_k)\,e^{-x_k}\left[2\,L_1+x_k/(1+x_k)\,L'_1\right],$$

where $L_k \triangleq L(x_k)$, $x_0=\lambda-\varepsilon$, $x_1=\lambda$, $x_2=\lambda+\varepsilon$, $x_k=x_{k-3}/2+\lambda$, and we let $L'_1=(L_2-L_0)/(2\,\varepsilon)$. Fayolle *et al.* [Fayol85] discuss more elaborate solution techniques for the FEs that arise from free access stack algorithms.

2.5. On the Unconditional CRI Length

To obtain unconditional results we need to know the statistics of the number of packets at the beginning of the CRI. Determining these statistics typically requires studying the dynamics of the system, which we address in section 5. An interesting exception is the case of the SWA, where because of the constant size windows the number of packets at the beginning of any CRI are i.i.d. random variables, distributed according to a Poisson distribution with mean $x=\lambda w$, i.e., $P_n = e^{-x}\,x^n/n!$. Then it is easy to obtain the unconditional PGF for the CRI length from the PGF conditioned on the multiplicity of the conflict as

$$Q(x,z) = \sum_{n=0}^{\infty} P_n\,Q_n(x)\,.$$

Note that the PGF results of section 2.2 that are based on conditioning on the traffic intensity are already in final form since for the SWA the traffic intensity is constant across CRIs.

3. Capacity of Protocols

We define the *capacity* of random-access protocols, λ^*, as the *supremum* over all achievable input rates (throughputs), λ, such that the expected delay remains finite. Note that capacity is not the same as maximum throughput, because of the finite delay requirement. For example, consider that for any $M>0$ and any $\lambda<1$, M stations using round-robin TDMA can achieve a throughput of λ. Thus, in the infinite population limit as $M\to\infty$, TDMA has a maximum throughput of 1.0 but its capacity is 0.0 because the expected delay is infinite for all $\lambda>0$. The infinite population capacity is also an interesting statistic for finite population systems, as an indication of the point where the number of stations becomes a significant factor in the delay.

Capacity can also be found from a stability analysis of the underlying stochastic process describing the evolution of the protocol over time. In this approach, an embedded Markov chain is defined at some convenient points (e.g., CRI boundaries) such that stability of the Markov chain implies that the expected packet delays are bounded. For example, let τ_i be the time of the i-th embedding point, and let $N(\tau_i)$ represent the number of packets in the system at time τ_i. We define the drift

$$N_k \;=\; \mathrm{E}[\, N(\tau_{i+1}) - N(\tau_i) \mid N(\tau_i) \,=\, k \,] \tag{1}$$

to be the difference between the expected number of new arrivals joining the system and the expected number of departing packets between two consecutive embedding points starting from k in the system. In this case, we can apply Pakes' lemma to the drift to prove stability [Berts92, pp. 264-265]. In particular, if we can prove that $N_k < \infty$ for all k, and also that $N_k \le \delta < 0$ for all $k \ge k_0$, then the system is stable.

The capacity is the most important single number characterization of protocol performance. Some capacity results for various combinations of CRAs and Channel Access Algorithms (CAAs) are shown in Fig. 4. Upper bounds on the capacity of the ALOHA-type channel are also shown in the same figure (and discussed in section 4).

3.1. Gated Access, the ''Dynamic'' Tree Algorithm, and Free Access

The standard technique to determine the capacity and prove the stability of random access protocols is based on Pakes' lemma which provides sufficient conditions for the ergodicity of a Markov chain [Pakes69]. In order to illustrate the use of this technique we consider here the binary STA with Gated Access (GA), random addressing, and an infinite population generating Poisson arrivals. We observe that the length of the $(i+1)$-st CRI, l_{i+1}, depends only on the number of packets at the beginning of the CRI, n, which in turn depends only of the length of the i-th CRI. Therefore, $\{\, l_{i+1},\ i = 1, 2, \cdots \}$ is a Markov chain. According to Pakes' lemma this (irreducible, aperiodic, homogeneous Markov chain with state space the non-negative integers) is ergodic if

i. $\quad \mid \mathrm{E}[\, l_{i+1} - l_i \mid l_i = j \,] \mid \;< \infty \quad \forall j$, and

ii. $\quad \limsup_{j \to \infty} \mathrm{E}[\, l_{i+1} - l_i \mid l_i = j \,] \;< 0.$

To proceed with the proof we need to evaluate the following expression

$$\mathrm{E}[\, l_{i+1} \mid l_i = j \,] \;=\; \sum_{n=0}^{\infty} \mathrm{E}[\, l_{i+1} \mid l_i = j \text{ and number of arrivals in the } i\text{-th CRI} = n \,]\; P_n(\lambda j)$$

$$= \sum_{n=0}^{\infty} \mathrm{E}[\, l_{i+1} \mid \text{number of arrivals in} i\text{-th CRI} = n \,]\; (\lambda j) e^{-\lambda j} / n\,! = \sum_{n=0}^{\infty} L_n\, (\lambda j) e^{-\lambda j} / n\,!$$

where L_n is the mean CRI length conditioned on a conflict of multiplicity n and $P_n(x)$ is the probability of n arrivals in the i-th CRI generated from a Poisson stream with parameter $x = \lambda j$.

We have seen in section 2.1.1 that $L_n \approx \alpha\, n$ for $\alpha \approx 2.886$. Indeed, it can be shown [Masse81, Rom90] that

$$L_n \le \alpha\, n + 1 \qquad \forall n \ge 0$$

where it is numerically determined that $\alpha = 2.886$. Then we can see that

$$\mathrm{E}[\, l_{i+1} \mid l_i = j \,] \le \sum_{n=0}^{\infty} (\alpha n + 1)\, (\lambda j) e^{-\lambda j} / n\,! = \alpha \lambda j + 1$$

and thus

$$\mathrm{E}[\, l_{i+1} - l_i \mid l_i = j \,] \;\le (\alpha\lambda - 1)\, j + 1$$

from where we see that both conditions of the lemma hold for $\lambda < 1/\alpha \approx 0.346$. Thus, the capacity of the binary STA/GA is 0.346 and for any $\lambda < 0.346$ this protocol is stable. Note that this figure is below the capacity of (stabilized) slotted ALOHA ($= e^{-1} \approx 0.367$). Therefore, the attractiveness of the STA/GA protocol was limited to its stable behavior (which is achieved without external control). However, Capetanakis also developed the ''Dynamic'' Tree algorithm which extends the stability region to ≈ 0.430. Capacity results for various CRAs with GA can be found in Table 4 (which mainly presents results for window access) for $w = \infty$.

Mathys and Flajolet [Mathy85] have performed in-depth stability analysis for all combinations of the STA and MTA (with d-ary splitting) with gated and free channel access. It is interesting to note that they have determined that $\lim_{n\to\infty} L_n/n$ does not exist for gated channel access (with fair splitting). Instead , this quantity exhibits small oscillations (and includes a component that is a periodic function in $\log_d n$, where d is the degree of splitting). We present some of their results for Free Access in Table 5.

3.2. Window Access

For CRAs that satisfy our separability condition, average throughput and input rate, λ, coincide, as long as the system is stable. To find the maximum stable throughput of the protocol, we can apply Pakes' lemma using Eq. (3:1). Recognizing that whenever the lag is at least Δ, both the WA and the SWA select a window size of Δ, we obtain

$$\lambda L(\Delta\lambda) - \lambda\Delta < 0 \tag{1}$$

which is equivalent to both

$$L(\Delta\lambda) < \Delta \tag{2}$$

(i.e., the expected change in the lag must be negative), and

$$\lambda < \frac{x}{L(x)} \tag{3}$$

(i.e., the input rate should be less than the efficiency of the protocol). This stability condition can also be obtained from a queueing model for the system, from which the delay is obtained in section 5. An investigation of the stability region for different values of the window size reveals the results shown in Table 4. It is apparent from this table that as the window size increases the regulating ability of the channel access algorithm is gradually lost, and at the limit, Window Access has the same capacity with Gated Access. Actually, for $\Delta = \infty$ (non-simplified) Window Access is identical to Gated Access.

Note that the capacity is rather insensitive to the window size around the optimal. Furthermore, if we concentrate on integer window sizes, the most interesting values of w are 2, 3 and 4. We can also remark that d-ary splitting with $d > 2$ is not more efficient than binary splitting for optimal, or small, near optimal window sizes. Apparently, strict regulation of the size of the input rate (using an optimal window size) is more effective than aggressive splitting *after* the first conflict has occurred. This observation was first made by Capetanakis, and led to the ''Dynamic'' Tree Algorithm.

As the window size increases with no bound, the problem of the non existence of $\lim_{n\to\infty} L_n/n$ is manifested in the form of oscillations in the capacity of the algorithms. The oscillations are of the order reported in [Mathy85]. This is the reason that fewer than six digits are provided in some of the entries in Table 4 for $w = \infty$. Also, we can see that for large windows (that make the system essentially equivalent to one with gated access), ternary splitting is more efficient than binary in the case of the STA.

The bias for the biased MTA is chosen in a way that maximizes the capacity of the algorithm at the optimal window size. For binary splitting, $\pi = 0.418$ is optimal for the the optimal window size, i.e., $w = 2.716$, but the optimal π is very flat over the whole domain. Verification of this is provided by the fact that for $w\to\infty$ (gated access, actually), the optimal bias, as given in [Mathy85], is $\pi = .418$ (to three decimal places). For ternary splitting, the optimal $\bar{\pi}$ varies a little bit more with the window size, decreasing from 0.384 for $w = 2$, to 0.373 as $w\to\infty$.

3.3. The 0.487 Algorithm

Because of the tree pruning rule, the standard stability condition for window access must be modified for the FCFS 0.487 Algorithm. Using Eq. (3.2:3) we obtain

$$\lambda < \frac{N(x)}{L(x)} \tag{1}$$

i.e., throughput is bounded by the ratio of the mean number of successful transmissions per CRI, to the mean CRI length.[1] Maximizing this ratio with respect to x for $p = \bar{p} = 0.5$, i.e., the original algorithm with symmetrical splitting, we obtain as capacity $\lambda^* \approx 0.4871$ at $x^* \approx 1.266$. Note that $x^* = \lambda^* w$, and thus the optimal maximum window size is ≈ 2.6. However, we can also optimize over p. In this case, $\lambda^* \approx 0.48757$ at $x^* \approx 1.271$ and $p^* \approx 0.475$.

[1] We have computed these metrics in section 2.2.3.

An alternative technique to obtain the capacity of the FCFS 0.487 Algorithm would be to use Eq. (3.2:1) above, but with the mean *augmented* CRI length instead of the mean (regular) CRI length.

In [Mosel85] the splitting bias is optimized at every step, by dynamic programming, and the capacity is improved to $\lambda^* \approx 0.48776$. This last variation of the FCFS 0.487 algorithm, with capacity rounded to three decimal places of 0.488 is the most efficient multiple-access protocol. Tsybakov [Tsyba85b] discusses various other studies and optimizations related to improving the capacity of the FCFS 0.487 Algorithm. In particular, Vvedenskaya and Pinsker [Vvede83] prove that the algorithm is not optimal.

Stability results for the Limited Sensing 0.487 Algorithm have been obtained using Pakes' lemma in [Humbl86, Georg87] and are shown in Table 6.

4. Algorithm Independent Bounds on Capacity

We have now seen how a series of algorithmic refinements have led to corresponding increases in capacity. The question that naturally arises then is: are there any inherent limits to how much further we can push the capacity envelope by adding even more algorithmic refinements? Pippenger [Pippe81], using entropy arguments, was the first to show that such a limit does indeed exist, and that under the infinite population Poisson model no protocol can have a capacity greater than ≈ 0.744. From this point on, work on sharpening Pippenger's capacity bound has continued, in parallel, with the development of implementable multiple access algorithms—each of which represents a constructive lower bound to the capacity of the multiple-access channel.

In the literature, several unrelated methods have been used in constructing upper bounds to capacity. Below, we outline the main ideas from three well-known upper bounds, but, to simplify the arguments, we follow an unifying treatment introduced by Reuppel [Reupp83]. The key to his approach is the following result:

Leaf-Average Node-Sum Interchange Theorem:

Let T be a rooted tree with probabilities. That is, every node $n \in T$ has associated with it a probability, $p(n)$, such that: (i) $p(n)$ may be arbitrarily assigned if $n \in L$, the set of leaf (i.e., terminal) nodes, subject to the constraints that $p(n) \geq 0$ and $\sum_{n \in L} p(n) = 1$; and (ii) $p(n) = \sum_{c \in child(n)} p(c)$ for $n \in T-L$, the set of non-terminal nodes. (Notice that these conditions imply that $p(r) \stackrel{\Delta}{=} 1$ for r the root of T.) Also, let $f(n)$ be an arbitrary function that assigns a real value to each node n, and for $n \in T-L$ let

$$\Delta f(n) = \frac{1}{p(n)} \sum_{c \in child(n)} \big(f(c) - f(n)\big) \cdot p(c)$$

be the average change in $f(\cdot)$ between node n and its children. Then

$$f(r) + \sum_{n \in T-L} p(n) \cdot \Delta f(n) = \sum_{l \in L} p(l) \cdot f(l). \tag{1}$$

Proof: Substituting the definition of $\Delta f(n)$ into the left-hand side of Eq. (1) and expanding, we obtain

$$f(r) + \sum_{\substack{n \in T-L \\ c \in child(n)}} \sum p(c) \cdot f(c) - \sum_{\substack{n \in T-L \\ c \in child(n)}} \sum p(n) \cdot f(c).$$

Applying rule (ii) to the right-most term, we obtain $\sum_{n \in T-L} p(n) \cdot f(n)$. Recognizing that all nodes in T except r have exactly one parent, the middle term can be rewritten as

$$\sum_{\substack{n \in T-L \\ c \in child(n)}} \sum p(c) \cdot f(c) \stackrel{\Delta}{=} \sum_{n \in T-\{r\}} p(n) \cdot f(n).$$

Thus, since $f(r) \stackrel{\Delta}{=} 1$, the left-hand side of Eq. (1) reduces to

$$\sum_{n \in T} p(n) \cdot f(n) - \sum_{n \in T-L} p(n) \cdot f(n) \stackrel{\Delta}{=} \sum_{n \in L} p(n) \cdot f(n),$$

which is identical to the right-hand side.

To illustrate the use of the theorem, let us now prove the *path length lemma* [Galla78b], which states that the average length of a path from the root to a leaf is the sum of the probabilities of passing through each node. In this case, we let $f(n)$ be the number edges in the path from r to n. Clearly $\Delta f(n) \stackrel{\Delta}{=} 1$, so that the left-hand side of Eq. (1) reduces to $\sum_{n \in T-L} p(n)$. Since the right-hand side is (obviously) the average path length, the result is proven.

Now consider the decision tree, T, obtained by merging all possible sample execution traces when an arbitrary conflict resolution algorithm, A, is applied to an interval $I = [0, \nu]$ that contains Poisson packet arrivals at unit intensity. Assume that at node n in the decision tree, a subset $I(n) \subseteq I$ remains *unresolved* (i.e., no points in $I(n)$ were selected at any previous step that resulted in an idle slot or a successful transmission, and hence all packets that were initially included in $I(n)$ must still be waiting for transmission). Notice that $I(\cdot)$ satisfies: (i) $I(r) \stackrel{\Delta}{=} I$; (ii) $I(l) \stackrel{\Delta}{=} \varnothing$ for every leaf l; and (iii) $I(c) \subseteq I(n)$ for $c \in \text{child}(n)$. Also assume that $\sigma(n)$ packets have been transmitted successfully along the sample path leading to node n, and that the continuation of the algorithm at node n consists of enabling all the packets in $E(n) \subseteq I(n)$ to be transmitted.

To apply the above theorem to T, we first define $p(n)$, $n \in T$, to be the proportion of sample paths that include node n. Then we define $f(\cdot)$ to be some bound-dependent *objective function* that measures how much ''progress'' A has made in resolving I up to the given node. The bounding argument itself follows easily from the proof of the path length lemma. However, it uses two key assumptions about $f(\cdot)$ that make it difficult to find suitable objective functions. The first assumption is the existence of a reasonably tight *lower bound*, $V_f(\nu, A)$ say, to the leaf average that appears on the right-hand side of Eq. (1). The second assumption is the existence of a reasonably tight node-independent *upper bound*, $U_f(\nu, A)$ say, to $\Delta f(n)$. Furthermore, without loss of generality we can assume that $f(r) \stackrel{\Delta}{=} 0$, simply by subtracting $f(r)$ from each term in Eq. (1) if necessary. Substituting these postulated bounds into Eq. (1), we obtain

$$U_f(\nu, A) \sum_{n \in T-L} p(n) \geq \sum_{n \in T-L} p(n) \cdot \Delta f(n) = \sum_{l \in L} p(l) \cdot f(l) \geq V_f(\nu, A). \tag{2}$$

Recognizing that the left-most summation in Eq. (2) is the average path length in T (and hence the expected number of steps required by A), we see immediately that $V_f(\nu, A)/U_f(\nu, A)$ is a lower bound to the average running time for A when it is applied to I. But since A must not terminate until all the packets in I have been transmitted successfully (and the expected number of such packets is ν by construction), an elementary renewal argument shows that the efficiency of A, when applied to I, is bounded above by

$$C_f(\nu, A) \stackrel{\Delta}{=} \frac{\nu \cdot U_f(\nu, A)}{V_f(\nu, A)}. \tag{3}$$

To convert Eq. (3) into an algorithm-independent upper bound to capacity, we must take the limit $\nu \to \infty$ (since unfavorable boundary effects could decrease the efficiency of A over some steps), and the supremum over all conflict resolution algorithms. There is no need to optimize over $f(\cdot)$ except to tighten the bound, since every objective function used in this way defines a bound. Thus, to find the various bounds, it remains to define suitable functions $f(\cdot)$ and then evaluate

$$C_f \stackrel{\Delta}{=} \limsup_{\nu \to \infty} \sup_{A} C_f(\nu, A). \tag{4}$$

In Pippenger's bound [Pippe81], we let $f(n) = -\log_2 p(n)$. In this case, the right-hand side of Eq. (1) becomes

$$H(L) = -\sum_{l \in L} p(l) \cdot \log_2 p(l), \tag{5}$$

which is the *leaf entropy*, i.e., a measure of the uncertainty of where a given sample path will terminate in the decision tree. Let $L_N = \{l : \sigma(l) = N\}$ be the set of leaves terminating sample paths where exactly N packets were transmitted successfully. Also let $p_N \stackrel{\Delta}{=} \sum_{l \in L_N} p(l) \stackrel{\Delta}{=} \frac{\nu^N}{N!} e^{-\nu}$ be the probability that I contains exactly N packets, and let $p^*(N) \stackrel{\Delta}{=} \max_{l \in L_N} p(l)$ be the highest probability assigned to any leaf in L_N. Since $-\log_2(\cdot)$ is a monotonically *decreasing* function, we can substitute these definitions into Eq. (5) to show that $H(L) \geq -\sum_N p_N \cdot \log_2 p^*(N)$.

Pippenger observed that each leaf $l \in L_N$ determines a partition $U(l) = \{u_1(l), \ldots, u_N(l)\}$ of the interval $I = [0, \nu]$, where $u_k(l) = E(n_k^{(s)})$ is the query subset at node $n_k^{(s)}$ where the kth successful transmission took place along this sample path, $1 \le k < N$, and $u_N(l) = I - \bigcup_{j=1}^{N-1} u_j$. But $p(l)$ cannot be greater than the joint probability that each of its N elements contain exactly one packet. Thus, after overbounding the geometric mean by the arithmetic mean, it can be shown that

$$p(l) \le \prod_{k=1}^{N} u_k e^{-u_k(l)} \le \left(\nu / N\right) e^{-\nu} \tag{6}$$

holds for all $l \in L_N$. Thus, $p^*(N) \le (\nu/N)e^{-\nu}$ and $-\log_2 p^*(N) \ge \nu \log_2 e - N\log_2(\nu/N)$. Substituting this expression into Eq. (6), and recognizing that $\sum_N p_N \left(-N \log_2(\nu/N)\right)$ is non-negative, we see that

$$V_f(\nu, A) \triangleq \nu \log_2 e \tag{7}$$

is a lower bound to $H(L)$.

Now consider the term $\Delta f(n)$ on the left-hand side of Eq. (1), namely

$$\Delta f(n) = \frac{1}{p(n)} \sum_{c \in child(n)} [-\log_2 p(c) + \log_2 p(n)] \cdot p(c) = -\sum_{c \in child(n)} \left(p(c)/p(n)\right) \log_2 \left(p(c)/p(n)\right),$$

which is the *branching entropy* at *n*, i.e., a measure of the uncertainty of the direction in which the given sample path continues from node *n*. Clearly, the way to maximize the branching entropy would be to make each of the three possible continuations (i.e., 'idle', 'success', or 'collision') equally likely. But this (unconstrained) maximum for the branching entropy would fix the efficiency of the algorithm at 1/3; what we really want is the constrained maximum, subject to the requirement that a fraction C_f of the time it uses the 'success' branch. Again, it is clear that in this case, branching entropy is maximized at any node *n* by selecting (independently and at random) the 'success' branch with probability C_f, and either of the other branches with half of the remaining probability. Thus, we have

$$\Delta f(n) \le U_f(\nu, A, C_f) \triangleq -C_f \cdot \log_2 C_f - (1 - C_f) \cdot \log_2\left((1 - C_f)/2\right). \tag{8}$$

Substituting Eqs. (7–8) into Eq. (4), and recognizing that our final result is independent of ν and A, we obtain Pippenger's bound, as the supremum over all values of C_f for which the inequality

$$C_f \ge -C_f \cdot \ln C_f - (1 - C_f) \cdot \ln\left((1 - C_f)/2\right)$$

holds. Computations show that the critical value is $C_f \approx 0.744$.

Molle's bound [Molle82] takes advantage of the memoryless property of the Poisson distribution, through the introduction of a ''helpful genie'' who provides (at no cost) extra information about the state of the conflict resolution beyond the usual 'idle'/'success'/'collision' feedback. In particular, after the algorithm has received the channel feedback from selecting *E(n)*, the genie identifies the *first two* packets (if any) he finds after searching *E(n)* for: (i) previously identified packets, and then (ii) other packets in chronological order. Notice that the genie tells us nothing useful except after collisions, in which case genie's information gives us *complete information* about how to resolve that part of *E(n)* up to the newest packet that was identified (if any), and *no information* about anything else. Consequently, at any node *n* in the conflict resolution tree, $I(n)$ for a 'genie-aided' protocol consists of an *unexamined set* in which the distribution of packets is Poisson with unit intensity, and possibly a list of $\Gamma(n)$ previously-identified packets. Thus, *E(n)* cannot contain anything other than some unexamined Poisson arrivals (with mean $\lambda(n)$, say) and/or some previously-identified packets (say, $\gamma(n)$ of them).

To obtain Molle's bound, we consider an objective function of the form $f(n) = \sigma(n) + \alpha \cdot \Gamma(n)$, where α is any non-negative weight. Clearly $V_f(\nu, A) \triangleq \nu = E[\, \sigma(l)\,]$, since all packets in I (including any that were identified by the genie) must have been transmitted successfully before A terminates. Thus Eq. (3) reduces to $C_f(\nu, A) = U_f(\nu, A)$, which we wish to minimize by carefully selecting the constant α.

Now consider $\Delta f(n)$ for various values of $\lambda(n)$ and $\gamma(n)$. Notice that $f(\cdot)$: (i) does not change following an idle slot; (ii) increases by $1-\alpha\cdot\gamma(n)$ following a successful transmission; and (iii) increases by $\alpha\cdot(2-\min\{\gamma(n), 2\})$ following a collision. Thus:

$$\Delta f(n \mid \gamma(n)) = \begin{cases} \lambda(n)e^{-\lambda(n)} + 2\alpha\cdot[1-(1+\lambda(n))e^{-\lambda(n)}], & \gamma(n) = 0 \\ (1-\alpha)\cdot e^{-\lambda(n)} + \alpha\cdot(1-e^{-\lambda(n)}), & \gamma(n) = 1 \\ 0, & \gamma(n) > 1. \end{cases} \tag{9}$$

Notice that $\alpha > 0.5$ would give $\Delta f(\cdot) > 1.0$ when $\gamma(n) = 0$ and $\lambda(n) >> 1$, so we may as well restrict our attention to the range $0 \le \alpha \le 0.5$. But in this case $1-\alpha \ge \alpha$, so if $\gamma(n) = 1$ then Eq. (9) attains its maximum as $\lambda(n) \to 0$. It is also easy to show that for $\gamma(n) = 0$ and any given α, Eq. (9) attains its maximum at $\lambda(n) = 1/(1-2\alpha)$, or conversely that a given value for $\lambda(n)$ would be optimal for $\alpha = (1-1/\lambda(n))/2$. Thus

$$U_f(\nu, A) = \min_{\lambda(n)\ge 1} \max\left\{ \frac{1+1/\lambda(n)}{2},\ 1-\frac{1-e^{-\lambda(n)}}{\lambda(n)} \right\}. \tag{10}$$

Since the first term in the minimization is a decreasing function of $\lambda(n)$ while the second term is an increasing function, Eq. (10) attains its maximum at the point where the two terms are equal (i.e., $\lambda^* \approx 2.98$, the solution to $3-\lambda = 2e^{-\lambda}$). Substituting λ^* into Eq. (10) yields $C_f \approx 0.673$ for genie-aided algorithms. Since the optimal genie-aided algorithm is free to ignore the hints provided by the genie, $C_f \approx 0.673$ is also an upper bound for unaided algorithms. Cruz and Hajek ([Cruz82]) later established a much stronger genie-aided upper bound by proving that $C_f < 0.613$, even if a ''less-helpful genie'' were to provide (at no cost) enough extra information to reduce the state at each node of the conflict resolution tree to some combination of unexamined Poisson arrivals, previously-identified packets, and disjoint subsets known to contain at least two packets.

Mikhailov and Tsybakov's bound depends on a powerful lemma, which is proven in [Mikha81]. Their lemma showed that the probabilities that an idle slot or a successful transmission results from enabling $E(n)$, $|E(n)| = \varepsilon$, are at most $\phi_0(n) \triangleq e^{-\varepsilon}$ and $\phi_1(n) \triangleq \frac{\varepsilon e^{-\varepsilon}}{1-e^{-\varepsilon}}\cdot(1-\phi_0(n))$ respectively, independent of which algorithm A and node n in its conflict resolution tree we consider. In other words, information about the distribution of packets in I (encoded via the identity of the current node n in the conflict resolution tree) can *never decrease* the probability of finding packets in $E(n)$ below the number attributable to the original Poisson distribution.

To obtain their bound, we consider an objective function of the form $f(n) = |I - I(n)| + \alpha\cdot\sigma(n)$, where the constant α controls a tradeoff between the two goals of resolving points versus transmitting packets as quickly as possible. Since both of these objectives must be fully realized before the algorithm can terminate, it is clear that

$$V_f(\nu, A, \alpha) \triangleq (1+\alpha)\cdot\nu. \tag{11}$$

Now consider the change in $f(\cdot)$ after A has enabled $E(n)$. First, if the result is an idle slot, then all points in $E(n)$ have been resolved and $f(\cdot)$ increases by ε. Second, if the result is a successful transmission, then $\sigma(\cdot)$ increases by one and all points in $E(n)$ have again been resolved, so $f(\cdot)$ increases by $\varepsilon+\alpha$. And finally, if the result is a collision, then $f(\cdot)$ remains unchanged. Thus,

$$\Delta f(n) \le \max_{E(n)} \left\{ \varepsilon\cdot\phi_0(n) + (\varepsilon+\alpha)\cdot\phi_1(n) \right\} = \max_{\varepsilon} \left\{ \varepsilon\cdot\phi_0(n) + (1-\phi_0(n))\cdot\frac{\varepsilon(\varepsilon+\alpha)e^{-\varepsilon}}{1-e^{-\varepsilon}} \right\}. \tag{12}$$

Recognizing that the right-hand side of Eq. (12) is linear combination of ε and and $\varepsilon(\varepsilon+\alpha)/(e^{\varepsilon}-1)$, it is clear that its maximum must occur at one of the extreme points. Thus, since the lemma implies that the minimum and maximum feasible ''weights'' on the first term are 0 and $e^{-\varepsilon}$ respectively, we obtain

$$U_f(\alpha, \varepsilon) = \limsup_{\nu\to\infty} \sup_{A} U_f(\nu, A, \alpha, \varepsilon) = \varepsilon e^{-\varepsilon}\cdot\max\{(\varepsilon+\alpha)/(1-e^{-\varepsilon}),\ 1+\varepsilon+\alpha\}.$$

The two remaining parameters, α and ε, require different treatments. First, since we have no control over how the algorithm A chooses ε, we must assume the most-favorable case to construct an upper bound, i.e.,

$$U_f(\alpha) = \sup_{\varepsilon>0} U_f(\alpha, \varepsilon). \tag{13}$$

Thus, substituting Eqs. (11) and (13) into Eq.(3), we obtain

$$C_f(\alpha) = \frac{U_f(\alpha)}{1+\alpha}, \tag{14}$$

which is an upper bound to capacity for every choice of α. Second, since α is *not* under the algorithm's control, we are free to minimize Eq. (14) with respect to α to obtain the tightest bound. Thus, we have reached the final result, namely that

$$C_f = \min_{\alpha \geq 0} C_f(\alpha) \simeq 0.587$$

is an upper bound to capacity for any protocol. This general approach was later sharpened by Zhang and Berger ([Zhang85) and again by Tsybakov and Likhanov ([Tsyba87]), who showed that $C_f \leq 0.578$ and $C_f \leq 0.568$ respectively.

There are also some tighter bounds available for restricted classes of algorithms. For example, a bound of $C \leq 0.505$ for the class of free access algorithms is readily obtainable using Mikhailov and Tsybakov's technique [Tsyba85b]. In addition, using numerical optimization to sharpen an earlier ''helpful genie'' argument by Molle [Molle81], Panwar, Towsley and Wolf [Panwa85] were able to show that $C \leq 0.5$ holds for the restricted class of algorithms that deliver their packets in First-Come–First-Served order and/or use only nested query sets. This class includes essentially all popular algorithms except for the ones that use free access. It is also widely believed that a bound of $C \leq 0.5$ should also apply to arbitrary algorithms. However, a proof of this conjecture has eluded researchers for more than a dozen years.

5. Delay Analysis

A very important performance metric for multiple access protocols is their delay characteristics. Indeed, we should not forget that the incentive for their development was the expectation that their delay would be low (at least at light traffic conditions). Yet, good analyses of the delay characteristics of such algorithms have appeared relatively late in the development of the area.

In the following sections, we present an overview of some of the basic approaches to packet delay analysis that have appeared in the literature for various combinations of conflict resolution and channel access algorithms. For consistency, we will use a uniform notation, in which the *total delay*, t, experienced by a randomly-chosen ''tagged'' packet is partitioned into the following set of components (refer to Fig. 1):

- t_0: Each newly-generated packet must be assigned to an active set by the channel access algorithm before it can be transmitted. The *initial delay* measures the time elapsed from the moment of arrival for the ''tagged'' packet until this set has been completely specified.
- t_1: With some channel access algorithms (notably window access), there may be a *lag* between the specification of an active set and the start of the corresponding conflict resolution process.
- t_2: Once the active set containing the ''tagged'' packet begins its CRI, one or more algorithmic steps of the CRA will be required before that packet has been transmitted successfully. We call this the *conflict resolution time* for the ''tagged'' packet. We have obtained various conditional results for t_2 in section 2.

5.1. Gated Access Algorithms

Capetanakis was probably the first to attempt the delay analysis for a CRA. His results [Capet78, Capet79, Capet79b] consist of bounds on the mean delay for the STA with gated access. Comparable results can be found in [Masse81].[1] Unfortunately, these bounds are not very tight (they diverge as throughput approaches capacity), and large numbers of ''magic'' constants appear in the derivation. Thus, they do not provide

[1] There is a minor error in Massey's derivation of t_2 that makes his calculated bounds too optimistic. See [Tsyba86] or [Molle92] for a correction.

much insight into the performance of the algorithms.

Below we present an alternate approach due to Molle and Shih [Molle92], who showed how to combine the results we have obtained so far into a complete delay analysis. It should be obvious that under gated access, t_0 (which in this case represents the *residual life* of the ongoing CRI when the tagged packet arrives) and t_2 are the only non-zero components in the total delay for the ''tagged'' packet. Furthermore, the sequence of CRI lengths during the execution of the protocol forms an embedded Markov chain.

We begin by solving for the distribution of CRI lengths in steady state:

$$\vec{\pi} = (\pi_1, \pi_2, \pi_3, \cdots)$$

where π_j is the probability that a randomly chosen CRI will be j slots long. Since we are dealing with a Markov chain, $\vec{\pi}$ can be found from the usual global balance conditions as the solution to the vector equation

$$\vec{\pi} = \vec{\pi}\mathbf{P}. \tag{1}$$

Note that the (i, j)-th entry in the transition matrix, $\mathbf{P}$, represents the probability that a CRI of length i is followed by a CRI of length j slots. But this is just $q_j(\lambda i)$, the j-th term in the CRI length distribution when the initial number of packets at the beginning of a CRI is Poisson distributed with mean λi. Methods for finding the CRI length distribution were discussed in section 2 and, in the case of the STA, the result we need is given by Eq. (2.4:1).

Next, we use the fact that we have Poisson packet arrivals to apply standard renewal-theoretic results to obtain $\tilde{\pi}_j$, the probability that our ''tagged'' arrival joins the system during a CRI of length j:

$$\tilde{\pi}_j = \frac{j\,\pi_j}{\sum_k k\,\pi_k} = \frac{j\,\pi_j}{\mathrm{E}[\,l\,]}.$$

If we condition on the event that our ''tagged'' packet arrives during a CRI of length l, then t_0 has a uniform distribution over the interval $[0, l)$, and the distribution of t_2 should be conditioned on the fact that the ''tagged'' packet will be competing with a group of other packets having a Poisson distribution with mean λl. Thus, assuming we use random addressing, these two conditional distributions are independent of one another and the distribution of their sum factors into the product of their marginal distributions. Therefore, the Laplace transform for the total delay can be written immediately in the form:

$$D^*(\lambda, s) = \sum_{n=0}^{\infty} \tilde{\pi}_n \cdot U^*(n,s) \cdot G(\lambda\, n, e^{-s}) \tag{2}$$

where

$$U^*(n, s) = \frac{1 - e^{-sn}}{ns}$$

is the Laplace transform of a uniform distribution over $[0, n]$, with mean $\bar{u}_n = n/2$, and variance $\sigma^2_{u|n} = n^2/12$, and $G(x,z)$ is the PGF for t_2 given by Eq. (2.4:2). Using standard techniques, we can obtain all the moments of the packet delay from Eq. (2). In particular, the mean packet delay is:

$$\mathrm{E}[\,t\,] = \sum_{n=0}^{\infty} \tilde{\pi}_n \cdot [\,n/2 + \mathrm{E}[\,t_2(\lambda n)\,]\,]. \tag{3}$$

5.2. Free Access Stack Algorithms

A regenerative approach to finding the mean delay for Stack Algorithms is presented in [Fayol85]. The basic idea is that under free access, each CRI boundary forms a renewal point for the packet delay process, so that the mean delay $\mathrm{E}[\,t\,]$ can be obtained as

$$\mathrm{E}[\,t\,] = \frac{D(\lambda)}{N(\lambda)}, \tag{1}$$

where $D(\lambda)$ represents the expected value of the cumulative sum of the total delays for all packets served in the

CRI, and $N(\lambda)$ represents the expected number of packets[1] contributing to that sum. Fortunately, from the renewal nature of the process,

$$N(\lambda) = \lambda\, L(\lambda)\,,$$

where $L(\lambda)$ is the mean CRI length. It remains to find an expression for $D(\lambda)$. The approach in [Fayol85] consisted of defining the set of random variables $\{d_m\}$ representing cumulative delay conditioned on collision multiplicity (rather than on traffic intensity, which we need), which must satisfy:

$$d_m = \begin{cases} 0 & ,\ m = 0 \\ 1 & ,\ m = 1 \\ m + d_{i+x_1} + d_{m-i+x_2} + (m-i)\, l_{i+x_1} & ,\ m \geq 2 \end{cases}$$

where the binomial random variable i represents the number of colliding packets that join the first subgroup, and the Poisson random variables x_1 and x_2 represent the number of new arrivals that join each of those subgroups. Observe that the same random variables i and x_1 appear in several places in the above equation, so finding distributions is hard because the terms in the functional equation are not independent. Fortunately, independence is not an issue for the mean, where $D_m \stackrel{\Delta}{=} \mathrm{E}[\, d_m\,]$. Thus, after some effort, and recognizing that

$$D(x) \stackrel{\Delta}{=} \sum_{m=1}^{\infty} D_m\, \frac{x^m}{m!}\, e^{-x}\,,$$

they were able to obtain a functional equation for $D(x)$.

We can obtain the functional equation more directly using the approach in section 2.3. Again, we compare the mean cumulative delay over an ordinary CRI starting from traffic intensity x and an auxiliary CRI in which we unconditionally skip the root node. In the ordinary CRI, an average of x packets accumulate one slot of delay at the root node, which does not appear in the auxiliary CRI. On the other hand, if the root has a non-conflict, the auxiliary CRI will accumulate more delay. In particular, given an idle slot at the original root node, the auxiliary system contains two (independent) ordinary CRIs each with Poisson intensity λ. Similarly, given a success at the original root node, the auxiliary system contains two (independent) CRIs, one of which is ordinary with intensity λ and the other includes a single ''extra'' packet. Hence

$$\begin{aligned} D(x) - x \;&=\; D(px+\lambda) + D(\bar{p}x+\lambda) + \bar{p}x\, L(px+\lambda) - e^{-x}\,[2\,D(\lambda)] - x\, e^{-x}\big[D^{(+1)}(\lambda) + D(\lambda) + \bar{p}\, L(\lambda)\big] \\ &=\; D(px+\lambda) + D(\bar{p}x+\lambda) + \bar{p}x\, L(px+\lambda) - (1+x)\, e^{-x}\,[2\,D(\lambda)] - x\, e^{-x}\big[D'(\lambda) + \bar{p}\, L(\lambda)\big]\,. \end{aligned}$$

5.3. The Window Access Method of Huang and Berger

Huang and Berger [Huang85] described a methodology that is essentially a generalization of the approach we took for gated access above. It may be applied to the class of blocked access algorithms (since they regard the window size as a random variable, and hence include gated access). They also allowed an arbitrary conflict resolution algorithm to be used, although they had to assume that it satisfies the following ''separability'' condition [Polyz93] which states that the CRI does not terminate until all packets with arrival times in the window have been successfully transmitted. Thus, the ''tree pruning'' optimization in the 0.487 algorithm which can lead to a regression on the window boundary cannot be directly accommodated.

In their method, the total delay is partitioned into two components: the delay until the beginning of the CRI in which the ''tagged'' packet will be successfully transmitted (which is t_0+t_1 in our notation); and the conflict resolution time (which we call t_2). They defined the lag, h, at the end of a CRI to be the age of the left (i.e., older) edge of the window. Their approach involves finding the distribution of h in steady-state, and then using it to determine the unconditional means of $t_0 + t_1$ and of t_2.

---

[1] Remember that $N(\lambda) \geq \lambda$, since new packets can join an ongoing CRI under free access.

Even in this more general setting, the length of each CRI must be an integer number of slots. However, we cannot assume that $w(h)$, the window size given a lag of h, will be integer valued (or even rational). Thus, the evolution of the lag is treated as a continuous state embedded Markov process with probability density function $f_h(u)$ in steady state. The global balance condition we used in Eq. (5.1:1) now leads to the following integral equation:

$$f_h(v) = \int_1^\infty K(v \mid u)\, f_h(u)\, du\,, \tag{1}$$

where

$$\int_1^\infty f_h(u)\, du = 1$$

and $K(v \mid u)$ is the conditional probability density function of the current lag, given the previous one. If the conflict resolution algorithm is separable then this kernel can be obtained from the distribution of CRI lengths, i.e.,

$$K(v \mid u) \;=\; \sum_{l=1}^{\infty} q_l(\lambda \cdot w(u))\; \delta(v - u - l + w(u))\,, \tag{2}$$

where $\delta(t)$ is the unit impulse.

The distribution for h allows us to combine the conditional values for t_0+t_1 and for t_2 to obtain the unconditional delay components. That is, if $t_0(h)$, $t_1(h)$ and $t_2(h)$ represent the conditional values for the three delay components, given a lag of h, then we let:

$$\mathrm{E}[\, t\,] \;=\; \frac{\displaystyle\int_1^\infty \lambda \cdot w(h)\, f_h(u) \left[\, \mathrm{E}[\, t_0(u)\,] + \mathrm{E}[\, t_1(u)\,] + \mathrm{E}[\, t_2(u)\,]\,\right] du}{\displaystyle\int_1^\infty \lambda \cdot w(h)\, f_h(u) du}\,. \tag{3}$$

Huang and Berger interpreted the above equation as a ratio of expectations for the accumulated total delay experienced over a single CRI to the total number of packets served in a single CRI. The denominator can be simplied using the relation

$$\mathrm{E}[\, N\,] \;=\; \lambda \cdot \mathrm{E}[\, w\,]\,. \tag{4}$$

Similarly, the numerator can be reduced to:

$$\mathrm{E}[\, N(h) \cdot (t_0(h) + t_1(h))\,] \;+\; \mathrm{E}[\, N(h) \cdot t_2(h)\,]\,.$$

Since we have Poisson arrivals, the arrival times for the packets are uniformly distributed over the window, so the first two terms in the numerator become:

$$\mathrm{E}[\, N(h) \cdot (t_0(h) + t_1(h))\,] \;=\; \mathrm{E}[\, \lambda w(h) \cdot (h - w(h)/2)\,] \;=\; \lambda\big(\, \mathrm{E}[\, w(h) \cdot h\,] - \mathrm{E}[\, (w(h))^2\,]\,/\,2\big)\,. \tag{5}$$

It remains to determine the final term in the numerator, i.e., $\mathrm{E}[\, N(h) \cdot t_2(h)\,]$. Huang and Berger chose to condition on the collision multiplicity, n, rather than the Poisson sample size and found that

$$\mathrm{E}[\, N(h) \cdot t_2(h)\,] \;=\; \int_1^\infty \lambda \cdot w(u)\, f_h(u) \left[\, \sum_{n=1}^{\infty} \frac{(\lambda\, w(u))^{n-1}}{(n-1)!}\, e^{-\lambda\, w(u)} \cdot \mathrm{E}[\, t_2 \mid n\,] \,\right] du\,. \tag{6}$$

Huang and Berger also applied the above methodology to the 0.487 algorithm [Huang86]. This algorithm violates the separability assumption because of the window regression effect from ''tree pruning.'' In this case, we have $w(u) = \min\{u, \Delta\}$ and we require a new random variable $\delta(u)$ to represent the length of the resolved part of the window when the CRI terminates. Huang and Berger proved that Eqs. (4)–(6) still hold if we substitute $\delta(u)$ in place of $w(u)$. However, they were unable to obtain the kernel of the integral equation exactly and had therefore resorted to bounds in this case.[1]

[1] This difficulty could be avoided by defining the kernel of the integral equation in terms of the *augmented* CRI length for the 0.487 Algorithm [Polyz93]. Note, also that their results for the 0.487 Algorithm are unreliable for large λ, which we attribute to numerical problems in the calculation of $f_h(x)$ in this case.

5.4. The Regenerative Method of Georgiadis *et al.*

In [Georg87b], Georgiadis *et al.* describe a general methodology for applying renewal theory to the analysis of any multiple access protocol that induces a regenerative delay process. They define a *session* (or busy period) to consist of a sequence of one or more consecutive CRIs starting and ending from a regeneration point. As we have seen in the analysis of stack algorithms by Fayolle *et al.*, the mean delay, E[t] , can be obtained as a ratio of expectations over a regenerative cycle, i.e.,

$$\mathrm{E}[\,t\,] = \frac{\mathrm{E}[\,d^*\,]}{\mathrm{E}[\,n^*\,]}$$

where the random variable d^* represents the cumulative delay experienced by all packets over a session, and n^* represents the number of packets transmitted in the same interval. Obviously, we have

$$\mathrm{E}[\,n^*\,] = \lambda\, Y$$

where λ is the arrival rate and Y is the mean session length.

We will use the FCFS 0.487 Algorithm as an example to illustrate the method. Obvious regeneration points in this case are the moments when, at the beginning of a CRI, the lag of the algorithm has a given value, most naturally when the lag is equal to one slot.[1] If we define the random variable y_h as the time to return to lag one, starting from lag h, we have

$$y_h = \begin{cases} 1\,, & \text{if } l(\lambda h) = 1 \text{ and } 1 \le h \le \Delta \\ l(\lambda\, w(h)) + y_{h-\delta(h)+l(\lambda\, w(h))}\,, & \text{o.w.} \end{cases}$$

where Δ is the maximum window size, $w(h) = \min\{h, \Delta\}$ is the chosen window size, $\delta(h)$ is the resolved part of the window, and $l(x)$ the CRI length starting from a Poisson set with mean x. Defining $Y_h \stackrel{\Delta}{=} \mathrm{E}[\,y_h\,]$ we get

$$Y_h = \mathrm{E}[\,l(\lambda h)\,] + \sum_{\substack{r,s \\ s\neq 1}} \Pr\{\,\delta(h) = r \text{ and } l(\lambda h) = s\,\}\; Y_{h-r+s}\,, \qquad 1 \le h \le \Delta$$

$$Y_h = \mathrm{E}[\,l(\lambda \Delta)\,] + \sum_{r,s} \Pr\{\,\delta(h) = r \text{ and } l(\lambda \Delta) = s\,\}\; Y_{h-r+s}\,, \qquad h > \Delta$$

which is an infinite system of linear equations that in matrix form becomes:

$$\vec{Y} = \vec{B} + \mathbf{C}\cdot\vec{Y}. \tag{1}$$

Notice that by definition, $Y \stackrel{\Delta}{=} Y_1$, and therefore, a solution of the above linear system would provide us with the denominator of the expression for the mean delay.

In a similar way, we can obtain a system of linear equations for the cumulative delay, by considering a particular CRI and defining the random variable for the cumulative delay until the return to lag one (starting from lag h), as d^*_h. As above, we define $D^*_h \stackrel{\Delta}{=} \mathrm{E}[\,d^*_h\,]$ and $D^* \stackrel{\Delta}{=} D^*_1$.

Letting $\omega(h)$ and $\psi(h)$ be the cumulative values for t_0 and t_2 for all packets that depart from the system in a CRI that began with a lag of h, and recognizing that $t_1 \stackrel{\Delta}{=} h-\Delta$ for these same packets, we obtain the following FE for d^*_h (as a random variable)

$$d^*_h = \begin{cases} \omega(h) + \psi(h)\,, & l(\lambda h)=1 \text{ and } 1 \le h \le \Delta \\ \omega(h) + \psi(h) + d^*_{h-\delta(h)+l(\lambda h)}\,, & l(\lambda h) > 1 \text{ and } 1 \le h \le \Delta \\ \omega(h) + \psi(h) + (\,h-\Delta\,)\,N(h) + d^*_{h-\delta(h)+l(\lambda \Delta)}\,, & h > \Delta \end{cases}$$

Taking expectations, we obtain a FE for the mean:

---

[1] This is the minimum value possible (and not zero) because of the particular way the lag is defined.

$$D^*_h = \mathrm{E}[\,\omega(h)\,] + \mathrm{E}[\,\psi(h)\,] + \sum_{\substack{r,s \\ s\neq 1}} \Pr\{\,\delta(h)=r \text{ and } l(\lambda h)=s\,\}\; D^*_{h-r+s}$$

$$= N(h)\;\mathrm{E}[\,t_0(\lambda h)+t_2(\lambda h)\} + \sum_{\substack{r,s \\ s\neq 1}} \Pr\{\,\delta(h)=r \text{ and } l(\lambda h)=s\,\}\; D^*_{h-r+s}\,, \qquad 1\le h\le \Delta$$

$$D^*_h = \mathrm{E}[\,\omega(h)\,] + \mathrm{E}[\,\psi(h)\,] + N(h)\,[h-\Delta] + \sum_{r,s} \Pr\{\,\delta(h)=r \text{ and } l(\lambda\Delta)=s\,\}\; D^*_{h-r+s}$$

$$= N(h)\big[\,\mathrm{E}[\,t_0(\lambda\Delta)\,] + \mathrm{E}[\,t_2(\lambda\Delta)\,] + h-\Delta\big] + \sum_{\substack{r,s \\ s\neq 1}} \Pr\{\,\delta(h)=r \text{ and } l(\lambda h)=s\,\}\; D^*_{h-r+s}\,, \qquad h>\Delta$$

i.e., a system of linear equations of the form

$$\vec{D^*} = \vec{B'} + \mathbf{C}\cdot\vec{D^*} \tag{2}$$

that uses the same matrix **C** as the system of Eq. (1).

The method as described so far is equivalent to the approaches we saw in gated access and in Huang and Berger's method in the sense that they all involve using global balance conditions to find the steady-state solution to an embedded Markov chain defined at CRI boundaries. In particular, this would lead to all of the problems associated with solving (numerically) a system of linear equations of infinite degree. However, the methodology of Georgiadis *et al.* also includes an important contribution in the area of solution techniques.

In [Georg87b], the authors considered iterative solutions to Eqs. (1)–(2) of the type

$$\vec{Y}^{(k+1)} = \vec{B} + \mathbf{C}\cdot\vec{Y}^{(k)}\,,$$

and found a sufficient condition for convergence to the solution vector to be monotonic. If for some k we have that $Y_h^{(k+1)} \le Y_h^{(k)}$ holds for every h, then they showed that the sequence $\vec{Y}^{(k)}, \vec{Y}^{(k+1)}, \vec{Y}^{(k+2)} \cdots$ are all upper bounds to the solution, and $\vec{Y}^{(k+j)} \to \vec{Y}$ as $j \to \infty$. Similarly, if $Y_h^{(k+1)} \ge Y_h(k)$ holds for every h, they showed that the sequence $\vec{Y}^{(k)}, \vec{Y}^{(k+1)}, \vec{Y}^{(k+2)} \cdots$ is a monotonically convergent sequence of lower bounds. The authors use this condition to determine, analytically, a pair of linear functions to bound the solution to $\vec{Y}$:

$$Y_h^l \triangleq \alpha_l\, h + \beta_l \;\le\; Y_h \;\le\; \alpha_u\, h + \beta_u \triangleq Y_h^u$$

and a pair of quadratic functions to bound the solution to D^*:

$$D^*_h{}^l \triangleq \gamma_l\, h^2 + \delta_l\, h + \varepsilon_l \;\le\; D^*_h \;\le\; \gamma_u\, h^2 + \delta_u\, h + \varepsilon_u \triangleq D^*_h{}^u$$

By combining these bounds, they obtain the following bounds on the mean packet delay

$$\frac{D^*_1{}^l}{\lambda\, Y_1^u} \;\le\; \mathrm{E}[\,t\,] \;\le\; \frac{D^*_1{}^u}{\lambda\, Y_1^l}\,.$$

5.5. The Modified Regenerative Method of Tsybakov *et al.*

Yet another variation on the regenerative approach has been suggested by Tsybakov and Privalov [Tsyba92]. The innovation in their approach is to ''fold'' the two systems of equations (defining Markov chains for the quantities appearing in the numerator and denominator, respectively) into a single system of equations and a scalar parameter. Continuing with the notation of the previous section, we let

$$\mathrm{E}[\,t\,] = \frac{D^*_1}{\lambda\, Y_1}\,, \tag{1}$$

represent the regenerative solution to the mean packet delay. Tsybakov *et al.* recognized that, because we have a linear system, we can subtract $\lambda\ \mathrm{E}[\,t\,]$ times Eq. (5.4:1) from Eq. (5.4.2) to obtain

$$\vec{\gamma} = \vec{B^*} - \lambda \vec{B} + \mathbf{C} \cdot \vec{\gamma}, \tag{2}$$

where $\gamma_h = D^*_h - \lambda \cdot \mathrm{E}[\, t \,] \cdot Y_h$. Substituting Eq. (1) into the component of the solution corresponding to $h=1$ and simplifying, we obtain:

$$\gamma_1 \stackrel{\Delta}{=} D^*_1 - \lambda \cdot \mathrm{E}[\, t \,] \cdot Y_1 = D^*_1 - \lambda \cdot \frac{D^*_1}{\lambda Y_1} \cdot Y_1 \stackrel{\Delta}{=} 0\,. \tag{3}$$

In other words, if we can ''guess'' the value of $\mathrm{E}[\, t \,]$, we need only solve one (instead of two) systems of linear equations to verify that our guess is correct. Furthermore, if the calculated value of γ_1 is non-zero, we know how to improve our ''guess'' since a negative value implies our ''guess'' is too large and vice versa.

In general, it is not clear that ''guessing'' the value of $\mathrm{E}[\, t \,]$ would be easier than solving a second system of equations. However, the real advantage of this approach becomes evident when one tries to optimize the performance of the protocol. For example, suppose we wish to find the optimal state-dependent window size function that minimizes $\mathrm{E}[\, t \,]$ for a given conflict resolution algorithm. For a CRI starting from a given lag, h, we can select any window size we like up to a maximum of size h (or even insert a rest period to allow larger windows to be selected). In general, every window size will lead to a different value for D^*_h, but we cannot simply take the one that minimized D^*_h since it does not account for the fact that Y_h is changing too. ''Folding'' the two systems of equations together eliminates this problem, so the optimal policy for a lag of h can be found simply by selecting $\Delta(h)$ to minimize γ_h.

5.6. Window Access via Queueing Theoretic Decomposition

This approach follows the decomposition of t into t_0, t_1 and t_2 that we defined in section 5.0. The analysis of the marginal distribution for each component is done completely independently, and the results are then combined to obtain the overall packet delay. It first appeared in [Polyz84] where the mean delay for the SRA, combined with the SWA was analyzed. The results were subsequently extended to produce the (steady-state) *distribution* of the packet delay for a finite (in general non-homogeneous) population with a Bernoulli-per-window packet arrival model [Polyz87], to handle the (non-simplified) Window Algorithm [Polyz87b], and a variety of conflict resolution algorithms including the FCFS 0.487 Algorithm [Polyz93].

The key idea in this approach is to take advantage of the fact that in many cases, the three delay components defined above are *mutually independent* random variables. To see this, consider the delay for a randomly chosen ''tagged'' packet assuming random addressing:

- t_0 depends only on the *position* of the tagged packet's arrival within the window;
- t_1 depends only on the arrival process *before* the window of arrival for the tagged packet; and
- t_2 depends only on the *number of other packets* that arrived in the same window.

Observe that t_1 depends on the arrival process over a disjoint interval compared with that for t_0 and t_2. Furthermore, t_0 and t_2 are independent of one another[1] because the position of a random arrival is independent of the total number of arrivals in a Poisson process.

If the SWA is used for channel access, then it is easy to see how to take advantage of the above decomposition. Since all windows are of constant size, the distributions for t_0 and t_2 are independent of the lag. Furthermore, a randomly-chosen packet is equally likely to fall into any window, so the distribution of t_1 is the same whether we sample the state of the lag from a randomly chosen window or a randomly chosen packet. In this case, we recognize that t_1 is equivalent to the waiting time in a discrete time D/G/1 queueing system [Servi86], in which each window is a ''customer'' whose ''service time'' occupies the channel for one CRI. Thus, we can write down immediately the PGF for t as:

---

[1] Unless we use arrival time addressing, like the FCFS 0.487 algorithm, in which case we must work with $t_{0,2} \stackrel{\Delta}{=} t_0 + t_2$ to get distributional results.

$$D^*_\Delta(\lambda,s) \;=\; U^*(\Delta,s)\; W(\Delta,e^{-s})\; G(\lambda\Delta,\, e^{-s})$$

where $U^*(\Delta,s)$ represents (the Laplace transform of) a uniform distribution over $[0,\Delta)$, $W(\Delta,z)$ represents the (PGF of the) D/G/1 waiting time distribution, and $G(\lambda\Delta,z)$ represents the (PGF of the) conflict resolution time distribution.

If the WA is used for channel access, then the above D/G/1 model for the lag is not applicable. However, if we restrict our choices for the maximum window size to $\Delta \in \{2, 3\}$, then we can still obtain an exact solution to the distribution of t_1 by solving an elementary queueing system. Thus we introduce a series of problem reductions to show that the lag is equivalent to the waiting time in a queueing system with ''Moving Server'' overhead [Molle89]. First, we expand the small windows to the size of the large ones, without however, modification to the number of arrivals in each window. Clearly this transformation does not affect the waiting times because it does not affect the busy periods (i.e., the time periods when the algorithm chooses large windows and there is possibility for $t_1 \neq 0$). Then, we notice that if we ''shorten'' both arrival and service time axes by removing one slot from the interarrival time between customers i and $i+1$ and from the service time for customer i, *the waiting times for every customer remain exactly the same.* To see this, consider the standard, general, queueing system equation (see for example [Cohen69]):

$$\omega_{n+1} = \max\,\{\,\omega_n + l_n - \alpha_{n+1}\,,\; 0\,\} \tag{1}$$

where, ω_n is the waiting time for the n-th customer, l_n the service time for the n-th customer (the length of the n-th epoch in our case), and α_{n+1} the interarrival time between the n-th and the $(n+1)$-st customers. Therefore, since every service time and interarrival time is at least 1 slot long, we can apply this transformation everywhere. The result is a new system with service times $\tilde{l} = l - 1$, and constant interarrival times $\tilde{\Delta} = \Delta - 1$. However, some service times (i.e., those corresponding to no-conflict windows) are now zero after the transformation, so we can ''erase'' them if we like, without affecting the waiting times of the remaining customers. In this case we find that the remaining customers exhibit almost geometric interarrival times (in units of $\tilde{\Delta}$) and independent service times. The only difference is that when the system is idle the probability of an arrival is q_0 instead of q, the probability of an arrival during a busy period.

There is, however, one more complication. Unlike every other non-erased customer, the first customer in each busy period comes from an (initially) small window. Its CRI is distributed according to $Q^{(c)}(\lambda, z)$ and not $Q^{(c)}(\lambda\Delta,z)$, where $Q^{(c)}(x,z)$ is the conditional PGF for CRIs that begin with a conflict. The question is now whether this system is any easier to analyze than the original one. In some interesting cases, the transformed system can be analyzed as a Generalized Busy Period discrete time M/G/1 queue [Klein76]. The only requirement is a restriction on $\tilde{l}$ such that $\tilde{\Delta}$ can be used as the discrete time unit, namely, that for all possible values of $\tilde{l}$, we have $\tilde{l}$ **mod** $\tilde{\Delta} = 0$ or

$$(\,l - 1\,) \;\textbf{mod}\; (\,\Delta - 1\,) = 0\,.$$

For any separable algorithm, this condition is obviously satisfied for $\Delta = 2$, since l is an integer-valued random variable. Furthermore, for the STA it is guaranteed that l is an odd integer and thus this condition is also satisfied when $\Delta = 3$.[1]

The distribution of the delay for a discrete time M/G/1 queue with generalized busy periods is given by

$$W_b(z) = \frac{1-\rho}{\rho_0}\;\frac{q_0\,[\,B_0(z) - 1\,]}{z - 1 - q\,[\,B(z) - 1\,]}$$

where $B_0(z)$ and $B(z)$ represent the respective PGFs for exceptional and ordinary service times, q_0 and q represent the respective Bernoulli arrival probabilities for exceptional and ordinary customers, and the parameters ρ_0 and ρ are defined by

$$\rho_0 = \frac{L\,(\lambda) - 1}{\Delta - 1} \quad \text{and} \quad \rho = \frac{L\,(\Delta\lambda) - 1}{\Delta - 1}\,.$$

Then the distribution of t_1 is given by

---

[1] A proof of this condition in a more general setting is provided in [Polyz87c].

$$W(\Delta,z) \;=\; W_b(z^{\Delta-1}) \;=\; \frac{1-\rho}{\rho_0}\;\frac{Q(\lambda,z)-z}{z^{\Delta}-Q(\Delta\lambda,z)}\;.$$

Since t_0, t_1, and t_2 conditioned on the tagged packet coming from a window of a certain size are independent, the final result, the PGF of the total delay can then be obtained as the product of the PGFs of the three components of the delay for packets of large windows, and is just the distribution of t_2 for packets of small windows. The overall delay distribution is then given by:

$$D^*_{\Delta}(\lambda,s) = u\;U(\Delta,s)\;W(\lambda,e^{-s})\;G(\lambda\Delta,e^{-s}) + (1-u)\;G(\lambda,e^{-s})$$

where

$$u = \frac{\Delta\,\rho_0}{1-\rho+\Delta\,\rho_0}$$

represent the proportion of packets that belong to large windows.

We now turn our attention to the celebrated FCFS 0.487 Algorithm. Here we have to address a number of additional complications. First, we have to deal with arrival-time addressing (to obtain a true FCFS system) which requires joint treatment of the initial and final components of the delay. We have obtained a FE on the transform of the sum of t_0+t_1 in section 2.2.3. Furthermore, because the initial windows for successive CRIs may overlap we must be careful with packets that are part of more than one initial window; they must be included in the window that leads to their successful transmission; t_0 is then the time until the right end of the initial size of that window.

Let us define the n-th window to be the interval $(\tau_n,\ \tau_n+w_n]$. Up to this point, we have assumed that a separability condition, i.e., $\tau_{n+1} \;=\; \tau_n + w_n$ holds. We now address the *window regression* feature in the 0.487 Algorithm that makes it non-separable. That is, we now assume that $\tau_{n+1} \;=\; \tau_n + w_n - r_n$, where r_n represents the portion of the n-th window that is left unexamined at the end of the epoch. We note that the distribution of r_n is non-negative, and that it is *positively correlated* with the epoch length, l_n. This complicates the approach of modeling t_1 as the waiting time in a queueing system, since we are faced with a queueing problem which has both an *irregular* sequence of interarrival times, $\alpha_n \;=\; w_n - r_n$, and *dependence* between one customer's service time, l_n, and the next customer's interarrival time, α_{n+1}. These complications can be avoided by transforming our system into an *augmented* queueing problem, in which we use $\tilde{l}_n \stackrel{\Delta}{=} l_n + r_n$ as the augmented service time, and $\tilde{\alpha}_{n+1} \stackrel{\Delta}{=} \alpha_{n+1} + r_n$ as the augmented interarrival time. Substituting these augmented quantities into Eq. (1), we see that

$$\omega_{n+1} \;=\; \max\{\,\omega_n + \tilde{l}_n - \tilde{\alpha}_{n+1},\,0\,\} = \;\max\{\,\omega_n + (l_n + r_n) - (\alpha_{n+1} + r_n),\,0\,\} = \max\{\,\omega_n + l_n - \alpha_{n+1},\,0\,\}\,,$$

so it is clear that this transformation does not introduce any approximations into our analysis. Furthermore, because $\tilde{\alpha}_n = \Delta$ and the memoryless property of the Poisson distribution, it is clear that the transformation has decoupled the customer interarrival process from the service time process. The ''cost'' of this change is that we must now solve for the joint statistics of l_i and r_i. We have addressed this problem in section 2.2.3. Solving Eq. (2.2.3:2) we can obtain the distribution of the augmented epoch length.

To use this approach with the WA we need to specify the statistics of the window that initiates the Generalized Busy Period (GBP). This is always going to be a 1 slot long window. On the other hand, all windows of the GBP, except possibly the last, are of the maximum size, Δ. The only intermediate window sizes can appear at the end of the GBP, as the GBP is completing, but the lag has not grown to Δ yet. The windows in this last category (and the packets that are transmitted during their epochs), are a very small proportion of all windows. A basic approximation we make in the analysis is to assume that windows are either of size 1 or Δ.

Fig. 5 compares mean packet delay for various channel access algorithms (WA, SWA, and GA) with the binary STA for conflict resolution. Fig. 6 shows mean, standard deviation, and percentiles (90%, 95%, and 99%) of the packet delay for the binary STA/WA ($\Delta = 3$). Finally, in Fig. 7 we show mean and standard deviation for various CRAs (STA, optimally biased MTA, 3CA, and FCFS 0.487) with the SWA for channel access ($\Delta = 3$).

5.7. Limited Sensing Algorithms

The delay analysis of limited sensing algorithms is generally quite complex. Few results are available for these algorithms, and the methods used to derive them are not well documented in the literature. To see where the complexity arises, consider the application of the regenerative method to the limited sensing version of the 0.487 algorithm [Humbl86, Georg87]. In this case, we would need to define a two dimensional embedded Markov chain at the CRI boundaries. As before, one dimension corresponds to the lag, *h*. However, in this case the unfinished part of the arrival time axis at time τ will, in general, consist of a sequence of unresolved intervals of total length *h* (the ''virtual'' lag) that have been ''left behind'' by the channel access algorithm. The other dimension represents the time elapsed, *v*, since the last synchronization event at which point recently generated packets were able to join in. Thus, the FEs given in section 5.4 would require significant modifications. Furthermore, their solution would be of greater numerical complexity because of the added dimension.

On the other hand, the analysis of LCFS preemptive window algorithms is relatively straightforward. Below we summarize Woodruff's decomposition approach [Woodr92] to the case Δ=2 in combination with any conflict resolution algorithm. Woodruff's method easily extends to $\Delta = 3$ as long as the CRI lengths are always odd (as they are for the binary STA). Other combinations can be solved using a regenerative approach [Tsyba85].

Woodruff observed that the only non-zero components of the delay are t_0 (which is uniformly distributed over a constant-sized window) and t_2. Furthermore, they are independent, so we can obtain distributional results by combining the marginal distributions of each component. Thus, the only complication we face is accounting for the *interruptions* to the given CRI that occur whenever we find ourselves at a window boundary at the end of an algorithmic step.

Let $Q(x, z)$ be the CRI length distribution under blocked access, as shown in section 2. Under LCFS preemptive window access, each CRI will require the same number of algorithmic steps, but it will be *interrupted* after every step except the first one. Let the random variable $i(x)$, with PGF $I(x, z)$, represent the length of an interruption when the traffic intensity is *x* packets/window. Observe that $z\,I(x, z)$ is therefore the PGF for the number of slots between the completion time of the one algorithmic step by the ''tagged'' CRI and its next algorithmic step, given that they are separated by an interruption. Thus, we obtain the following FE for the distribution of interrupt lengths:

$$I(x, z) = z\,\left[Q(x, \zeta) / \zeta\right]_{\zeta = z\,I(x, z)}$$

or

$$I^2(x, z) = Q(x, z\,I(x, z)), \qquad \Delta = 2,$$

from which we obtain the following expression for its mean:

$$\mathrm{E}[\, i(x)\,] = \frac{\mathrm{E}[\, l(x)\,]}{2 - \mathrm{E}[\, l(x)\,]}, \qquad \Delta = 2.$$

The conflict resolution time distribution, including interruptions, can be found as follows. Let

$$G(x, z) \stackrel{\Delta}{=} \sum_{i=1}^{\infty} g_i(x)\, z^i$$

represent the conflict resolution time distribution under blocked access, where $g_1(x) = e^{-x}$ and $g_2(x) = (e^{-x/2} - e^{-x})/2$. Then we have:

$$G_{LCFS}(x, z) = g_1(x)\,z + g_2(x)\,z^2 + \left(1 - g_1(x) - g_2(x)\right) z^2 \left[\frac{G(x, \zeta) - g_1(x)\,\zeta - g_2(x)\,\zeta^2}{\left(1 - g_1(x) - g_2(x)\right)\zeta^2}\right]_{\zeta = zI(x, z)},$$

and

$$\mathrm{E}[\, t_{2,LCFS}(x)\,] = \mathrm{E}[\, t_2(x)\,] + \mathrm{E}[\, i(x)\,]\,\left[\mathrm{E}[\, t_2(x)\,] - 2 + e^{-x}\right].$$

6. Extensions to Non-Standard Environments

In this section we would like to bring to the reader's attention a few studies that have departed from the standard assumptions for the ALOHA-type channel we gave in section 1.1, and have either developed new algorithms to better suit their environments, or have evaluated the performance of existing CRAs in different environments. Unfortunately, space does not permit us to provide an exhaustive listing of all proposed schemes and analytical results. Thus, we decided not to include schemes that incorporate CRAs as a component of a pure reservation system, although CSMA type systems are, in some sense, implicit reservation schemes. We also do not discuss schemes that require additional information to be included on the transmitted packets.

We have organized the various contributions we present into a number of basic categories. However, this is not always an easy task because some works span multiple categories and also because we felt we had to limit the number of areas; therefore, some papers do not fit perfectly in the category we discuss them in. Nevertheless, we hope that our approach is more useful than a simple listing of references.

6.1. Type of Feedback

A major characterization of the environment comes from the type and timing of the feedback that is available to the stations. The standard assumptions call for either binary feedback of the type ''conflict—no-conflict'' or ternary feedback of the type ''idle—success—conflict.'' Two other types of binary feedback, ''success—failure'' and ''something—nothing,'' are considered in [Berge84]. In general, these types of feedback are more difficult to work with and algorithms for these environments attain lower capacities. In [Pater89b] the authors consider binary ''success—failure'' feedback and propose and analyze a limited sensing algorithm that achieves a capacity of 0.322. Tsybakov and Beloyarov [Tsyba90] have developed another algorithm for the same environment with capacity $1/e \approx 0.367$; they obtain delay upper bounds for protocols using ''success—failure'' and ''nothing—something'' feedback in [Tsyba90b].

Massey and Mathys [Masse85] consider the standard ALOHA-type channel but with no feedback. It is shown that the capacity under these conditions is $1/e \approx 0.367$, i.e., equal to the capacity of the (stabilized) Slotted ALOHA protocol. Tsybakov and Likhanov [Tsyba83] investigate a similar problem.

On the other hand, Tsybakov [Tsyba80c], Georgiadis and Papantoni-Kazakos [Georg82], and Georgiopoulos *et al.* [Georg84] have investigated the case where more than ternary feedback is available, i.e., when stations know not only that a conflict occurred, but also the number of packets involved (i.e., the multiplicity of the conflict), up to a given level. Energy level detectors were suggested in [Georg82] as a way to implement this type of feedback in conjunction with some (additive) channels. The idea of energy level detectors was applied to the ''nothing—something'' feedback model by Mehravari [Mehra88], who developed an interval mixing CRA for the case of ''nothing—something—something-else'' feedback. Greenberg *et al.* [Green87] and Cidon and Sidi [Cidon88] present and analyze algorithms that allow stations to compute cooperatively a stochastic estimate of the conflict multiplicity as a function of the (ternary) feedback history. Combining this estimation procedure with a CRA leads to hybrid protocols, the most efficient of which resolve conflicts about 20 per cent faster on average than comparable algorithms without conflict multiplicity estimation.

A similar but different approach is taken by Komlos and Greenberg [Komlo85]. The authors develop a non-adaptive algorithm that resolves conflicts among k out of n stations in time $\Theta(k+k\log(n/k))$ in the worst case (same as for the STA with deterministic addressing), assuming that the conflict multiplicity k is given a priori. A similar problem is considered in [Khasi89]. Greenberg and Winograd [Green85] introduce a general model for deterministic CRAs with a population of n stations and conflicts of multiplicity k and establish that for all k and n ($2 \le k \le n$), $\Omega(k(\log n)/(\log k))$ time must elapse in the worst case before all k transmissions succeed. Finally, Tsybakov [Tsyba89] has developed a technique that enables construction of equivalent non-randomized algorithms (i.e., with equal capacity, delay, etc.), from any randomized algorithm with Poisson input.

6.2. Systems with Early or Late Feedback

A different class of problems results when the feedback is available not at the end of each slot as is the standard assumption, but at some other point in time. There are two cases. First, early feedback systems, where, usually because of the limited span of the network, carrier sensing and/or collision detection techniques can provide feedback much sooner than the end of a full packet transmission. Second, delayed feedback systems, where, because of the long propagation delay (relative to the packet transmission time), the feedback is not available for many packet transmission times (slots). This last environment is usually associated with the satellite channel. Although all of the protocols can operate without modification in the case of early feedback systems, and can trivially be modified to operate under delayed feedback conditions, it is possible to obtain systems with better performance if the design takes into account these characteristics of the channels.

Tree Algorithms in the Local Area Network (LAN) environment have been suggested from early on by Massey [Masse81] and Towsley and Venkatesh [Towsl82]. Notice also, that Hayes' original paper on the subject [Hayes78] was entitled ''An Adaptive Technique for Local Distribution,'' and his model for ''Probing'' is very close to this setting (even though not a direct transmission system). When carrier sensing and/or collision detection is used, the capacity of the protocols increases dramatically. Merakos and Kazakos [Merak83] and Tsybakov and Fedortsov [Tsyba86b, Tsyba87b, Tsyba89b] investigate LANs that use Stack Algorithms with carrier sensing. In [Tsyba86b] upper and lower bounds for the maximum packet transmission rate in the network are obtained for stations that transmit packets of arbitrary length and it is shown that the maximum rate tends rapidly to one as the packet length increases. In [Tsyba87b] they study packet delay for a stack algorithm and arbitrary packet length distribution and in [Tsyba89b] they consider several algorithms for LANs with channel errors.

Mean packet delay results for the STA/SWA combination with carrier sensing and/or collision detection are given in [Polyz87c, Polyz94]. Merakos and Exley [Merak92] present a stack algorithm for CSMA and CSMA/CD channels, and then utilize the regenerative delay analysis method to obtain tight upper and lower bounds on the mean packet delay and an estimate of the delay distribution.

With satellite channels, on the other hand, the question is not about capacity (for which nothing better than that of the standard model can be expected), but how to avoid the very large delays and/or insertion of ''rest periods'' that would be induced by the usual assumption of waiting for the feedback from one algorithmic step before continuing with the next. As a result, a single copy of a conventional CRA would only make use of one slot out of every R, where $R \gg 1$ is the round-trip feedback delay. Note that the original work of Capetanakis was motivated by the application to satellite channels and his parallel tree traversal was designed to reduce t_2 by visiting pairs of nodes in parallel[1] so that a single CRA can utilize two slots out of every R. Of course, all of the ''wasted'' slots can be recovered by interleaving the executions of multiple copies of a conventional CRA, each responsible for a fraction of the total traffic. Several strategies have also been proposed for reducing the delay penalty, including breadth-first traversal of the large conflict resolution trees that result from gated access [Masse81], sharing a common ''gate'' among R independent copies of a gated access CRA to reduce the mean variability of the initial conflict multiplicity [Tsyba81], and even sharing a common stack among R copies of a window access CRA to combine the traversals of several trees of low initial conflict multiplicity [Brown87].

6.3. Channel Errors and Capture

Many authors have investigated CRA performance with various channel and feedback error models. The first treatment of channel errors is due, again, to Massey [Masse81] who has considered shared channel errors, i.e., errors in the forward channel or the feedback channel, but which all stations perceive in the same way. His model assumes that idle or successful slots can be erroneously recognized as conflicts, while conflict slots are always correctly detected. As we have seen, Paterakis and Papantoni-Kazakos have proposed and analyzed the 2CA, a better alternative than the STA in noisy environments [Pater89]. Vvedenskaya and Tsybakov [Vvede88] compute and compare packet delays for two stack algorithms for the case of a channel with feedback errors. Tsybakov and Likhanov [Tsyba89c] derive an upper bound on the capacity of the ALOHA-type channel as a function of the error probability.

---

[1] His perspective into problem of CRAs might have cost him the discovery of the MTA and other more advanced CRAs, since these require a serial transmission schedule.

Various authors, including Vvedenskaya and Tsybakov [Vvede83b] and Cidon and Sidi [Cidon87], have considered a more general model for the channel, where noise errors and erasures (i.e., single packet transmissions or conflicts that are interpreted as idle slots), are possible, and have developed algorithms that can operate in such environments. In [Cidon87] it is shown that it is possible to devise algorithms that ensure that all packets are eventually successfully transmitted, including packets that are erased.

Further work on noisy channels has considered time varying channels. The channel is assumed to be in one of two states. In each state, the channel is assumed to be a discrete memoryless channel and the transitions between the two states are assumed to be Markovian. Kessler and Sidi [Kessl89] consider the STA in a noisy channel with memory. They introduce a two-state, first-order Markovian model for the channel and obtain a stability result. Extensions to more general channel models are also discussed. Ho *et al.* [Ho90] also obtain capacity results for the STA under these assumptions. Gong and Paterakis [Gong91] analyze the 2CA in such an environment and find that it outperforms the STA.

Finally, Suda *et al.* [Suda90] and Kurose *et al.* [Kuros90] consider the difficult problem of unshared errors, i.e., systems where stations interpret the channel feedback differently. In [Suda90] the authors show that the STA is robust in unshared error environments. They also quantify, through simulation and analysis, the performance degradation caused by unshared errors. In [Kuros90] Stack Algorithms are examined. It is found that those algorithms which tend to treat the receipt of corrupted feedback by a station as a collision show superior performance for throughput values greater than approximately 0.2, whereas, at low throughput values, there is relatively little difference in the performance among the various approaches studied.

Cidon and Sidi [Sidi85, Cidon85] consider environments with capture, i.e., systems where while more than one packet are transmitted simultaneously, stations can still receive one of the packets successfully. In [Cidon85] they suggest extensions to the algorithms for the two situations in which the receiver can and cannot distinguish between success slots and capture slots. In particular, the authors present a class of retransmission schemes for packets that have been transmitted during capture slots but have not been received correctly. Garg and Mohan [Garg87] design a CRA for a ternary feedback channel with capture and a finite population of users split into two groups with different transmission powers (assuming a Bernoulli model for packet generation). They prove that the protocol is always superior to TDMA as long as there is at least one user in the high-power group. Cidon *et al.* [Cidon88b] introduce CRAs that handle erasures as well as captures and argue that in practice, the positive effect of captures compensates the negative effect of erasures. An approach that effectively utilizes the capture phenomena is introduced. This approach incorporates a random power-level-selection scheme that allows each node to choose randomly to transmit in one of several allowable levels of power.

An environment with capture is also considered in [Lyons89]. The authors propose and analyze a modification of the 2CA assuming ternary feedback, and in the presence of capture, identification of the captured packet by all the users in the system. Mehravari [Mehra90] presents CRAs and determines lower bounds to the channel capacity for many different feedback models. Finally, Schmid [Schmi92] investigates some underlying characteristic parameters of CRA trees for channels with capture.

Stavrakakis and Kazakos [Stavr86] consider a frequency-hopping Code Division Multiple Access (CDMA) system with orthogonal codes and frequency-hopping pattern sensing that provides ternary feedback to the users. They adopt a Stack Algorithm for conflict resolution and investigate system performance in terms of throughput and average packet delay analytically and by simulation. Georgiopoulos and Papantoni-Kazakos [Georg86] propose and analyze a limited sensing CRA for a similar CDMA system and compute throughputs and expected packet delays; in the presence of interference between transmissions, they compute throughputs subject to an upper bound on the probability of erroneous decoding. Hu and Chang [Hu90] study the STA/WA and the FCFS 0.487 Algorithm in a CDMA environment that leads to captures.

6.4. Priorities and Delay Bounded Communication

Panwar *et al.* [Panwa87] consider CRAs for the ternary feedback channel with time constraints. Their objective is to maximize the number of packets that are successfully transmitted within a fixed deadline after their arrival for transmission. Kurose *et al.* [Kuros88] examine the use of window CRAs for supporting time-constrained communication. They formulate a policy for controlling protocol operation to minimize the percentage of messages with waiting times greater than some given bound and develop a performance model based on a queueing system

with impatient customers. Paterakis *et al.* [Pater89c] consider CRAs for messages with strict delay constraints. They present a version of the 2CA and compute the fraction and the expected delay for the successfully transmitted traffic.

Stavrakakis and Kazakos [Stavr91] develop and analyze a binary feedback CRA for a nonhomogeneous user population consisting of two classes with different priorities. Liu and Papantoni-Kazakos [Liu92] consider data networks carrying mixed low and high priority traffic. They propose a synchronous limited sensing CRA which gives a delay advantage to the high priority traffic. The algorithm basically consists of two dynamically coupled window algorithms, one for the high and one for the low-priority packets. Finally, Martel and Moh [Marte91] deal with messages with associated priorities or deadlines (denoted by an integer i), and propose a distributed prioritized CRA. They prove that the expected waiting time of the i-th message is $\Theta(i+\log c)$, which is optimal.

6.5. Other Models and Performance Analysis Results

Paterakis *et al.* [Pater87] consider various synchronous full sensing algorithms for finite independent and identical users in the system. They prove that for any i.i.d. arrival process per user, the algorithms are stable provided that the total input rate is less than one. However, as the population size increases, the stability of an algorithm in the class is determined by its throughput in the presence of the infinite population model for all practical purposes. Polyzos *et al.* [Polyz87] provide delay distribution results for various finite non-homogeneous population models. Kurtz and Sidi [Kurtz88] apply group testing techniques to the design of efficient algorithms for systems with a heterogeneous population of users considering both finite and infinite population systems. Polyzos [Polyz89] has considered a more general arrivals model for window access CRAs that allows delay distribution results to be obtained for any arrival process with i.i.d. number of arrivals per window.

Tsybakov and Fedortsov [Tsyba86] derive an upper bound on packet delay for an unmodified treelike CRA which is linear for low intensities of the incoming packet stream. Merakos and Bisdikian [Merak88] present delay analysis of a d-ary Stack Algorithm, deriving tight upper and lower bounds on the mean and variance of the packet delay.

An important departure from the standard model, which assumes synchronous transmissions, has been proposed by Molle in [Molle83] and also by Georgiopoulos *et al.* in [Georg85] and [Georg86b] by the introduction of an asynchronous implementation of CRAs, which is particularly relevant to the LAN environment. The idea here is to simplify the implementation by allowing each user to change state based on *local* observations of the channel rather than requiring all users to agree on the outcome from one ‘‘step’’ before proceeding to the next one. Not only does this eliminate the need for all users to access a common global clock, it may even lead to slight increases in efficiency in network topologies where all inter-station distances are not equal to the worst case [Molle87].

References

[Aldou87] D. J. Aldous, ‘‘Ultimate Instability of Exponential Back-Off Protocol for Acknowledgement-Based Transmission Control of Random Access Communication Channels,’’ *IEEE Transactions on Information Theory* **33**(2), pp. 219-223 (March 1987).

[Berge83] T. Berger and N. Mehravari, ‘‘Conflict Resolution Protocols for Secure Multiple-Access Communication Systems,’’ in *Secure Digital Communications*, ed. G. Longo, Springer-Verlag, Udine, Italy (1983).

[Berge84] T. Berger, N. Mehravari, D. Towsley, and J. Wolf, ‘‘Random Multiple-Access Communication and Group Testing,’’ *IEEE Transactions on Communications* **32**(7), pp. 769-779 (July 1984).

[Berts92] D. Bertsekas and R. Gallager, *Data Networks, 2nd ed.*, Prentice-Hall, Englewood Cliffs, NJ (1992).

[Brown87] M. Brownlie, ‘‘On the Application of Tree Conflict Resolution Algorithms to Satellite Channels, or What to Do Until the Feedback Arrives?,’’ Technical Note CSRI-45, CSRI, University of Toronto, Toronto, Canada (May 1987).

[Capet78] J. I. Capetanakis, ‘‘The Multiple Access Broadcast Channel: Protocol and Capacity Considerations,’’ ESL-R-806, Electronic Systems Laboratory, MIT, Cambridge, Mass. (March 1978). (Also, Ph.D. Dissertation, Dept. of Electrical Engineering, August 1977.).

[Capet79] J. I. Capetanakis, ‘‘Generalized TDMA: The Multi-Accessing Tree Protocol,’’ *IEEE Transactions on Communications* **27**(10), pp. 1476-1484 (October 1979).

[Capet79b] J. I. Capetanakis, ‘‘Tree Algorithms for Packet Broadcast Channels,’’ *IEEE Transactions on Information Theory* **25**(5), pp. 505-515 (September 1979).

[Cidon85] I. Cidon and M. Sidi, ‘‘The Effect of Capture on Collision Resolution Algorithms,’’ *IEEE Transactions on Communications* **33**(4), pp. 317-324 (April 1985).

[Cidon87] I. Cidon and M. Sidi, ‘‘Erasures and Noise in Splitting Multiple Access Algorithms,’’ *IEEE Transactions on Information Theory* **33**(1), pp. 132-143 (January 1987).

[Cidon88b] I. Cidon, H. Kodesh, and M. Sidi, ‘‘Erasure, capture, and random power level selection in multiple-access systems,’’ *IEEE Transactions on Communications* **36**(3), pp. 263-271 (March 1988).

[Cidon88] I. Cidon and M. Sidi, ‘‘Conflict multiplicity estimation and batch resolution algorithms,’’ *IEEE Transactions on Information Theory* **34**(1), pp. 101-110 (Jan. 1988).

[Cohen69] J. W. Cohen, *The Single Server Queue*, North-Holland (1969).

[Cruz82] R. Cruz and B. Hajek, ‘‘A New Upper Bound to the Throughput of a Multi-Access Broadcast Channel,’’ *IEEE Transactions on Information Theory* **28**(3), pp. 402-405 (May 1982).

[Dorfm43] R. Dorfman, ‘‘The Detection of Defective Members of Large Populations,’’ *Ann. Math. Statist.* **14**, pp. 436-440 (December 1943).

[Fayol77] G. Fayolle, E. Gelenbe, and J. Labetoulle, ‘‘Stability and Optimal Control of the Packet Switching Broadcast Channel,’’ *Journal of the ACM* **24**(3), pp. 375-386 (July 1977).

[Fayol85] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet, ‘‘Analysis of a Stack Algorithm for Random Multiple-Access Communication,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 244-254 (March 1985).

[Fergu75] M. J. Ferguson, ‘‘On the Control, Stability, and Waiting Time in a Slotted ALOHA Random Access System,’’ *IEEE Transactions on Communications* **23**(11), pp. 1306-1311 (November 1975).

[Galla78b] R. Gallager, ‘‘Variations on a Theme of Huffman,’’ *IEEE Transactions on Information Theory* **24**(6), pp. 668-674 (November 1978).

[Galla78] R. G. Gallager, ‘‘Conflict Resolution in Random Access Broadcast Networks,’’ Proc. *of the AFOSR Workshop in Communication Theory and Applications*, pp. 74-76 (September 17-20, 1978).

[Garg87] N. K. Garg and S. Mohan, ‘‘Group testing protocol with capture for random access communication,’’ *IEEE Transactions on Communications* **35**(8), pp. 849-54 (Aug. 1987).

[Georg82] L. Georgiadis and P. Papantoni-Kazakos, ‘‘A Collision Resolution Protocol for Random Access Channels with Energy Detectors,’’ *IEEE Transactions on Communications* **30**(11), pp. 2413-2420 (November 1982).

[Georg87] L. Georgiadis and P. Papantoni-Kazakos, ‘‘A 0.487 Throughput Limited Sensing Algorithm,’’ *IEEE Transactions on Information Theory* **33**(2), pp. 233-237 (March 1987).

[Georg87b] L. Georgiadis, L. F. Merakos, and P. Papantoni-Kazakos, ‘‘A Method for the Delay Analysis of Random Multiple-Access Algorithms Whose Delay Process is Regenerative,’’ *IEEE Journal on Selected Areas in Communications* **5**(6), pp. 1051-1062 (July 1987).

[Georg84] M. Georgiopoulos, L. Merakos, and P. Papantoni-Kazakos, ‘‘Collision Resolution Protocols for Random Access Channels with Bandwidth and Energy Overhead,’’ Proc. *IEEE Globecom ’84*, pp. 3541-3545 (December 1984).

[Georg85] M. Georgiopoulos, L. Merakos, and P. Papantoni-Kazakos, ‘‘An Asynchronous Stack Algorithm for CSMA and CSMA-CD Channels,’’ Proc. *IEEE Infocom ’85*, pp. 404-409 (March 1985).

[Georg86] M. Georgiopoulos and P. Papantoni-Kazakos, ‘‘Slotted random access spread spectrum (frequency-hopped) packet radio networks,’’ Proc. *IEEE GLOBECOM ’86*, pp. 1734-1739 (1-4 Dec. 1986).

[Georg86b] M. Georgiopoulos, L. Merakos, and P. Papantoni-Kazakos, ‘‘High Performance Asynchronous Limited Sensing Algorithms for CSMA and CSMA-CD Channels,’’ pp. 185-214 in *Local Area and Multiple Access Networks*, ed. R. L. Pickholtz, Computer Science Press (1986).

[Gong91] Y. Gong and M. Paterakis, ‘‘A random multiple-access algorithm for the dependent feedback error channel,’’ Proc. *IEEE INFOCOM ’91*, pp. 620-627 (April 1991).

[Green85] A. G. Greenberg and S. Winograd, ‘‘A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels,’’ *Journal of the Association for Computing Machinery* **32**(3), pp. 589-596 (July 1985).

[Green87] A. G. Greenberg, P. Flajolet, and R. E. Ladner, ‘‘Estimating the multiplicities of conflicts to speed their resolution in multiple access channels,’’ *Journal of the Association for Computing Machinery* **34**(2), pp. 289-325 (April 1987).

[Hayes78] J. F. Hayes, ‘‘An Adaptive Technique for Local Distribution,’’ *IEEE Transactions on Communications* **26**(8), pp. 1178-1186 (August 1978).

[Ho90] K. K. Y. Ho, R. R. Rao, and J. K. Wolf, ‘‘Random-access systems with a time varying channel,’’ *IEEE Transactions on Communications* **38**(9), pp. 1293-1297 (Sept. 1990).

[Hu90] M.-C. Hu and J.-F. Chang, ‘‘Collision resolution algorithms for CDMA systems,’’ *IEEE Journal on Selected Areas in Communications* **8**(4), pp. 542-554 (May 1990).

[Huang85] J. Huang and T. Berger, ‘‘Delay Analysis of Interval-Searching Contention Resolution Algorithms,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 264-273 (March 1985).

[Huang86] J.-C. Huang and T. Berger, ‘‘Delay Analysis of 0.487 Contention Resolution Algorithms,’’ *IEEE Transactions on Communications* **34**(9), pp. 916-926 (September 1986).

[Humbl86] P. A. Humblet, ‘‘On the Throughput of Channel Access Algorithms with Limited Sensing,’’ *IEEE Transactions on Communications* **34**(4), pp. 345-347 (April 1986).

[Kessl89] I. Kessler and M. Sidi, ‘‘Splitting Algorithms in Noisy Channels with Memory,’’ *IEEE Transactions on Information Theory* **35**(5), pp. 1034-1043 (September 1989).

[Khasi89] L. S. Khasin, ‘‘Conflict resolution in a multiple access channel,’’ *Problemy Peredachi Informatsii* **25,**(4), pp. 63-68 (308-312) (Oct.-Dec. 1989).

[Kingm62] J. F. C. Kingman, ‘‘The Effect of Queue Discipline on Waiting Time Variance,’’ *Proceedings of the Cambridge Philosophical Society* **58**, pp. 163-164 (1962).

[Klein76] L. Kleinrock, *Queueing Systems, Vol. II, Computer Applications*, Wiley-Interscience, New York, NY (1976).

[Knuth68] D. E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Reading, MA (1968).

[Komlo85] J. Komlos and A. G. Greenberg, ‘‘An Asymptotically Fast Nonadaptive Algorithm for Conflict Resolution in Multiple-Access Channels,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 302-306 (March 1985).

[Kuros88] J. F. Kurose, M. Schwartz, and Y. Yemini, ‘‘Controlling Window Protocols for Time-Constrained Communication in Multiple Access Networks,’’ *IEEE Transactions on Communications* **36**(1), pp. 41-9 (January 1988).

[Kuros90] J. F. Kurose, A. Shrivastava, and D. Towsley, ‘‘Stack Algorithms for Random Multiple-Access Networks in the Presence of Asymmetric Feedback,’’ *IEEE Transactions on Communications* **38**(9), pp. 1308-1313 (September 1990).

[Kurtz88] D. Kurtz and M. Sidi, ‘‘Multiple access algorithms via group testing for heterogeneous population of users,’’ *IEEE Transactions on Communications* **36**(12), pp. 1316-1323 (Dec. 1988).

[Liu92] M. Liu and P. Papantoni-Kazakos, ‘‘A random-access algorithm for data networks carrying high-priority traffic,’’ *IEEE Transactions on Communications* **40**(1), pp. 84-96 (Jan. 1992).

[Liu87] Y.-C. Liu and G. L. Wise, ‘‘Performance of a CSMA/CD Protocol for Local Area Networks,’’ *IEEE Journal on Selected Areas in Communications* **5**(6), pp. 948-955 (July 1987).

[Lyons89] D. F. Lyons and P. Papantoni-Kazakos, ‘‘A window random access algorithm for environments with capture,’’ *IEEE Transactions on Communications* **37**(7), pp. 766-770 (July 1989).

[Marte91] C. U. Martel and W.-H. L. M. Moh, ‘‘Optimal prioritized conflict resolution on a multiple access channel,’’ *IEEE Transactions on Computers* **40**(10), pp. 1102-1108 (Oct. 1991).

[Masse81] J. L. Massey, ‘‘Collision-Resolution Algorithms and Random-Access Communications,’’ in *Multi-User Communications*, ed. G. Longo, Springer-Verlag, New York (1981).

[Masse85] J. L. Massey and P. Mathys, ‘‘The Collision Channel Without Feedback,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 192-204 (March 1985).

[Mathy84] P. Mathys, ‘‘Analysis of Random-Access Algorithms,’’ Doctoral Dissertation (ETH No. 7713), Swiss Federal Institute of Technology, Zurich, Switzerland (1984).

[Mathy85] P. Mathys and P. Flajolet, ‘‘Q-ary Collision Resolution Algorithms in Random-Access Systems with Free or Blocked Channel Access,’’ *IEEE Transactions on Information Theory* **31**(2) (March 1985).

[Mehra88] N. Mehravari, ‘‘On the Poisson contention resolution problem with feedback based on conflict intensity,’’ *IEEE Transactions on Communications* **36**(4), pp. 513-516 (April 1988).

[Mehra90] N. Mehravari, ‘‘Random-access communication with multiple reception,’’ *IEEE Transactions on Information Theory* **36**(3), pp. 614-22 (May 1990).

[Merak83] L. Merakos and D. Kazakos, ‘‘Stack Algorithms for Local Area Networks,’’ Proc. *IEEE Globecom ’83*, pp. 3311-3313 (November 1983).

[Merak88] L. Merakos and C. Bisdikian, ‘‘Delay Analysis of the n-ary Stack Random-Access Algorithm,’’ *IEEE Transactions on Information Theory* **34**(5), pp. 931-942 (September 1988).

[Merak92] L. F. Merakos and G. M. Exley, ‘‘Performance Analysis of a Stack Random Access Algorithm for CSMA and CSMA/CD Channels,’’ *IEEE Transactions on Communications* **40**(6), pp. 1047-1058 (June 1992).

[Mikha81] V. A. Mikhailov and B. S. Tsybakov, ‘‘An Upper Bound to Capacity of Random Multiple Access Systems,’’ *Problemy Peredachi Informatsii* **17**(1), pp. 90-95 (Jan.-March, 1981).

[Molle81] M. L. Molle, ‘‘Unifications and Extensions and Extensions of the Multiple Access Communications Problem,’’ CSD Report No. 810730 (UCLA-ENG-8118), UCLA Computer Science Department (July 1981). Ph.D. Dissertation.

[Molle82] M. L. Molle, ‘‘On the Capacity of Infinite Population Multiple Access Protocols,’’ *IEEE Transactions on Information Theory* **28**(3), pp. 396-401 (May 1982).

[Molle83] M. L. Molle, ‘‘Asynchronous Multiple Access Tree Algorithms,’’ Proc. *ACM Symposium on Communications Architectures and Protocols (ACM SIGCOMM ’83)* (March 1983).

[Molle87] M. L. Molle, K. Sohraby, and A. N. Venetsanopoulos, ‘‘Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks,’’ *IEEE Journal on Selected Areas in Communications* **5**(6), pp. 956-968 (July 1987).

[Molle89] M. L. Molle, ‘‘Analysis of a Class of Distributed Queues with Applications,’’ *Performance Evaluation* **9**(4), pp. 271-286 (August 1989).

[Molle92] M. L. Molle and A. C. Shih, ‘‘Computation of the Packet Delay in Massey’s Standard and Modified Tree Conflict Resolution Algorithms with Gated Access,’’ Technical Report CSRI-264, Computer Systems Research Institute, University of Toronto, Toronto, Canada (February 1992).

[Mollo85] M. K. Molloy, ‘‘Collision Resolution on the CSMA/CD Bus,’’ *Computer Networks and ISDN Systems* **9**(3), pp. 209-214 (March 1985).

[Mosel85] J. Mosely and P. A. Humblet, ‘‘A Class of Efficient Contention Resolution Algorithms for Multiple Access,’’ *IEEE Transactions on Communications* **33**(2), pp. 145-151 (February 1985).

[Pakes69] A. G. Pakes, ‘‘Some Conditions for Ergodicity and Recurrence of Markov Chains,’’ *Operations Research* **17**, pp. 1058-1061 (1969).

[Panwa85] S. S. Panwar, D. Towsley, and J. K. Wolf, ‘‘On the Throughput of Degenerate Intersection and First-Come First-Served Collision Resolution Algorithms,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 274-279 (March 1985).

[Panwa87] S. S. Panwar, D. Towsley, J. K. Wolf, and Y. Armoni, ‘‘Collision resolution algorithms for a time-constrained multiaccess channel,’’ *Proceedings of the Twenty-Fifth Annual Allerton Conference on Communication, Control, and Computing*, pp. 1081-8 (30 Sept.-2 Oct. 1987).

[Pater87] M. Paterakis, L. Georgiadis, and P. Papantoni-Kazakos, ‘‘On the Relation Between the Finite and the Infinite Population Models for a Class of RAA’s,’’ *IEEE Transactions on Communications* **35**(11), pp. 1239-1240 (November 1987).

[Pater89c] M. Paterakis, L. Georgiadis, and P. Papantoni-Kazakos, ‘‘A Full Sensing Window Random-Access Algorithm for Messages with Strict Delay Constraints,’’ *Algorithmica* **4**(3), pp. 313-328 (1989).

[Pater89] M. Paterakis and P. Papantoni-Kazakos, ‘‘A Simple Window Random Access Algorithm with Advantageous Properties,’’ *IEEE Transactions on Information Theory* **35**(5), pp. 1124-1130 (September 1989).

[Pater89b] M. Paterakis and P. Papantoni-Kazakos, ‘‘A Limited Sensing Random-Access Algorithm with Binary Success-Failure Feedback,’’ *IEEE Transactions on Communications* **37**(5), pp. 526-530 (May 1989).

[Pippe81] N. Pippenger, ‘‘Bounds on the Performance of Protocols for a Multiple-Access Broadcast Channel,’’ *IEEE Transactions on Information Theory* **27**(2), pp. 145-151 (March 1981).

[Polyz84] G. C. Polyzos, ‘‘Tree Conflict Resolution Algorithms: The Non-Homogeneous Case,’’ M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto, Toronto, Ontario, Canada (December 1984).

[Polyz87] G. C. Polyzos, M. L. Molle, and A. N. Venetsanopoulos, ‘‘Performance Analysis of Finite Nonhomogeneous Population Tree Conflict Resolution Algorithms Using Constant Size Window Access,’’ *IEEE Transactions on Communications* **35**(11), pp. 1124-1138 (November 1987).

[Polyz87b] G. C. Polyzos and M. L. Molle, ‘‘A Generalized Busy Period Approach to the Delay Analysis of Window Access Tree Conflict Resolution Algorithms,’’ Proc. *IEEE ICC ’87* (June 1987).

[Polyz87c] G. C. Polyzos and M. L. Molle, ‘‘Delay Analysis of a Window Tree Conflict Resolution Algorithm in a Local Area Network Environment,’’ Proc. *ACM SIGMETRICS ’87* (May 1987).

[Polyz89] G. C. Polyzos, ‘‘A Queueing Theoretic Approach to the Delay Analysis for a Class of Conflict Resolution Algorithms,’’ Technical Report CSRI-224, Computer Systems Research Institute, University of Toronto, Toronto, Canada (March 1989). (Ph.D. Dissertation.).

[Polyz93] G. C. Polyzos and M. L. Molle, ‘‘A Queueing Theoretic Approach to the Delay Analysis for the FCFS 0.487 Conflict Resolution Algorithm,’’ *IEEE Transactions on Information Theory* **39**(11) (November 1993).

[Polyz94] G. C. Polyzos and M. L. Molle, ‘‘A Queueing Theoretic Methodology for the Performance Analysis of Separable Window Access Conflict Resolution Algorithms with Variable Length Elementary Events,’’ *Queueing Systems: Theory and Applications (QUESTA)* **15** (1994).

[Reupp83] R. A. Reuppel, ‘‘A Unification of All Existing Upper Bounds for the Random-Access Problem,’’ *1983 International Symposium on Information Theory*, p. 31 (September 1983).

[Rom90] R. Rom and M. Sidi, *Multiple Access Protocols*, Springer-Verlag, New York, NY (1990).

[Ruget81] G. Ruget, ‘‘Some Tools for the Study of Channel-Sharing Algorithms,’’ in *Multi-User Communications*, ed. G. Longo, Springer-Verlag, Udine, Italy (1981).

[Ryter80] D. Ryter, ‘‘A Conflict Resolution Algorithm for Noisy Multiaccess Channels,’’ TH-1007, Laboratory for Information and Decision Systems, MIT, Cambridge, MA (May 1980).

[Schmi92] U. Schmid, ‘‘On a tree collision resolution algorithm in the presence of capture,’’ *Informatique Theorique et Applications* **26**(2), pp. 163-97 (1992).

[Servi86] L. D. Servi, ‘‘D/G/1 Queues with Vacations,’’ *Operations Research* **34**(4), pp. 619-629 (July-August 1986).

[Sidi85] M. Sidi and I. Cidon, ‘‘Splitting Protocols in the Presence of Capture,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 295-301 (March 1985).

[Sobel59] M. Sobel and P. A. Groll, ‘‘Group Testing to Eliminate Efficiently All Defectives in a Binomial Sample,’’ *Bell Syst. Tech. Journal* **38**, pp. 1178-1252 (September 1959).

[Stavr86] I. Stavrakakis and D. Kazakos, ‘‘A simple stack algorithm for a code division multiple access communication system,’’ *Proceedings of the 25th IEEE Conference on Decision and Control*, pp. 2085-9 (10-12 Dec. 1986).

[Stavr91] I. Stavrakakis and D. Kazakos, ‘‘A multiuser random-access communication system for users with different priorities,’’ *IEEE Transactions on Communications* **39**(11), pp. 1538-1541 (Nov. 1991).

[Suda90] T. Suda, J. Jungok Bae, and D. C. Baxter, ‘‘The Robustness and Performance of Tree Collision Resolution Algorithms in an Unshared Feedback Error Environment,’’ *Computer Networks and ISDN Systems* **18**(4), pp. 275-292 (May 1990).

[Towsl82] D. Towsley and G. Venkatesh, ‘‘Window Random Access Protocols for Local Computer Networks,’’ *IEEE Transactions on Computers* **31**(8), pp. 715-722 (August 1982).

[Tsyba78] B. S. Tsybakov and V. A. Mikhailov, ‘‘Free Synchronous Packet Access in a Broadcast Channel with Feedback,’’ *Problemy Peredachi Informatsii* **14**(4), pp. 32-59 (259-280) (Oct.-Dec. 1978).

[Tsyba80] B. S. Tsybakov and V. A. Mikhailov, ‘‘Random Multiple Packet Access: Part-and-Try Algorithm,’’ *Problemy Peredachi Informatsii* **16**(4), pp. 65-79 (Oct.-Dec. 1980).

[Tsyba80b] B. S. Tsybakov and N. D. Vvedenskaya, ‘‘Random Multiple-Access Stack Algorithm,’’ *Problemy Peredachi Informatsii* **16**(3), pp.80-94 (230-243) (July-September 1980).

[Tsyba80c] B. S. Tsybakov, ‘‘Resolution of a Conflict of Known Multiplicity,’’ *Problemy Peredachi Informatsii* **16**, pp. 69-82 (April-June 1980).

[Tsyba81] B. S. Tsybakov, V. A. Mikhailov, and S. P. Fedortsov, ‘‘Consideration of Packet Propagation Delay in Random Multiple Access Channel,’’ *Problemy Peredachi Informatsii* **17**, pp. 75-78 (Oct.-Dec. 1981).

[Tsyba83] B. S. Tsybakov and N. B. Likhanov, ‘‘Packet Switching in a Channel Without Feedback,’’ *Problemy Peredachi Informatsii* **19**(2), pp. 69-84 (April-June 1983).

[Tsyba85] B. S. Tsybakov and N. B. Likhanov, ‘‘Some New Random Multiple-Access Algorithms,’’ *Problemy Peredachi Informatsii* **21**(2), pp. 69-89 (April-June, 1985). (Also in Proc. *International Symposium on Information Theory*, Brighton, UK, p. 102, June 24-28, 1985.).

[Tsyba85b] B. S. Tsybakov, ‘‘Survey of USSR Contributions to Random Multiple-Access Communications,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 143-165 (March 1985).

[Tsyba86] B. S. Tsybakov and S. P. Fedortsov, ‘‘Packet transmission using a blocked unmodified random multiple access stack-algorithm,’’ *Problemy Peredachi Informatsii* **22**(3), pp. 96-102 (239-45) (July-Sept. 1986).

[Tsyba86b] B. S. Tsybakov and S. P. Fedortsov, ‘‘Local-Area Network with Random-Multiple-Access Stack Algorithm,’’ *Problemy Peredachi Informatsii* **22**(2), pp. 117-125 (April-June, 1986).

[Tsyba87] B. S. Tsybakov and N. B. Likhanov, ‘‘Upper bound on the capacity of a random multiple-access system,’’ *Problemy Peredachi Informatsii* **23**(3), pp.64-78 (224-236) (July-Sept. 1987).

[Tsyba87b] B. S. Tsybakov and S. P. Fedortsov, ‘‘Characteristics of a RMA Stack Algorithm and its Use in a Local Network,’’ *Problemy Peredachi Informatsii* **23**(4), pp. 87-101 (Oct.-Dec. 1987).

[Tsyba89] B. S. Tsybakov, ‘‘Randomized and Nonrandomized Algorithms of Random Multiple Access,’’ *Problemy Peredachi Informatsii* **25**(1), pp. 88-99 (67-76) (Jan.-March 1989).

[Tsyba89b] B. S. Tsybakov and S. P. Fedortsov, ‘‘Stack Algorithm in a Local Area Network with Errors in the Channel,’’ *Problemy Peredachi Informatsii* **25**(3), pp. 76-89 (July-Sept. 1989).

[Tsyba89c] B. S. Tsybakov and N. B. Likhanov, ‘‘Upper bound on the capacity of a packet random multiple access system with errors,’’ *Problemy Peredachi Informatsii* **25**(4), pp. 50-62 (297-308) (Oct.-Dec. 1989).

[Tsyba90] B. S. Tsybakov and A. N. Beloyarov, ‘‘Random Multiple Access in a Channel with Binary ’Success-No Success’ Feedback,’’ *Problemy Peredachi Informatsii* **26**(3), pp. 67-82 (245-260) (July-Sept. 1990).

[Tsyba90b] B. S. Tsybakov and A. N. Beloyarov, ‘‘Random Multiple Access in Binary Feedback Channel,’’ *Problemy Peredachi Informatsii* **26**(4), pp. 83-98 (354-366) (Oct.- Dec. 1990).

[Tsyba92] B. S. Tsybakov and A. Y. Privalov, ‘‘The throughput of the stack algorithm in a channel with N-conflicts,’’ *Problemy Peredachi Informatsii* **28**(2), pp.78-85 (168-174) (April-June 1992).

[Ungar60] P. Ungar, ‘‘The Cutoff Point for Group Testing,’’ *Communications on Pure and Applied Mathematics* **13**, pp. 49-54 (1960).

[Vvede83] N. D. Vvedenskaya and M. S. Pinsker, ‘‘Nonoptimality of the Part-and-Try Algorithm,’’ pp. 141-148 in *Abstracts of Papers, Int. Workshop on Convolutional Codes and Multiuser Communications*, Sochi, USSR (1983).

[Vvede83b] N. D. Vvedenskaya and B. S. Tsybakov, ‘‘Random Multiple Access of Packets to a Channel with Errors,’’ *Problemy Peredachi Informatsii* **19**(2), pp. 69-84 (Apr.-June, 1983).

[Vvede84] N. D. Vvedenskaya and B. S. Tsybakov, ‘‘Packet delay in the case of a multiple-access stack algorithm,’’ *Problemy Peredachi Informatsii* **20**(2), pp.77-97 (137-53) (April-June 1984).

[Vvede88] N. D. Vvedenskaya and B. S. Tsybakov, ‘‘Computing Packet Delay for Some Random Multiple Access Stack Algorithms,’’ *Problemy Peredachi Informatsii* **24**(3), pp. 94-101 (July-September 1988).

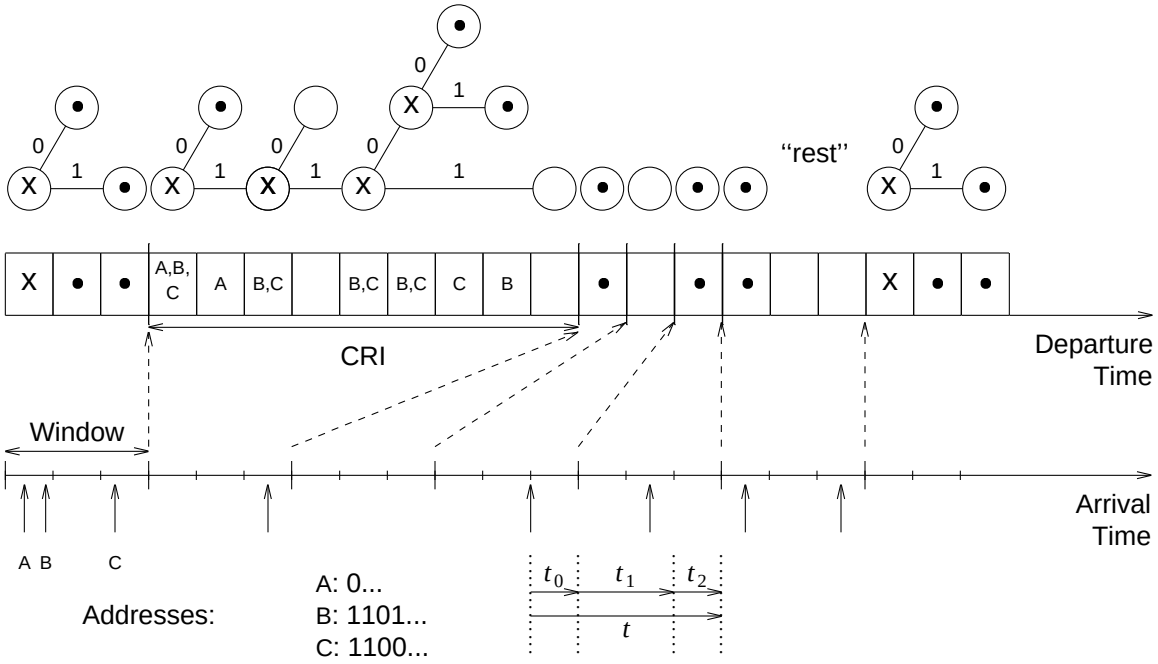
[Wolf85] J. K. Wolf, ‘‘Born Again Group Testing: Multiaccess Communications,’’ *IEEE Transactions on Information Theory* **31**(2), pp. 185-191 (March 1985).

[Woodr92] G. M. Woodruff, ‘‘A New Last-Come First-Served Preemptive Window Access Conflict Resolution Algorithm,’’ Technical Report CSRI-263, Computer Systems Research Institute, University of Toronto, Toronto, Canada (March 1992).

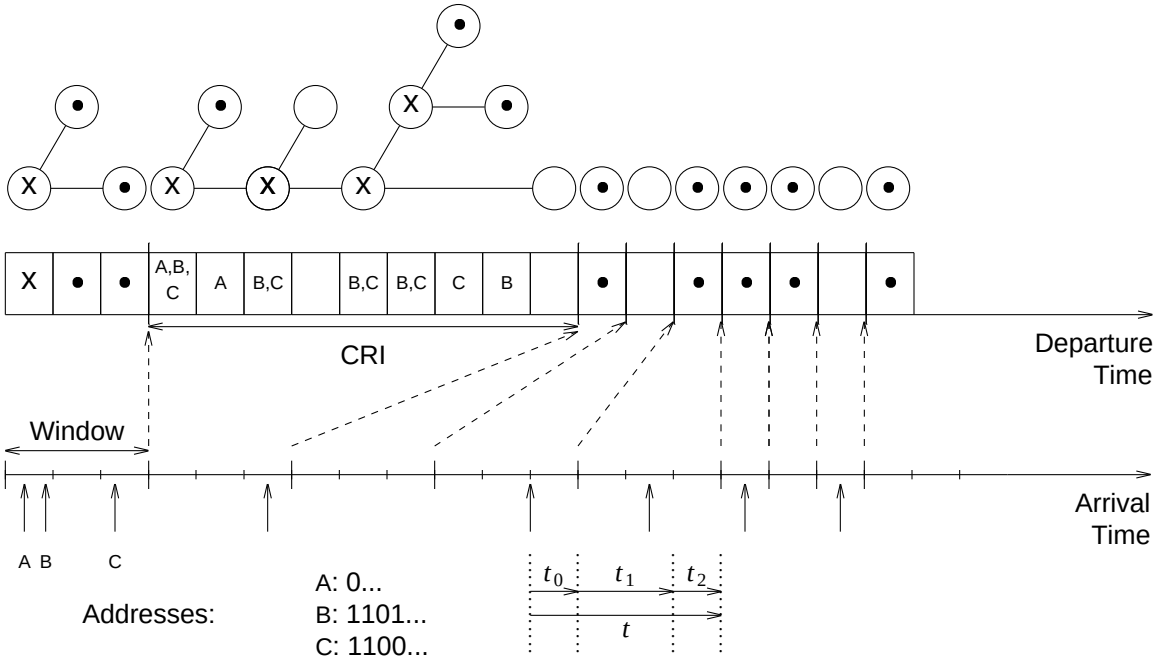
[Zhang85] Z. Zhang and T. Berger, ‘‘Improved upper bound for the capacity of the random-access channel,’’ *Problemy Peredachi Informatsii* **21**(4), pp.83-87 (Oct.-Dec. 1985).

Table of Contents

page



(a) The Standard Tree Algorithm (STA) with the Simplified Window Algorithm (SWA) for Channel Access.



(b) The Standard Tree Algorithm (STA) with the (non-simplified) Window Algorithm (WA) for Channel Access.

Fig. 1: Example of a Random Access Protocol. Binary STA for conflict resolution and (a) SWA or (b) WA for channel access. The same input scenario is presented to both protocols for comparison.

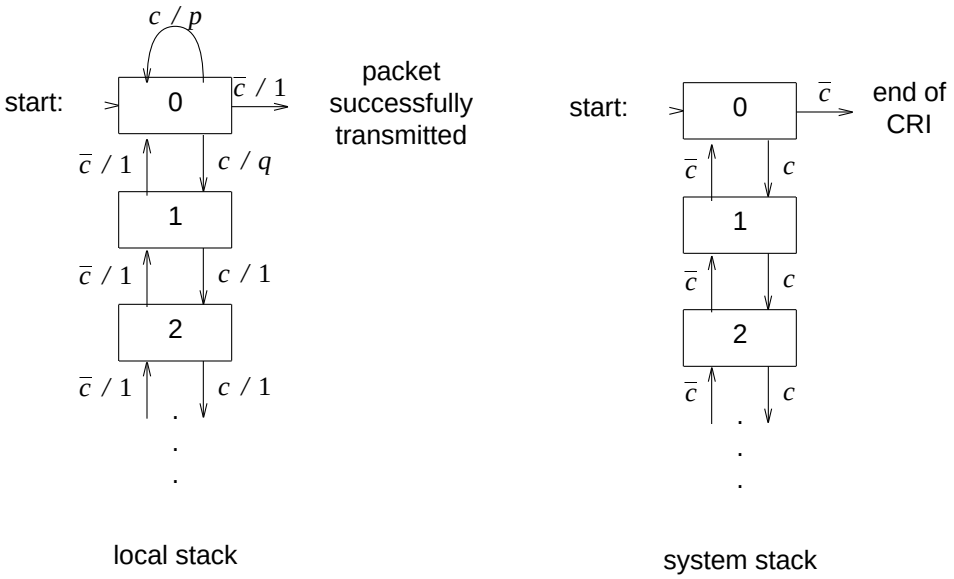


Fig. 2: The Stack Interpretation of the Standard Tree Algorithm. (Labels show feedback/probability of transition.)

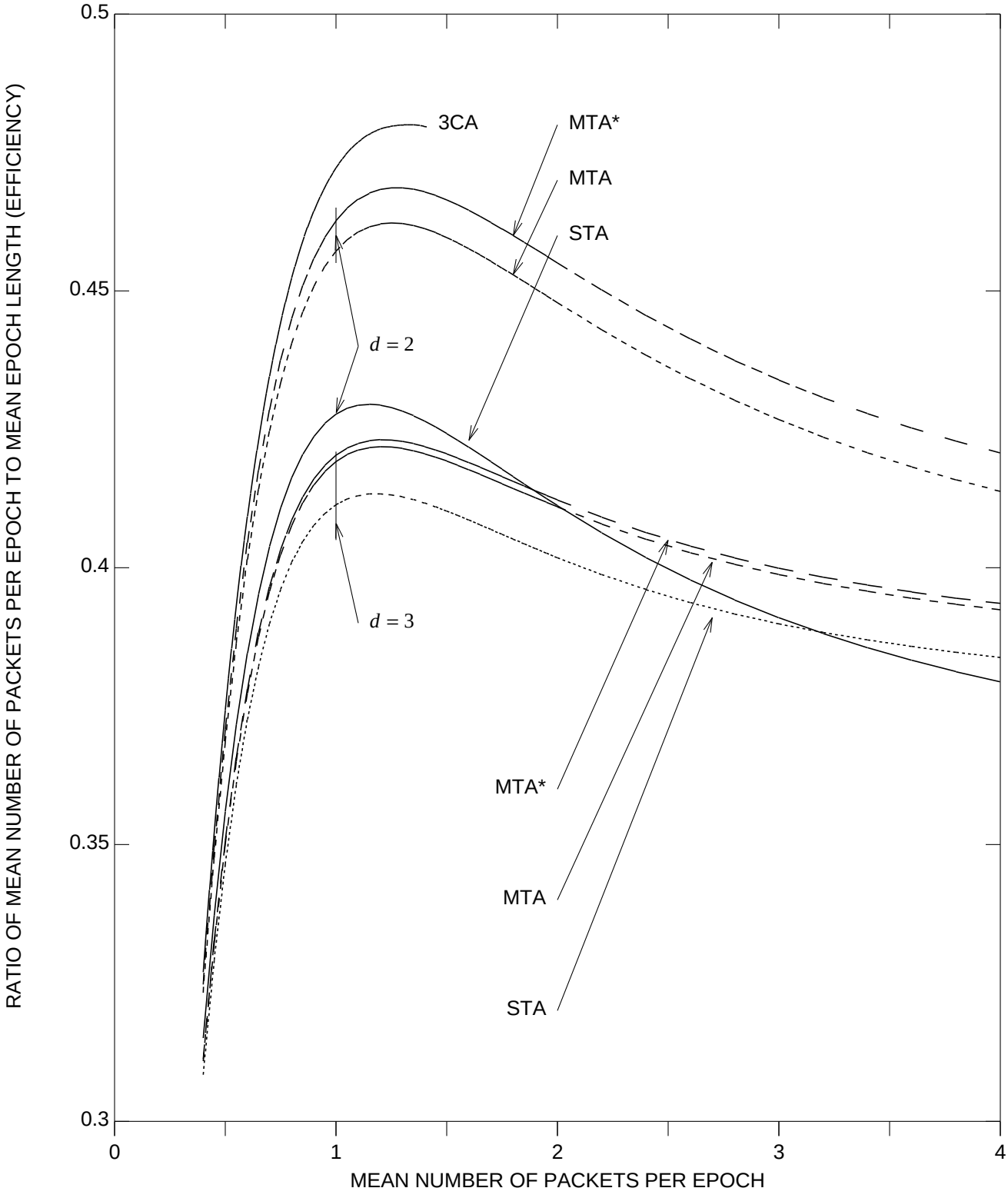


Fig. 3: Comparison of the Efficiency of Various Conflict Resolution Algorithms.

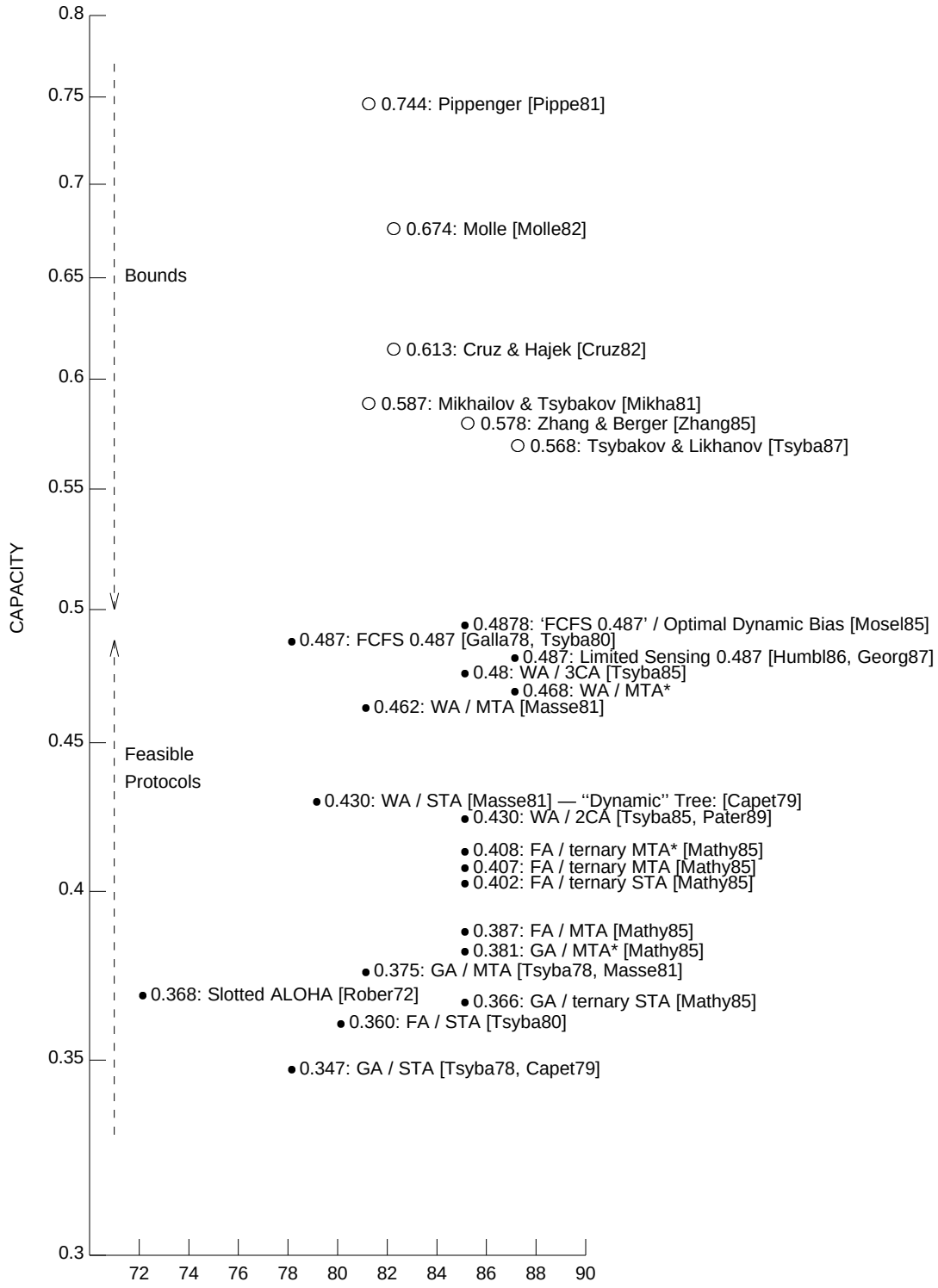


Fig. 4: Capacities of Various CRA/CAA Combinations and Upper Bounds on the Capacity of the ALOHA-Type Channel. (Dates shown are those of publication.)

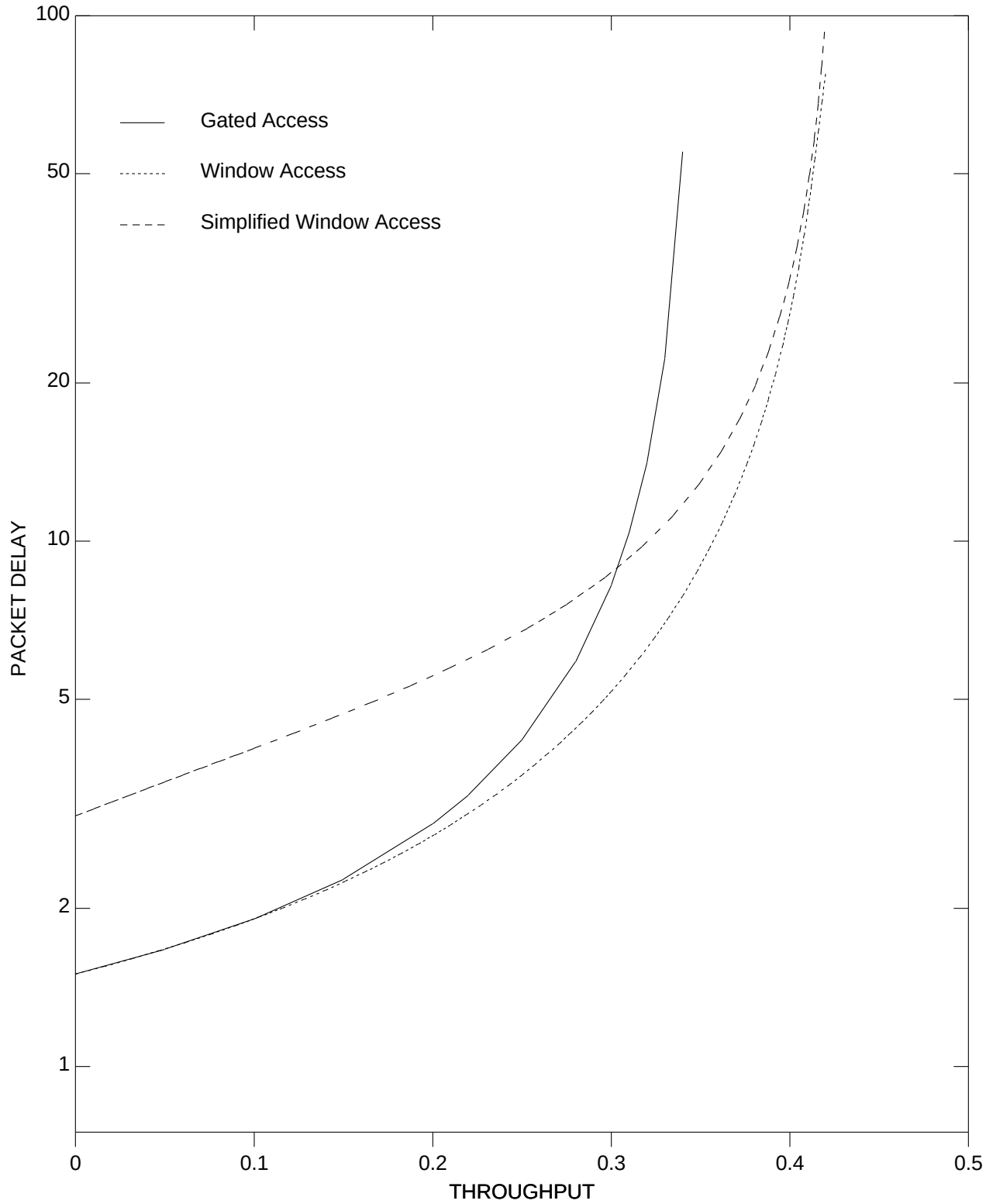


Fig. 5: Comparison of Mean Packet Delay for Various Channel Access Algorithms with the Binary Standard Tree Algorithm for Conflict Resolution ($\Delta = 3$ for WA and SWA).

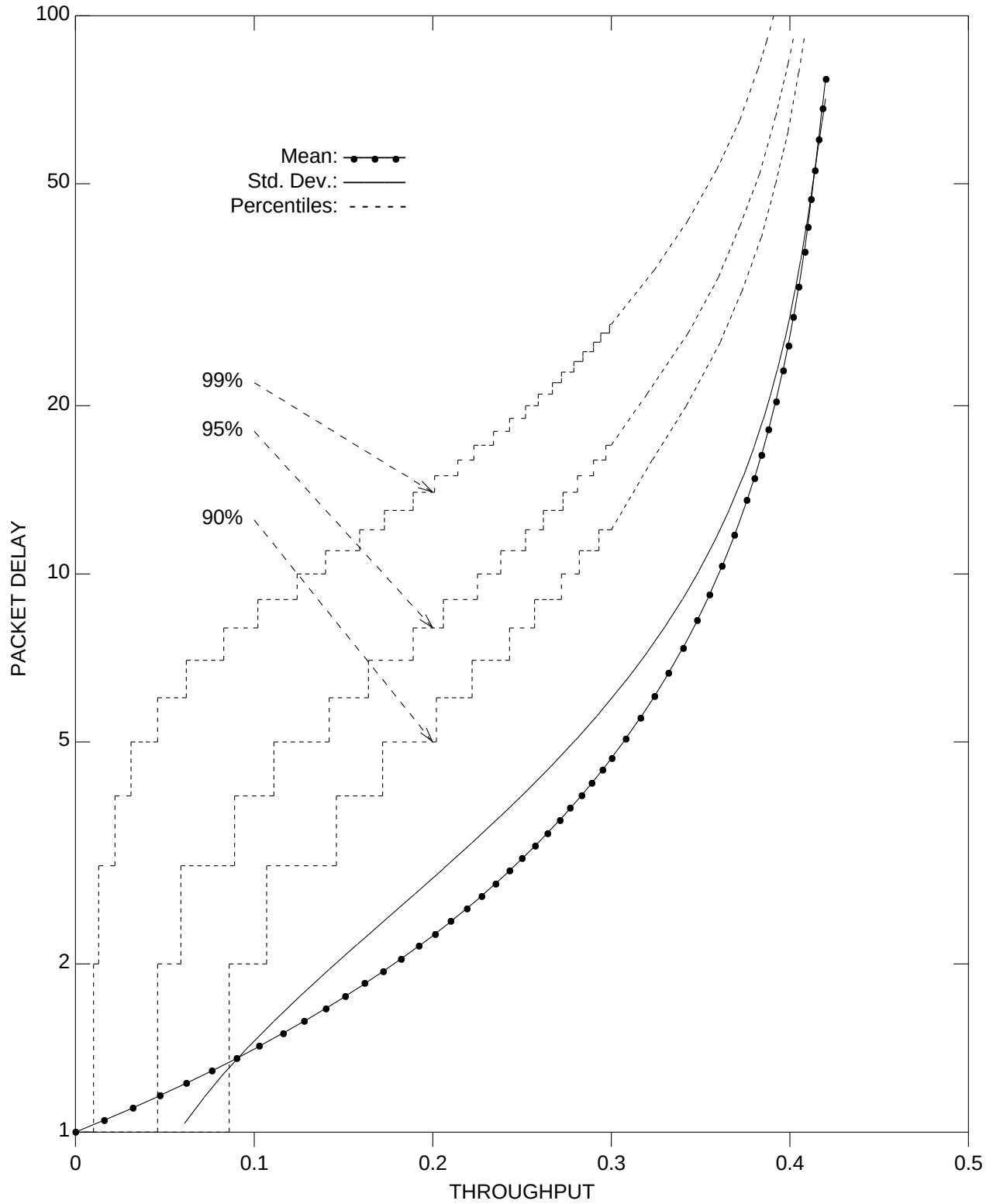


Fig. 6: Statistics of the Packet Delay for the Binary Standard Tree Algorithm (STA) with the Window Algorithm (WA) for Channel Access ($\Delta = 3$). (Random Addressing, Infinite Population, Poisson Arrivals, Discrete Time Model.)

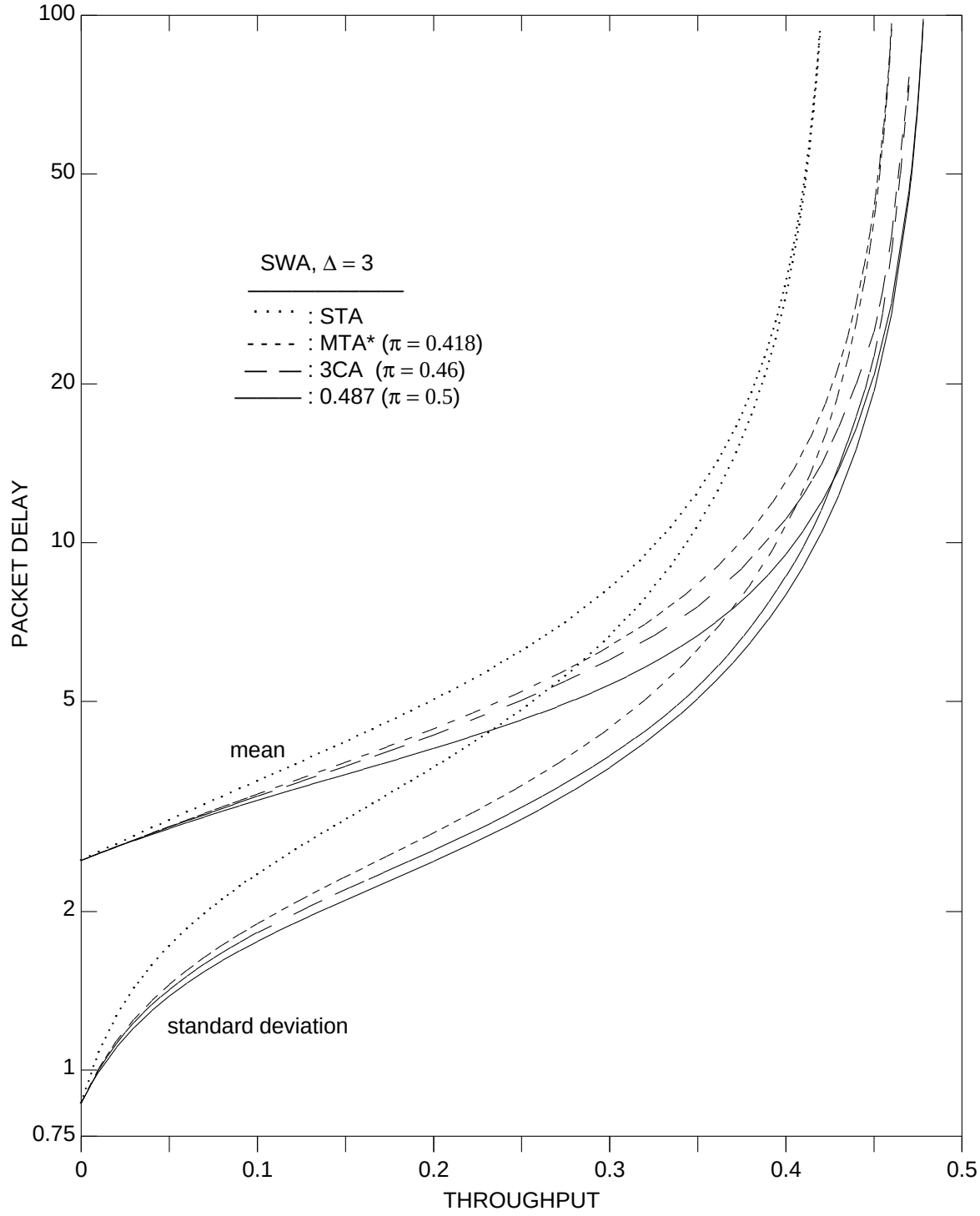


Fig. 7: Mean and Standard Deviation of Packet Delay for the STA, the Optimally Biased MTA (MTA*), the 3CA, and the FCFS 0.487 Algorithm.

Table 1: Conditional PGFs for CRI Lengths and the Delay in the Epoch of Transmission for Conflicts of Multiplicity n for the Binary STA.

n	$Q_n(z)$	$G_n(z)$
0	z	—
1	z	z
2	$-\frac{z^3}{z^2-2}$	$-\frac{z^3+z^2}{z^2+z-4}$
3	$\frac{3z^5}{z^4-6z^2+8}$	$\frac{3z^7+4z^6-7z^5-2z^4-2z^3-8z^2}{z^6+2z^5-13z^4-16z^3+54z^2+24z-64}$
4	$-\frac{15z^9-36z^7}{z^8-16z^6+84z^4-176z^2+128}$	$-\frac{15z^{11}+31z^{10}-115z^9-173z^8+\cdots+192z^3+256z^2}{z^{10}+3z^9-31z^8+\cdots-1784z^3+4640z^2+1792z-4096}$
5	$\frac{105z^9}{z^8-28z^6+228z^4-608z^2+512}$	$\frac{105z^{17}+292z^{16}-2274z^{15}-5340z^{14}+\cdots+102400z^3+65536z^2}{z^{16}+4z^{15}-70z^{14}-\cdots+1630720z^3-3723264z^2-983040z+2097152}$

Table 2: Conditional Mean, Second Moment, and Variance of CRI Length and Conditional Mean of the Delay in the Epoch of Transmission for Conflicts of Multiplicity n for the Binary STA.

n	L_n	H_n	$Var[\,l_n\,]$	$T_{2,n}$
0	1	1	0	—
1	1	1	0	1
2	5	33	8	4
3	$\frac{23}{3} \approx 7.667$	$\frac{617}{9} \approx 68.556$	$\frac{88}{9} \approx 9.778$	$\frac{17}{3} \approx 5.667$
4	$\frac{221}{21} \approx 10.524$	$\frac{54809}{441} \approx 124.283$	$\frac{5968}{441} \approx 13.533$	$\frac{22}{3} \approx 7.333$
5	$\frac{1409}{105} \approx 13.419$	$\frac{2172017}{11025} \approx 197.008$	$\frac{186736}{11025} \approx 16.938$	$\frac{941}{105} \approx 8.962$

Table 3: Conditional Mean CRI Length for the 3CA and the MTA.

n	Unbiased		Optimally Biased	
	3CA	MTA	3CA ($\pi = 0.46$)	MTA ($\pi = 0.418$)
0	1.000000	1.000000	1.000000	1.000000
1	1.000000	1.000000	1.000000	1.000000
2	4.333333	4.500000	4.311584	4.414385
3	6.547619	7.000000	6.559853	6.884642
4	9.304762	9.642857	9.247142	9.482545
5	11.991453	12.314286	11.899253	12.107792
6	14.812369	14.984793	14.650483	14.735235
7	17.686417	17.650691	17.454662	17.360549
8	20.630047	20.314014	20.315821	19.984126
9	23.630363	22.976759	23.227089	22.606864
10	26.684383	25.639897	26.184718	25.229350
11	29.787098	28.303634	29.185004	27.851860
12	32.934833	30.967819	32.224732	30.474487
13	36.124179	33.632213	35.301026	33.097238
14	39.352233	36.296623	38.411331	35.720086
15	42.616457	38.960935	41.553379	38.342996
.				
.				
.				

Table 4: Capacity of some CRAs with Window Access (Poisson arrivals). Gated Access results are equivalent to those for $\Delta = \infty$.

Δ	STA		Unbiased MTA		Biased MTA		3CA	0.487 Alg.
	d=2	d=3	d=2	d=3	d=2 $\bar{\pi}$=0.582	d=3 $\bar{\pi}$=0.381	$\bar{\pi}$=0.540	Unbiased
2	0.419685	0.401174	0.450985	0.409256	0.457046	0.410552	0.467465	0.476948
2.5	0.429095	0.412035	0.461643	0.420440	0.467961	0.421743	0.479117	0.486932
2.6	0.429443	0.412731	0.462114	0.414645	0.468458	0.422467	0.479696	**0.487117**
2.+2/3	0.429511	0.413037	0.462250	0.421492	0.468611	0.422788	0.479891	0.487043
2.673	**0.429512**							
2.7	0.429503	0.413147	0.462271	0.421611	0.468639	0.422906	0.479940	0.486955
2.709			**0.462272**					
2.716					**0.468642**			
2.853		**0.413362**						
2.861						**0.423143**		
2.865				**0.421856**				
3	0.428465	0.413206	0.461414	0.421719	0.467827	0.422999	0.479242	0.484997
4	0.419662	0.408039	0.452627	0.416575	0.459081	0.417790	0.470323	0.471765
5	0.409938	0.401659	0.442645	0.410151	0.449086	0.411306	0.459672	0.456995
10	0.381084	0.384481	0.412534	0.392734	0.418883	0.393752	0.423449	0.405272
. :								
∞	0.346574	0.366	0.375369	0.374	0.381	0.375	—	—

Table 5: Capacity for some CRAs with Free Access (Poisson arrivals)

d	STA	MTA (unbiased)	Optimally Biased MTA	
			Capacity	$\bar{p}$ *
2	0.360177	0.387222	0.393225	0.593200
3	0.401599	0.406970	0.407614	0.370911
4	0.399223	0.400746	0.400851	0.266662

Table 6: Capacity for the Limited Sensing 0.487 Algorithm (fair splitting, Poisson arrivals)

R	Capacity	Δ*
2	0.4493	2.58
3	0.4739	2.60
4	0.4853	2.60
5	0.4866	2.60
6	0.4870	2.60
7	0.48709	2.60
8	0.48711	2.60

Table 7: Packet Delay for the FCFS 0.487 Algorithm

λ	Mean							Standard Deviation		
	Analysis					Simulation		Analysis	Simulation	
	QT	[Georg87b]		[Huang85]		95% c.i.		QT	95% c.i.	
	(approx.)	lower	upper	lower	upper	mean	(+/−)	(approx.)	mean	(+/−)
0.01	1.53	1.53	1.53							
0.05	1.64	1.64	1.64							
0.10	1.81	1.80	1.81	1.80	1.80	1.807	0.004	1.057	1.060	0.011
0.20	2.31	2.27	2.35	2.29	2.30	2.312	0.005	1.78	1.782	0.016
0.25	2.73	2.66	2.80							
0.30	3.40	3.27	3.53	3.33	3.43	3.384	0.026	3.13	3.134	0.030
0.35	4.58	4.36	4.82							
0.40	7.22	6.80	7.67	6.53	7.31	7.191	0.160	7.34	7.297	0.097
0.42	9.43	8.80	10.11	7.91	9.30	9.440	0.223	9.66	9.657	0.233
0.44	13.57	12.58	14.63	9.94	12.6	13.46	0.35	13.91	13.674	0.451
0.45	17.34	16.03	18.75					17.75	18.149	0.907
0.46	23.92	22.04	25.95	14.00	19.2	23.78	1.03	24.40	24.771	1.426
0.48	92.91	85.09	101.45	22.4	38.5	94.09	13.11	93.55	100.071	22.702
0.487	5691.	5200.	6228.							