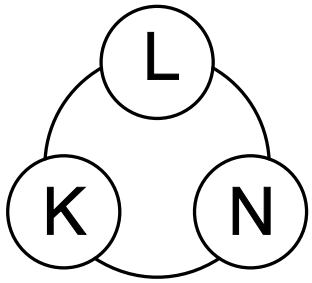


Technische Universitat München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Wolfgang Kellerer

Master‘s Thesis

Multi-Channel Tree Algorithms for LTE RACH
Reliable High Throughput Access

Author:	Martínez Alba, Alberto
Address:	Gabelsbergerstraße 79
	80333 München
	Germany
Matriculation Number:	03672751
Supervisor:	Murat Gürsu
Begin:	30. April 2016
End:	22. September 2016

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 22.09.2016	
Place, Date	Signature

München, 22.09.2016	
Place, Date	Signature

Kurzfassung

Das Aufkommen der Gerät-zu-Gerät Kommunikationen erfordert die aktuelle Mobilfunknetze in der nahen Zukunft fähig zu sein, mit Tausenden Verbindungen von Endgeräten gleichzeitig umzugehen. Darüber hinaus, um eine zuverlässige Dienstleistung sicherzustellen, müssen diese Verbindungen innerhalb strenge Verzögerungsbeschränkungen gewährt werden. In dieser Masterarbeit wird einen neuen Ansatz vorgestellt, um Zugangserfolg zu garantieren und um die Latenzzeit eines Mobilfunknetzes unter massive MTC-Eintreffen zu reduzieren. Dieser Ansatz basiert auf einer Mehrkanal-Umsetzung von Tree-Algorithmen, für welche ein detailliertes analytisches Modell entwickelt wurde. Zwecks Reduzierung der Komplexität dieses Modells, wird eine Approximation beruht auf der Markow-Eigenschaft angewendet. Schließlich wird die Verteilungsfunktion der Zugangsverzögerung eines mehrkanaliges Tree-Algorithmus erhalten. Die Gültigkeit des approximativen Modells wird durch Simulationen geprüft, die die vielversprechende Ergebnisse bestätigen.

Abstract

The emergence of Machine Type Communications (MTC) demands the current mobile networks to be able to handle thousands of simultaneous device connections in the near future. Moreover, in order to guarantee a reliable service, those connections need to be granted within strict delay constraints. In this thesis, a new approach to guarantee access success and reduce the latency of a mobile network under massive MTC arrivals is presented. This approach is based on a multi-channel implementation of Tree Algorithms, for which a detailed analytical model is developed. This model is achieved by means of a Markov approximation to simplify the analysis, and the cumulative distribution function of the access delay of a multi-channel Tree Algorithm is obtained from it. The validity of the approximate model is tested through simulations, which confirm the promising results yielded by the model.

Contents

Chapter 1

Introduction

Mobile networks are evolving towards 5G scenarios, in which connection speed will be faster, mobile coverage will be broader, latency will be shorter and the number of connected devices will be much larger [OBB+14]. In order to meet this requirements, numerous and significant modifications need to be applied to the current 4G (LTE/LTE-A) networks.

The predicted increase of the number of devices is mainly caused by the emergence of Machine Type Communications (MTC), through which thousands of different devices will be able to send their data in an unattended manner. However, the current mobile networks cannot handle the foretold increase in the number of simultaneous connections that MTC will generate, at least without seriously penalizing the latency, i.e. the access delay. New techniques to cope with a large amount of connected devices are needed, and at the same time latency needs to be kept low in order for delay sensitive devices to work.

In this thesis, a new approach is proposed, designed to lower the latency of LTE networks under massive arrivals. This approach is based on the application of so-called Tree Algorithms, a type of contention resolution algorithms that were created to resolve collisions in an efficient way.

In order to apply Tree Algorithms in a controlled manner, a complete analytical model will be obtained, from which delay related statistics will be derived. An approximation based on the Markov property will be used to simplify the model while keeping the accuracy of the derived statistics high. Those results obtained with the analytical model will be finally backed by means of simulations.

The rest of this thesis is organized as follows. In chapter 2, three background topics are covered: the current operation of the Random Access procedure of LTE is reviewed, the existing proposals to improve the LTE performance are explained and finally a description of Tree Algorithms is presented. In chapter 3, a description of the new approach proposed in this thesis is explained, which reveals the importance of the analysis of Tree Algorithm in multi-channel scenarios. In chapter 4, an analytical model for the delay related statistics

of multi-channel Tree Algorithms is derived. In chapter 5, simulations are presented to confirm the validity of the analytical model. Finally, chapter 6 contains conclusions and ideas for future work.

Chapter 2

Background

The emergence of Machine Type Communications (MTC) is creating new challenges and opportunities within mobile communication networks. Some industry analysts foretell that there will be more than 50 billion connected devices by 2020, most of which will be autonomous MTC devices, such as vehicles, utility meters or sensors [Eri11]. Owing to the vast deployment of these devices, mobile networks will be in charge of connecting these devices with their MTC servers and applications.

Current mobile networks, which are designed for Human Type Communications (HTC), need to be adapted in order to cope with MTC, since MTC exhibit a very different behavior with respect to classical HTC in at least three major aspects. In the first place, the large number of MTC devices will demand to redimension the current mobile networks in order to deal with so-called massive MTC (mMTC) [DSEAB+]. In second place, the type and size of messages from MTC devices will differ from those generated by handheld devices. Finally, the frequency of MTC transmissions will be less stochastic than human transmissions, due to scheduled reports and simultaneous responses to events.

Furthermore, some MTC applications will require strict delay constraints. For instance, data from eHealth devices or seismic sensors could be useless if they arrive to the MTC servers a few seconds after their generation. Therefore, new techniques to guarantee reliability, understood as the ability of the network to serve a device within the required time constraint, are needed. Namely, improving the Random Access Channel (RACH) of LTE/LTE-A has been defined as the first priority to achieve low delay MTC transmissions under massive arrivals [TR311].

In order to fully understand the situation and the possible solutions to this challenge, a review of the current LTE[1] specification regarding Random Access is presented in the first section of this chapter. The second section compiles the state-of-the-art proposals

[1] In the following, the acronym LTE will be used to refer to the standards proposed by 3GPP from Release 8 to 12, thus including LTE-Advanced.

to improve the LTE RACH for MTC, as an overview of the possible solutions. Finally, the third section is an introduction to Tree Algorithms, a type of Collision Resolution Algorithm that could be applied to the LTE RACH, being the central topic of this thesis.

2.1 Random Access procedure in LTE

In this section, the Random Access procedure in LTE as described by the current 3GPP specifications [3GP15a] and [3GP15b] is presented.

2.1.1 Procedure description

Let us look into a typical massive MTC scenario, where a large number of MTC devices are deployed within the coverage of at least one cell. The base station of each cell, called eNodeB in LTE, is in charge of providing the necessary resources for every MTC device to communicate with the MTC server. In this server, data is processed and forwarded to a specific MTC application. A diagram of this network is depicted in Figure 2.1.

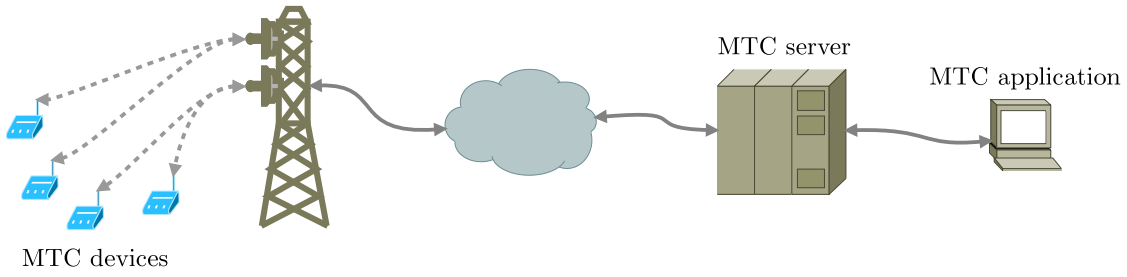


Figure 2.1: MTC scheme

When a MTC device has nothing to transmit, it may be disconnected from the mobile network so that resources are not wasted. A disconnected device is in an idle state and no allocated resources from PUSCH (Physical Uplink Shared Channel, for application data) or PUCCH (Physycal Uplink Control Channel, for control data) are granted to it [Cox14]. When the device gathers data that it intends to transmit, it needs to inform the network so that the eNodeB allocates the required resources. This is accomplished by applying the Random Access procedure, by which a disconnected device may ask for dedicated resources.

Apart from initial access, the Random Access procedure in LTE is used for other frequent situations, such as the following [DPS13]

1. During a Radio Resource Control (RRC) Connection Reestablishment, e.g. after some link failure.

2. When the uplink synchronization is lost.

3. When a handover is being performed.
4. When there are no further PUCCH resources available to perform a scheduled request.

Except for handovers and some other special cases, the Random Access procedure is contention-based. This means that collisions among transmissions of different devices might occur. Hence, it is crucial for those collisions to be resolved efficiently in order to optimize the operation of the network.

According to the 3GPP specification, a successful Random Access procedure in LTE consist in the exchange of four messages, depicted in Figure 2.2. The purpose and characteristics of such messages are the following [Cox14]:

- **Message 1** is a randomly selected preamble which informs the eNodeB that there is one device that is ready to be attached to the network.
- **Message 2** is also called Random Access Response (RAR). It consists of a scheduling command that informs the device which resources has been allocated for it, in order to continue with the request of definitive resources.
- **Message 3** is a standard RRC Connection request transmitted in a dedicated resource. This message informs the eNodeB about the identity of the device and the type of resource that is being demanded.
- **Message 4** is another scheduling command which grants the required resources to the device, or denies them in case of error or overload. Since it clearly identifies which device is being granted access, it serves as contention resolution step.

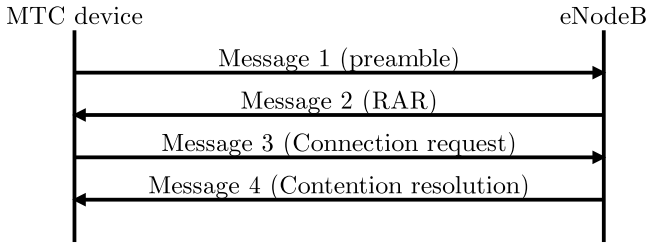


Figure 2.2: Sequence diagram of the Random Access procedure in LTE

Among this four messages, the singular characteristics and the importance of the successful reception of the first message make it worthy of special attention. This message is sent through the PRACH (Physical Random Access Channel), which consists of a set of time slots called Random Access Oportunities (RAOs). The duration and frequency of these RAOs, which are previously announced by the eNodeB in the System Information Block 2 (SIB-2), may be configured between one every 20 ms and one every 1 ms [DPS13]. Whenever a RAO is taking place, any device can transmit Message 1 and hence try to access the network.

The content of the Message 1 is different from any other data transmission in LTE. It consists of a randomly selected sequence called 'preamble', which is designed to cope with contention, i.e. with simultaneous transmissions. Indeed, those preambles are Zadoff-Chu sequences, which are complex-valued sequences that have good auto- and cross-correlation properties [DPS13]. This means that a preamble transmitted by one device should not be interfered by preambles transmitted by other devices or own echoes. A set of 64 orthogonal preambles are available at every eNodeB, but not all of them are eligible for a device which tries to access the network. Some of them are reserved for contention-free operations, such as handovers. In initial access, devices randomly choose one preamble out of the available preambles, what might cause collisions if two or more devices transmit the same preamble in the same RAO.

In case that two or more devices transmit the same preamble simultaneously, the eNodeB will not detect it immediately. Instead, only the activation of one preamble would be detected, so that only one Random Access Response would be generated and sent back. Every collided device would receive this response and would try to send Message 3 in the same time-frequency resource. As a consequence of this, the eNodeB would receive a collided combination of Messages 3, which would be usually impossible to decode. In this situation, the 3GPP specification does not provide a way to inform about collisions. Instead, all collided devices would time out while waiting for Message 4 and they would restart the procedure from the beginning. Since the restart of the process would generate the same result if nothing is changed, devices will wait a random back-off time before retransmitting [Cox14].

According to the described contention-based operation of LTE, it can be readily related to Multi-Channel Slotted ALOHA. The use of orthogonal preambles implies Code Domain Multiplexing (CDM), so that we can translate those preambles into separated channels. This means that within every RAO we have as many channels as available preambles, and those channels can be transmitted in parallel, hence forming a multi-channel scenario. Beyond this, every device freely decide to transmit within a given RAO, which is the identifier characteristic of Slotted ALOHA. Thus, we should expect the current Random Access procedure of LTE to have the same performance as that of multi-channel Slotted ALOHA.

2.1.2 Performance evaluation

At this point, the behavior of the current Random Access procedure in LTE can be evaluated in terms of two key performance indicators:

- **Access success probability:** probability that a given device successfully completes the Random Access procedure.
- **Access delay:** random variable that models the time difference between the initial transmission and the access completion of a given device. As a random variable,

its Cumulative Distribution Function (CDF) is required to completely characterize it. Nonetheless, sometimes only the average and some measure of the dispersion is provided for the sake of simplicity.

The 3GPP provided in [TR311] simulation results of typical MTC scenarios in terms of these two performance indicators. They showed that the current Random Access procedure is able to handle low to moderate congestion situations, up to 10 000 devices transmitting within 10 seconds. However, in an scenario where 30 000 devices transmit beta-distributed within 10 seconds, the access success probability yielded by the current standard is only 29.5%, with a maximum access delay of 180 ms. This means that more than 70% of devices would be unable to communicate with their servers in this case.

In light of this result, it is clear that the current Random Access procedure in LTE needs to be improved in order to guarantee access success even for highly congested situations. Furthermore, any improvement that were applied needs to provide low access delays for those devices that are delay sensitive. In the next section, a compilation of the proposed improvements to the LTE RACH is presented.

2.2 Proposed improvements to the LTE RACH

There are numerous proposed modifications to improve the performance of the LTE Random Access Channel, in terms of access success probability and access delay. Very different strategies can be followed to accomplish this goal. For instance, some proposals directly tackle the problem of contention: either they attempt to avoid collisions or they try to resolve collisions in an efficient way. Alternatively, other potential solutions propose to separate resources by priority, so that delay sensitive devices are served faster than delay tolerant devices. Finally, there are other proposals that cannot be classified as any of the above, but they benefit from significant modifications on the MAC layer of LTE.

In the following, a review of the current proposals that can be found in the literature is presented.

2.2.1 Collision avoidance techniques

The idea behind all these techniques is to reduce the number of collisions to an acceptable level, so that devices can be served efficiently and the probability of not completing the access procedure becomes negligible. This can be accomplished through different trade-offs, which are explained below.

Dynamic Resource Allocation

When the eNodeB can predict or detect that a large number of devices are about to access the network, the intuitive solution is to allocate more RACH resources in order to avoid contention. This is the idea behind this solution, which was suggested by 3GPP in [TR311].

As mentioned in the previous section, the period of Random Access Opportunities (RAOs) can be configured from 1 ms to 20 ms. This is accomplished by changing the PRACH Configuration Index [3GP14], whose effect on the uplink time sequence can be observed in the Figure 2.3. Assuming that 54 preambles are available for MTC devices, the different configurations imply that the maximum number of devices that can access the network within one second (under ideal conditions) ranges from 2700 to 54 000. Therefore, in case of congestion, the network could simply be reconfigured to serve more devices.

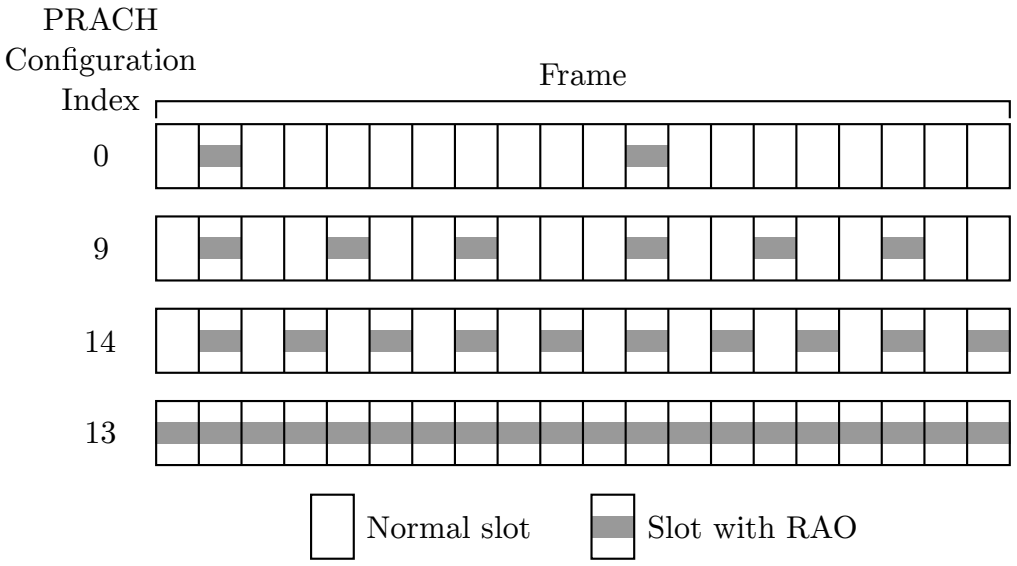


Figure 2.3: Different configurations for RAOs

The trade-off is however evident, the more resources are used for random access, the less resources remain for data transmission [LAAZ14]. Furthermore, the eNodeB needs to be aware of congestion situations in order to react to them.

The performance of a network applying Dynamic Resource Allocation was simulated in [3GP10]. They concluded that this technique is indeed useful to alleviate congestion, even for 30 000 contenders in 10 seconds. They showed that moving from 4 to 8 RAOs every 20 ms produces a twofold increment of the access success probability. Although it might be not enough to use this solution alone, the combination with other techniques could be more beneficial.

Slotted access

This solution proposes to look back into the roots of *multiple access* in order to avoid contention under massive arrivals. In general, *multiple access* is defined as the situation where many senders of information have to share a common channel [Mas14]. There are two approaches to address the issues derived from this situation: *random access* and *scheduled access*.

As we already know, in *random access* every device is allowed to seize the channel when it has some data to transmit. This approach is moderately simple and it yields good results when the number of simultaneous arrivals is not too high. For instance, Sloted-ALOHA reaches maximum efficiency when all devices transmit around 36.8% of the time.

However, when the number of devices is high and they transmit frequently, *scheduled access* is usually a better approach. In *scheduled access*, every device is granted exclusive access to the channel at some (scheduled) time. Since there are no contention, the efficiency can be better than using *random access*. Nevertheless, if devices rarely transmit, resources are wasted and long delays are introduced.

In case of massive MTC, it can be argued that the arrivals are numerous enough to justify *scheduled access*. In [TR311], they propose to relate every MTC device with a dedicated Random Access opportunity, whose location can be computed by using the device ID and some parameters broadcast by the eNodeB. The sequence of Random Access opportunities would finally repeat after every device has been able to transmit.

The drawback of this solution stems from the potentially excessive delay that it will introduce when the number of devices is high, even if few of them are actually transmitting [LAAZ14].

Pull-based scheme

The aforementioned solutions were both based on a push-based scheme: devices are the ones that decide when to transmit. However, in some situations this is not required, but the MTC server could demand the devices to transmit when it is necessary. The MTC server would simply inform the eNodeB about the devices that need to be paged for them to send the required information [TR311]. By doing this, the eNodeB would handle the congestion of paging messages and then collisions would be reduced or even removed. Nevertheless, a pull-based scheme is only suitable for some scenarios and would require intense use of control channels [AHK16].

Group-Based Access

This solution, proposed in [FI13], is a pull-based scheme that takes advantage of groups of MTC devices that are in close proximity from one another. As a pull-based solution, devices are only allowed to transmit after receiving a paging message. A paging message in LTE may contain up to 16 device IDs, so that 16 devices can be paged at the same time. The authors propose replacing these device IDs with group IDs, belonging to so-called access groups. Whenever an access group is paged, a single device (the group delegate) is responsible for performing the Random Access procedure to obtain a RRC connection. This connection will be reused by every device in the access group, as long as they are close enough to share the same Time Advance, needed for establishing a RRC connection.

Simulation results were also presented to confirm that this solution reduces RACH congestion and hence access delay when compared with individual access. Nonetheless, it can be only applied to pull-based scenarios where MTC devices are close together.

2.2.2 Collision resolution techniques

Instead of avoiding collisions, another possible approach is to let collisions occur and then try to resolve them efficiently, in terms of time and used resources. This means that some strategy to isolate contenders after a collision needs to be in place. This strategy can be either simple, such as using back-off times after every collision, or more elaborated, such as applying a Tree-splitting Algorithm. The characteristics of both options are explained below.

MTC-specific back-off

According to the current LTE specification [3GP15a], after a collision in the Random Access procedure a device will back off for some random time before retransmitting. This technique is useful when coping with arrival bursts, since it will scatter subsequent attempts after the first collision [Tan03]. Therefore, it might be used to alleviate congestion in the network in case of massive MTC arrivals.

Nonetheless, simulation results in [3GP10] showed that the improvement achieved by back-off tuning is only useful for low congestion scenarios. Otherwise, this solution yields low access success probabilities (below 30% for a maximum backoff of 960 ms and 30 000 contenders in 10 seconds) and long delays for those that succeded (up to 8 seconds).

Tree-splitting Algorithms

In [nSP14], a contention resolution algorithm based on Tree-splitting Algorithms is presented. Tree-splitting Algorithms are designed to cope with collisions by dividing collided devices into increasingly smaller groups, until every device successfully transmits [Mas14]. As an alternative to Slotted-ALOHA, it can be also applied to the Random Access procedure of LTE. Only some extra feedback after a detected collision is required for the algorithm to perform.

The simulation results provided for this solution in [nSP14] are very promising, since the access success probability for 30 000 arrivals in 10 seconds is very close to one and the maximum access delay is below 2 seconds. Indeed, the use Tree-splitting Algorithms is the solution that will be studied in this thesis. A more comprehensive description of these algorithms can be found in the section 2.3.

2.2.3 Resource separation techniques

The solutions hitherto mentioned assume that all devices in a congested network have the same requirements. However, usually only a fraction of the contending devices are delay sensitive, which can be exploited to handle congestion. Namely, the network could reserve resources for delay sensitive devices, so that only delay tolerant devices are affected by long delays or outages.

Although the separation of resources can be effective in common scenarios, it is important to note that this strategy only displaces the problem of congestion towards those devices which are least sensitive to delay. As a consequence, if the number of delay sensitive devices is too high, separating resources would not be effective unless it is combined with other techniques.

In any case, the idea of separating resources has been applied to numerous proposals. Most of them rely on blocking the access of delay tolerant devices in case of congestion through so-called *Access Class Barring*. In addition, some solutions propose to physically separate resources dedicated to MTC and HTC, such as *Prioritized Random Access* and *Self-optimization overload control*. All these proposals are explained in the following.

Access Class Barring (ACB)

Access Classes (ACs) are included in the 3GPP specification in [3GP15c] as a way to prioritize some user equipments (UEs) with respect to the rest. Sixteen ACs were originally defined to separate ordinary UEs (AC 0 to 9) from special UEs (AC 10 to 15), such as those belonging to emergency or security services. In case of congestion, the network might block or throttle the access of some ACs in order to allow others to communicate.

The control over ACs is performed in two different ways. For special UEs, a set of status flags is broadcast by the eNodeB in the System Information Block 2 (SIB2). These are boolean flags that indicate whether a specific AC has been barred or not.

For ordinary UEs, such as MTC devices, the eNodeB broadcasts two different parameters for each AC: a probability factor and a barring timer. Every device will check whether it is allowed to transmit by drawing a random number and comparing it with the probability factor. If the probability factor is greater than the random number, the device is allowed to transmit. Otherwise, the device will use the barring timer to compute a random back-off time, and it will await that time before trying again.

By controlling the probability factor and the barring timer, the eNodeB can control the number of devices trying to access the network without specific signaling for rejecting petitions. Therefore, cases of congestion might be efficiently handled through ACB. For this reason, in [TR311] the 3GPP considered ACB as one of the possible solutions to congestion caused by massive MTC. Furthermore, they proposed two different extensions to the basic ACB:

- **Individual ACB scaling.** As the name suggests, this technique allows for individual control of the ACB parameters for every device or group of devices with the same requirements. In case of massive MTC this is however clearly impractical, since it would require excessive signaling.
- **Extended Access Class Barring (EAB).** In 3GPP Release 11 [3GP16], an enhancement of ACB was introduced to further cope with MTC. In EAB, devices that are delay-tolerant are labeled as 'configured for EAB'. In case of congestion, such devices are barred and thus only delay sensitive devices are allowed to transmit. This barring is also done with an AC granularity, i.e. delay-tolerant devices of each AC can be blocked independently, what contributes to a finer control of the cell status.

  In order for MTC devices to be informed about their EAB situation, a combination of broadcast information and paging is used. The set of enabling flags are broadcast in SIB14, and in case of updates for a certain AC, devices are informed through paging. In this way, convergence of requests due to simultaneous reading of SIB14 is avoided. As a consequence, the frequencies of SIB14 broadcasts and paging become relevant for the performance of EAB.

  In [TR311], this technique it is believed to be a feasible solution of the congestion problem, and hence it is widely used in combination with other proposals.

In [3GP10], the performance of basic ACB was tested for several values of ACB parameters. It was found that under high congestion conditions (30 000 devices in 10 seconds), the probability for one device to successfully access the network is below 70%, with an access delay sometimes greater than 10 seconds. However, for lighter conditions (10 000 devices in 10 seconds), the network manages to serve every device with delays mostly less than 10 seconds. It was concluded that basic ACB may be useful under light congestion conditions,

although not for high congestion unless long delays are accepted.

Regarding EAB, in [CCCW15] an analytical model for predicting the performance of EAB under some enabling algorithm is presented. Among the results, it is worth mentioning that the access success probability increases as the ratio between frequency of SIB14 and frequency of paging increases. Furthermore, they selected a minimum paging cycle of 1000 ms and a minimum ratio between paging and SIB14 cycles of 5 in order to guarantee an access success probability greater than 90% for 30 000 contenders in 10 seconds. On the subject of delay, only average values are shown, which are around 10 seconds for the aforementioned values. This delay applies for delay tolerant devices, which explains the apparently high value.

Apart from the solutions considered by the 3GPP, other authors have proposed extensions to the ordinary ACB with the intention of making it responsive to changes in the congestion level (dynamic ACB) or exploiting multi-cell coverage (cooperative ACB). These extensions are described below:

- **Dynamic ACB**. This technique improves basic ACB through the continuous updating of the probability factor that is broadcast by the eNodeB, which needs to accurately monitor the load of the cell.

  Several approaches and implementations of Dynamic ACB have been proposed. In [DSMW13], all MTC devices are assumed to be contained in the same Access Class and thus they are all controlled by the same ACB probability factor p, which is employed as in basic ACB. Therefore, before attempting to transmit at the next RAO, a device draws a random number between 0 and 1 and compares it with p. According to the number of available preambles M and the number of devices transmitting in one RAO n, the optimum value of p is found to be:

  $$p^* = \min\left(1, \frac{M}{n}\right). \tag{2.1}$$

  Although this is the value of p that guarantees optimal performance, n is in general unknown by the eNodeB, therefore p cannot be exactly computed but approximated. With this purpose, the authors suggested an algorithm that uses the number of successful transmissions and collisions detected by the eNodeB instead of n. Their algorithm exhibits a near optimal performance and it achieves a threefold reduction of the access delay with respect of basic ACB.

  The Dynamic ACB approach is also frequently combined with other techniques. In [FA15], the authors suggest to use the ACB probability factor as a decision parameter for devices to select the less loaded eNodeB. Namely, eNodeBs dynamically set the ACB factor as the probability for one device to successfully transmit a preamble. They assumed a Poisson arrival process, so that this probability is simple to compute, but consequently their results cannot be directly applied to massive MTC.

Finally, in [LLCC14] Dynamic ACB is combined with *Prioritized Access*, which is explained later. This time, the access barring is performed in a different manner. Instead of using a probability factor, the authors propose that the eNodeB broadcasts a barring flag when it reaches a high-loading state. This state is declared when the expected number of successfully decoded preambles falls below some threshold. In this case, when a device is barred will wait some backoff time before attempting transmission.

- **Cooperative ACB**. This approach benefits from places covered by multiple cells, within which MTC devices can select to transmit to different eNodeBs. Currently, the LTE/LTE-A standard does not include the level of congestion of one cell as a relevant parameter to choose an eNodeB in initial access. This means that one device could choose an overloaded cell instead of a light-loaded cell due to less important reasons, like a small difference in the received power signal.

  In order to solve this problem, the authors in [LLKC12] propose to use the ACB factor as a decision parameter when choosing eNodeBs. For that it is necessary to jointly optimize the ACB parameters of the eNodeBs covering some given area, which is also offered by the authors. Simulation results show that Cooperative ACB outperforms ordinary ACB in terms of access delay with limited impact on the standard, although it is not useful when devices are covered by a single cell.

Prioritized Random Access

In this improvement, dynamic ACB is combined with the separation of RACH resources for MTC and HTC. In [ChLL11] and [LLCC14], five new ACs are suggested to group different types of devices: HTC devices, high priority MTC devices, low priority MTC devices, scheduled devices and emergency traffic. Each AC can only use some RAOs, so that the interference among different traffic is limited. For instance, the authors propose a distribution of RAOs in which those used by emergencies and scheduled traffic are disjoint from those used by MTC devices, whereas HTC traffic is allowed to use any RAO. Apart from this distribution of RAOs, they also propose different back-off policies, so that higher priority traffic is allowed to transmit first in case of collisions. This manner of separating RACH resources according to ACs is called Virtual Resource Allocation.

Furthermore, Virtual Resource Allocation is combined with dynamic ACB, in order to benefit from the advantages exposed before. The combination of both techniques was put to the test through simulations in [LLCC14], in which a traffic of around 30 000 low priority devices in 10 seconds along with high priority traffic was considered. Simulation results showed that Prioritized Random Access lowers the delay and increases the access success probability with respect to EAB for high priority classes. However, for the simulated emergency traffic, this technique achieved an average delay of 2.9 seconds, which could be problematic in a real scenario. Moreover, the performance of low priority devices was

disregarded in the simulations.

Self-optimization overload control (SOOC)

This scheme is described in [LLJK11] as an enhanced combination of dynamic resource allocation, separation of RACH resources and dynamic ACB. The proposal relies on an intelligent control loop that monitors and configures RACH parameters dynamically.

The first function of the control loop is to monitor the level of congestion of the cell. This is done by using feedback from devices that managed to successfully access the network. When accessing the network, SOOC includes the usage of Dynamic ACB as explained before, with two classes for MTC devices: low and high priority. By using Dynamic ACB, simultaneous arrivals are dispersed after the first collision, so that the access success probability increase. After successful transmission, devices inform the eNodeB about the number of retransmission that they performed. This parameter will be used as congestion indicator by the eNodeB.

Once the eNodeB has gathered enough feedback from devices, it is in charge of readjusting the RACH parameters to adapt them to the congestion status. Dynamic RACH resource allocation is used at this point and the number of RAOs per second required for some congestion level is computed as a function of the collision probability that is estimated by using the feedback. If the number of resources that can be allocated is not enough, low priority devices can be additionally barred. Finally, the eNodeB broadcasts the decided modifications before the loop starts again.

Unfortunately, the authors did not include simulative results along with their analysis, therefore the effectiveness of this solution is still unconfirmed.

2.2.4 Other techniques

Finally, there are some proposals that cannot be categorized under any of the above groups, since they propose substantial modifications on the MAC layer of the Random Access procedure in order to gain additional benefits.

Optimized MAC

For those cases where the amount of data transmitted per device is small, the current Random Access procedure in LTE may be burdensome. Instead, in [CW10] it is proposed that small pieces of data are sent in Message 3 or even in Message 1 during this procedure.

As first option, they propose to remove the RRC Connection requested in Message 3 and granted in Message 4 and transmit a MAC PDU (Protocol Data Unit) of information in

Message 3 instead. In this way, devices would not even need to support the RRC protocol and delay and congestion should be reduced.

As second option, a even more aggressive reduction of messaging is presented. Small pieces of data are suggested to be encoded as special preambles, so that when the eNodeB receives them, it interprets them as application information from the device. In this option, only a preamble transmission and an ACK would be needed.

Although their promising advantages, this work is not backed by any simulative results at the time of writing.

Spatial Group-based Random Access (SAGRA)

The proposal presented in [JKK+14] relies on spatial grouping of MTC devices in order to reuse preambles for devices that are far away from one another. Therefore, it is necessary to modify the way that preambles are detected, but an increased amount of Random Access opportunities can be obtained in return.

Code-expanded Random Access

In [PTSP12], the authors presented a generalization of the current LTE RACH procedure. Their proposed solution groups Random Access opportunities into virtual frames, so that the preambles transmitted by one device merge into a single *codeword*. which replaces a single preamble as the sequence used to detect transmissions. Applying this technique, the authors showed that Code-expanded Random Access increases the ratio of contenders that succeeded at accessing the network with respect to the standard LTE RACH, at least for high user loads. However, neither analytical nor simulation results regarding access delay were presented.

Proactive scheme

In [nPSP15], a method for estimating the number of devices trying to access the network within certain RAO is presented. This estimation is applied to provide reliability guarantees even for massive arrivals.

This solution uses an *access frame* consisting of a sequence of RAOs, with a given maximum length. The first RAO in the access frame is the *estimation phase*, in which the number of devices that are trying to access the network is estimated based on the number of collisions, successful transmissions and empty preambles, Once the number of devices is known, the number of RAOs that is needed for them to successfully transmit with a certain probability can be computed. Therefore, in the next phase, the *serving phase*, the necessary number of RAOs is allocated and devices continue transmitting as usual. If the number of necessary

RAOs happens to be greater than the allowed maximum, a barring factor is also computed and applied. In this way, delay sensitive devices can be served with a guaranteed reliability, i.e. a guaranteed access success probability.

Nevertheless, for this solution to work properly, the number of contenders cannot change along the serving phase. This implies that no new arrivals are allowed to access the network until the ongoing serving phase has finished. Therefore, the authors propose a gated access solution. One important drawback of this approach is that it may add intolerable delays to devices that are not being served but are waiting for others to be served. However, neither analytical nor simulation results are provided to measure the relevance of this problem.

With the help of some numerical results for a specific case study with two different traffic classes, the authors showed that (under their assumptions) their solution was able to outperform legacy LTE under strict reliability constraints. For instance, 30 000 devices activated within 10 seconds were able to successfully transmit in less than 5 seconds with a probability greater than 90%.

Probabilistic Rateless Multiple Access

A novel perspective is offered as another proposed solution in [SLD+15]. First, the set of available preambles for contention-based operation are divided into groups and related with different delay requirements. Then, the eNodeB should be able to estimate the number of devices that activated some preamble by checking the received power of that preamble. Once the eNodeB knows how many devices of each delay group are trying to access, it sends a Random Access response containing that information. Then every involved device has to generate orthogonal random seeds that will be used by the eNodeB to serve the devices through Multiple Access Analog Fountain Code (MA-AFC).

According to the simulation results provided by the authors, this technique is able to guarantee the access of 10 000 devices with an access delay lower than 1 second.

2.3 Tree-splitting Algorithms

In the section 2.2.2, Tree-splitting Algorithms were mentioned as the technique that is comprehensively studied in this thesis. With the intention of providing a basis for understanding the analysis that is presented in Chapter 4, an overview of the operation and types of Tree-splitting Algorithms is presented here.

Tree-splitting Algorithms, or simply Tree Algorithms, are a group of Contention Resolution Algorithms, which are designed to efficiently solve collisions occurred in a random access scenario. The idea behind all of them is simple: after a collision, the involved devices are divided into subsets before they are allowed to transmit again [AZKC+08]. This strategy

provides a structured way of resolving contention, which usually performs better than basic algorithms such as Slotted-ALOHA, although they need extra information feedback to work. A detailed description of their characteristics is presented in the following.

2.3.1 Operation of Tree Algorithms

Let us consider a scenario where four devices (A, B, C and D) are served by the same base station. In this scenario, we will be using the following assumptions:

1. The multiple access scheme is *random access*, i.e. every device initially transmits in an unscheduled manner regardless of other devices.
2. Time is *slotted*.
3. The base station broadcasts a *feedback* message after every time slot, informing whether a collision was detected. The feedback channel is assumed to be separated from the random access channel.

Given this assumptions, let us now suppose that those devices attempt to transmit at the same time slot, and hence they collide. The base station informs about a collision via the feedback channel, so that devices know that their transmissions were not successful. At this point, if Slotted-ALOHA were applied, every device would wait for a random back-off time before retransmitting. Instead, when using a Tree Algorithm, every device will draw a random number that will be used to group them into increasingly smaller subsets of the initial collision.

In our example, let us suppose that devices can only choose between 0 and 1. By doing that, we will be using a Binary Tree Algorithm (BTA), since devices have only two options. After the first collision, the following decisions were made: B and C chose 0, and A and D chose 1. Therefore, B and C are in the first group and A and D are in the second group. Every new group will be eventually translated into a new slot in time domain, hence a group with more than one device implies a new collision. The exact location of those slots depends on the implementation of the Tree Algorithm, which will be explained later.

After the first grouping, we are able to 'predict' two new collisions: B-C and A-D. In order to resolve these collisions, we apply the same operation. Each device has to generate a new random number in order to be split again. For instance, the result of the division of the first group could be: B chose 0 and C chose 1. Whereas the result of the second group could be: A and D chose 0 and no one chose 1. In this case, the collision of the first group would be resolved, since both new groups contain only one device. However, the division of the second group leads to an empty group and a new collision. Therefore, further splitting is needed until collisions disappear.

This splitting procedure can be easily understood when it is plotted as a tree diagram, like the one in Figure 2.4, which shows the previous example. In a tree diagram, every group

of devices is represented as a node of the tree, and nodes are related with others through the sequence of chosen random numbers. It can be observed that the initial node of the tree is the initial collision, and every collision branches off into two new nodes.

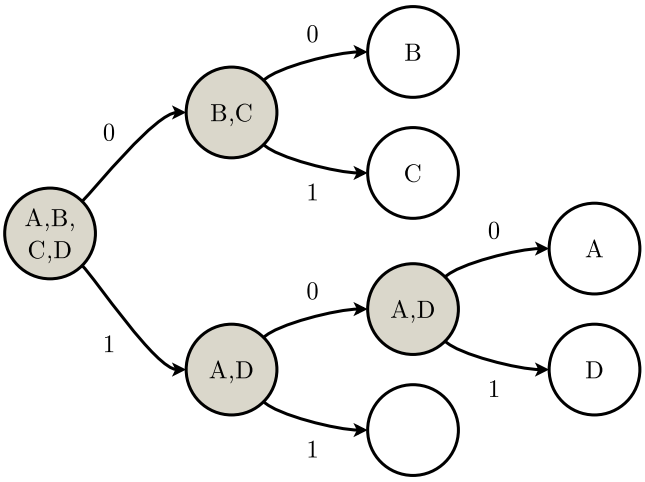


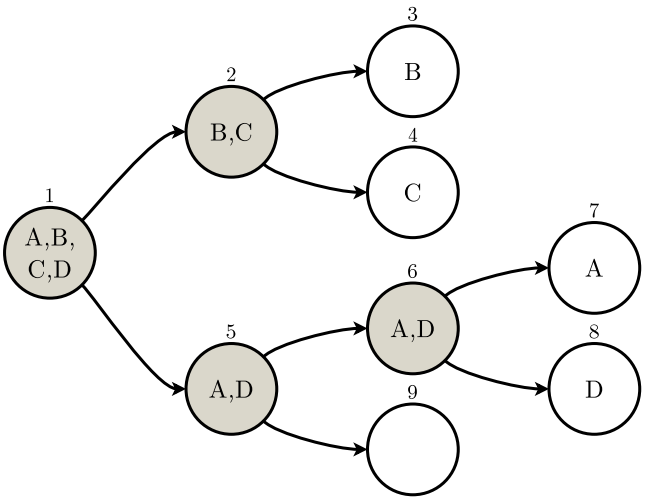
Figure 2.4: Example of Tree Algorithm

As it was mentioned before, every group of devices or, equivalently, every node of the tree can be translated into a time slot. Nonetheless, it is not evident how nodes can be arranged into a time sequence. In fact, there is not an unique manner to do that. The only limitation is that a child node cannot be transmitted before its parent nodes, which in time domain simply implies that a collision cannot be resolved before it happens. Other than that, one could choose to arrange nodes over time as desired.

Among the multiple possibilities for sequentially traversing the tree, there are two widely used ones [AZKC+08]:

- The first one, called **serial traversing** in this document, is depicted in Figure 2.5, where the number above each node denotes the slot in which it was transmitted. This way of traversing the tree follows collisions in a nested way. In our example, collision B-C (node 2) is fully resolved before starting to resolve collision A-D (node 5), although both are children of the initial collision. Furthermore, the second collision A-D (node 6) is resolved before the second, empty child node of node 5 is transmitted.
- The other possibility for traversing the tree is called **parallel traversing**, which is depicted in Figure 2.6. In this case, the closer a node is to the initial node, the sooner it is transmitted. If two nodes are at the same distance from the initial collision, the upper node is transmitted first.

Now that the basic operation of Tree Algorithms has been presented, we are ready to introduce the following definitions:

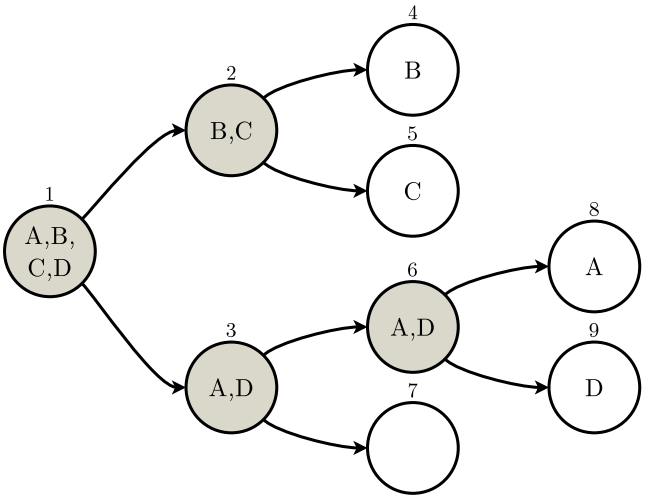


Feedback channel

Collision	Collision	No collision	No collision	Collision	Collision	No collision	No collision	No collision
A,B,C,D	B,C	B	C	A,D	A,D	A	D	

time

Figure 2.5: Example of a serial traversing of a tree



Feedback channel

Collision	Collision	Collision	No collision	No collision	Collision	No collision	No collision	No collision
A,B,C,D	B,C	A,D	B	C	A,D		A	D

time

Figure 2.6: Example of a parallel traversing of a tree

- **Node:** group of devices that share the same splitting decisions.
- **Nodal degree:** number of branches that sprout from every node. Henceforth it will be denoted as Q. For instance, in the Binary Tree Algorithm $Q = 2$.
- **Contention frame:** set of children nodes with the same parent node. The number of nodes in a contention frame is thus also Q.
- **Level:** set of nodes with the same number of ancestor nodes. The level number, denoted as m, ranges from 0 (the initial node) to infinity.
- **Tree length:** number of nodes needed to complete the tree. It is denoted as L, and it is equal to the number of slots needed to solve the initial collision.

In the Figure 2.7, an illustration of some of the previous definitions can be found.

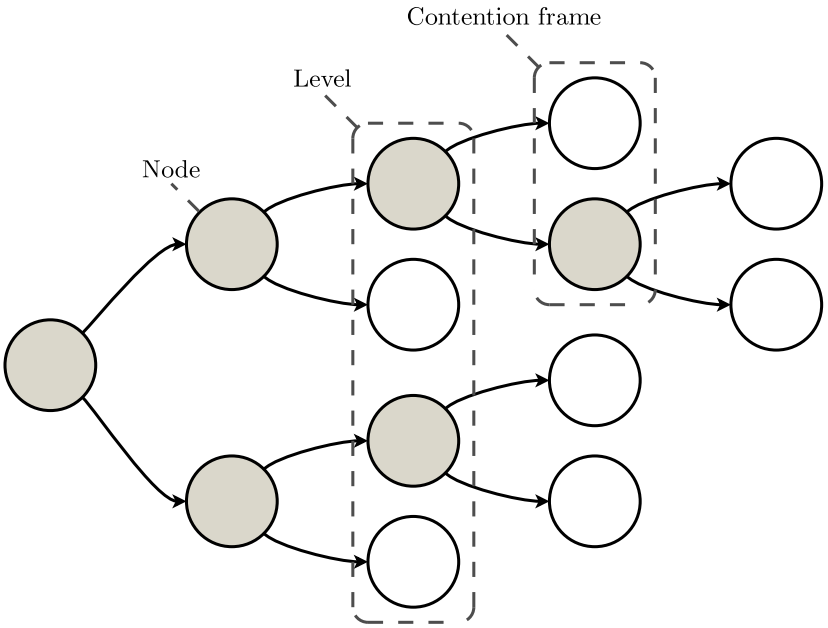


Figure 2.7: Terminology used in tree diagrams

2.3.2 Types of Tree Algorithms

Tree Algorithms can be classified according to their nodal degree, the way in which they are traversed and the use of different strategies to produce and terminate them. In this section, an overview of the most common Tree Algorithms is presented.

Binary Tree Algorithm

The Binary Tree Algorithm (BTA) is the algorithm explained in 2.3.1. It was the first Tree Algorithm to be described, by Tsybakov and Mikhailov in 1978 [TM78] and independently by Capetanakis in 1979 [Cap79b]. It is simple to implement and only requires binary feedback from the base station (collision or no-collision).

Regarding performance, the average tree length given N initial contenders is approximately $\frac{2N}{\ln(2)}$ [Mas14]. Therefore, the total throughput (ratio between successful slots and total slots) is $\frac{\ln(2)}{2} \approx 0.346$. One can notice that this throughput is lower than that of Slotted-ALOHA, which is said to be $\frac{1}{e} \approx 0.368$. Nonetheless, BTA offers a more stable behavior [AZKC+08].

Modified Binary Tree Algorithm (MBTA)

This is an improved version of the previous algorithm, that achieves a higher throughput. The improvement, called Massey's improvement as it was first described by Massey in [Mas14], is based on the skipping of predicted collisions. Collisions can be predicted to occur when the first node after one collision is an empty node. This is illustrated in Figure 2.8, where devices A and B collide. After the collision, they both choose 1, so that the node belonging to 0 is an empty node. In this situation, it is clear that the same collision will repeat after the empty node, therefore it would be better for devices A and B to draw a new random number instead of wasting a node in repeating the collision.

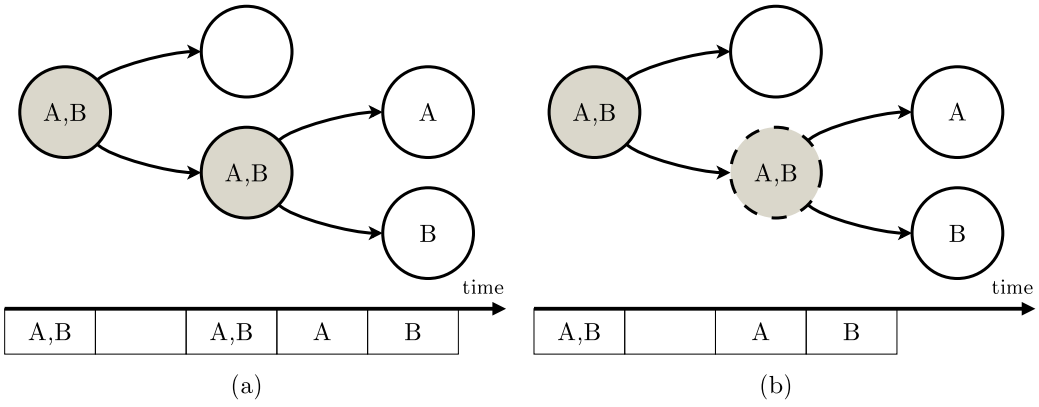


Figure 2.8: (a) Binary Tree Algorithm (b) Modified Binary Tree Algorithm

If this modified strategy is followed, the throughput of the Tree Algorithm increases to 0.375, outperforming Slotted-ALOHA. However, in this case the base station needs to provide ternary feedback (collision, empty slot, successful transmission) in order to apply the improvement. Furthermore, it has been shown that in case of error at receiving the feedback, devices might be unable to tell the end of the algorithm, leading to a deadlock situation [Mas14], which is a major drawback.

Q-ary Tree Algorithm (QTA)

This Tree Algorithm, first described in [MF85], follows the same principle as the previous ones, but with $Q > 2$ nodes per contention frame. The reason behind this modification is simple to understand. As we increase Q, the probability of a new collision after a parent collision decreases, since the same amount of devices are spread over a larger number of nodes. By reducing the number of collisions, we might expect a shorter tree and then enhanced throughput.

However, increasing Q also makes empty nodes more probable and hence more slots are wasted. As a consequence of this trade-off, it is clear that there must be some optimum value of Q that maximizes throughput. It has been found that for a standard QTA this optimum value is $Q = 3$, which yields a throughput around 0.366. Nevertheless, for Modified QTA (which also benefits from the Massey's improvement), the optimum value is $Q = 2$, i.e. the binary algorithm [AZKC+08].

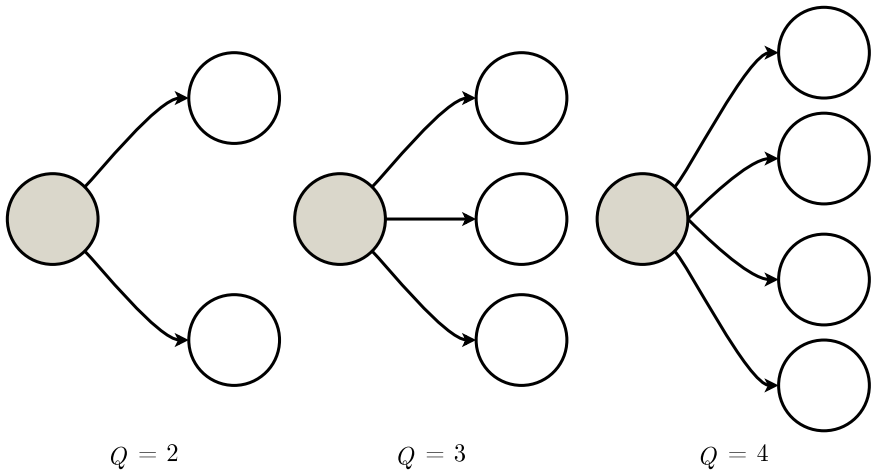


Figure 2.9: Examples of different nodal degrees

Dynamic Tree Algorithm (DTA)

A key characteristic of Tree Algorithms is that they are gated access algorithms, i.e. no new arrivals are allowed to transmit when a tree is being resolved. Therefore, those arrivals will be moved to the end of the tree, so that a new collision right after a tree resolution becomes probable if either the arrival rate or the length of the tree is high. The Dynamic Tree Algorithm, which was presented in [Cap79b] and [Cap79a] proposes to skip this collision and create one or several new trees to improve the performance. Nonetheless, this improvement is only achieved under Poisson arrivals.

First Come First Serve 0.487 Algorithm (FCFS 0.487)

This last algorithm, described by Gallager in [Gal78] and independently by Tsybakov and Mikhailov in [TM80] and by Ruget in [Rug81], benefits from windowed access, through which devices are separated according to their arrival time, and the so-called Gallager's improvement. This improvement relies on prematurely terminating a tree when a sub-optimal resolution is detected. This algorithm yields the best throughput (0.487) among Tree Algorithms, assuming a Poisson arrival process.

Chapter 3

Motivation

In the previous chapter, a comprehensive review of current proposals to improve the performance of LTE RACH was presented. In this chapter, we propose a new MAC approach to improve the access success probability and the access delay in case of massive MTC. This new option relies on dividing the RACH into two parallel systems: one devoted to new arrivals and the other dedicated to collision resolutions, in which a Binary Tree Algorithm is applied. As it is explained in the following, this new technique may benefit from the expansion of the ordinary Tree Algorithms into a multi-channel scenario, which is the motivation to study the characteristics of multi-channel Tree Algorithms in depth.

3.1 PASAT-Algorithm

The new proposal presented in this section is called PASAT-Algorithm (Parallel Slotted-ALOHA Tree Algorithm). This algorithm offers low access delay with respect to the current LTE RACH operation, even for highly congested scenarios. The description of the new medium access control (MAC) layer proposed by this algorithm is described below, along with promising simulation results.

3.1.1 MAC operation

As it was explained in the previous chapter, a Random Access Oportunity (RAO) in LTE consists of a time slot in which non-connected devices are allowed to transmit. Usually, up to 54 different orthogonal preambles are available for those devices to transmit, so that devices sending different preambles will not collide. Hence, the LTE RACH is a multi-channel system that can be represented as a preambles-time grid like in Figure 3.1.

In contrast to the homogeneous utilization of preambles in the current LTE RACH, the

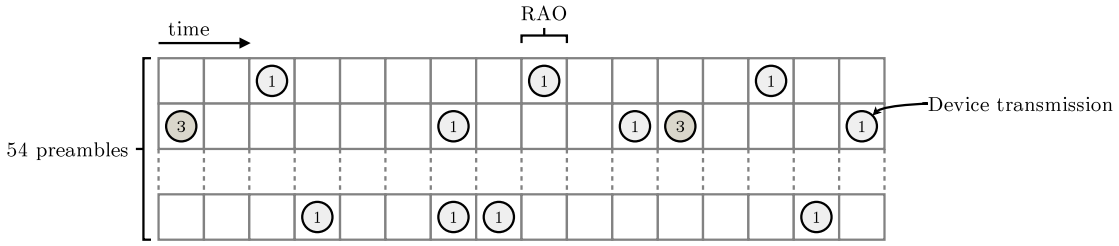


Figure 3.1: Grid representation of LTE RACH

PASAT-Algorithm proposes to divide preambles into two groups: a Contention Channel and Resolution Channel. New arrivals would randomly select a preamble within the Contention Channel. In case of collisions, these would be resolved through a Binary Tree Algorithm, which is performed in the Resolution Channel. In Figure 3.2, an example operation and the equivalence with a binary tree are depicted, assuming 6 preambles in total (3 in the Contention Channel and 3 in the Resolution Channel).

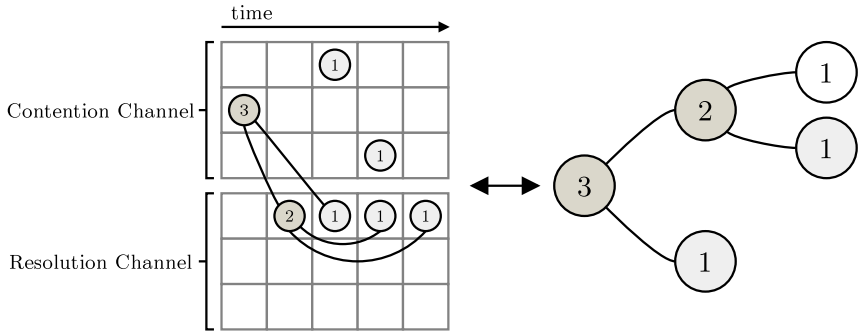


Figure 3.2: Example of single channel PASAT-Algorithm

As a variation of the idea shown in Figure 3.2, one could use preambles that are empty to transmit multiple nodes at the same time, and hence the access delay of the involved devices might be reduced. An illustration of this variation is shown in Figure 3.3, where two nodes are transmitted at the same time, so that the access delay is reduced.

It is important to notice that the number of resources (preamble-time square) remains constant even if a multi-channel approach is used. This implies a trade-off between access delay and the number of preambles occupied by a single tree in a given RAO, what might cause problems. For instance, if a second multi-channel tree were to run in parallel with that of the Figure 3.3, it would not have enough room to be executed until the completion of the first tree. Therefore its access delay would be higher than in the single channel case.

In order to check the viability of multi-channel Tree Algorithms like that presented in Figure 3.3, simulations were performed to compare single channel and multi-channel implementations of the PASAT-Algorithm.

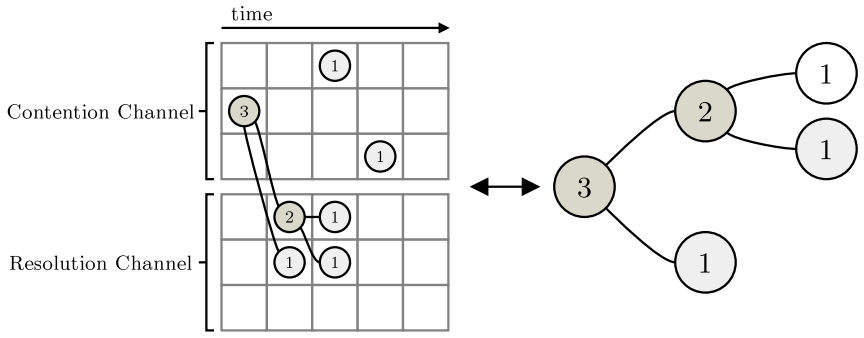


Figure 3.3: Example of multi-channel PASAT-Algorithm

3.1.2 Simulation results

We have seen that multi-channel Tree Algorithms might have potential to outperform ordinary Tree Algorithms when applied to the LTE RACH. In order to start to confirm this hypothesis, a simulator was built to measure the performance of a multi-channel implementation of PASAT with respect to a single channel one.

Simulations considered $H = 54$ available preambles, $H_{CC} = 18$ of which were dedicated to the Contention Channel and $H_{RC} = 36$ to the Resolution Channel. The reason for this division stems from the fact that a Tree Algorithm exhibits an stable behavior up to 0.34 arrivals per slot [Mas14], hence 2.88 slots per arrival are needed, which is the initial collision plus 1.88 additional slots. Since the initial collision is located in the Contention Channel and the remaining slots are in the Resolution Channel, the previous values are obtained after solving $H = \lfloor H_{CC} \rfloor + 1.88 \lceil H_{RC} \rceil$.

After the initial setup, for different scenarios were simulated:

- First, the most congested scenario considered by 3GPP in [TR311] was simulated: 30000 contenders beta-distributed over 10 seconds. In the following, we assume one RAO every 5 ms, therefore 10 seconds translate into 2000 RAOs. The result of this simulation is shown in Figure 3.4, from which we observe that the access delay was always below 1.5 seconds, a very promising result. Furthermore, we conclude that multi-channel Tree Algorithms do not make a difference in this case, due to the saturation of both Contention and Resolution Channels.

- Then, the first scenario covered by the 3GPP specification was put to the test: 10000 contenders in 10 seconds, which is a lighter loaded situation. The result of this simulation is depicted in Figure 3.5. In this case, multi-channel Tree Algorithms do perform better than single channel TAs, hence we might deduce that low congestion situations are more suitable for multi-channel TAs to properly work.

- Now, we can simulate a traffic burst with the same traffic density as in the first scenario, but with shorter duration. In Figure 3.6, the simulation results for a burst of

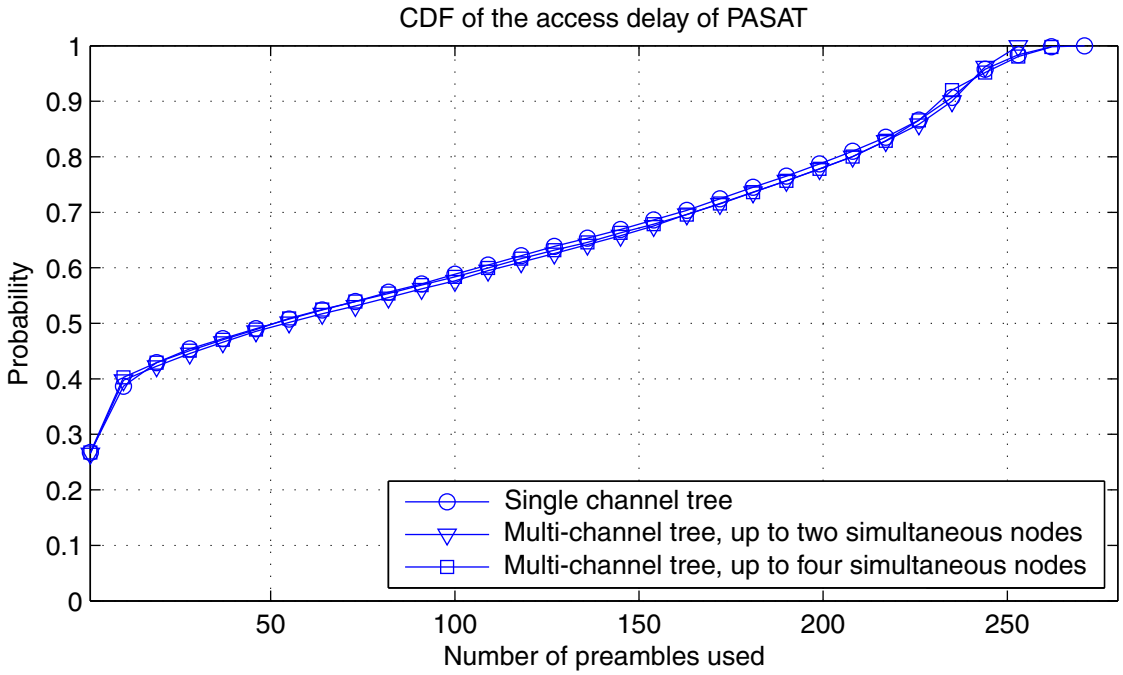


Figure 3.4: CDF of the access delay of the PASAT-Algorithm with 30000 contenders in 10 seconds

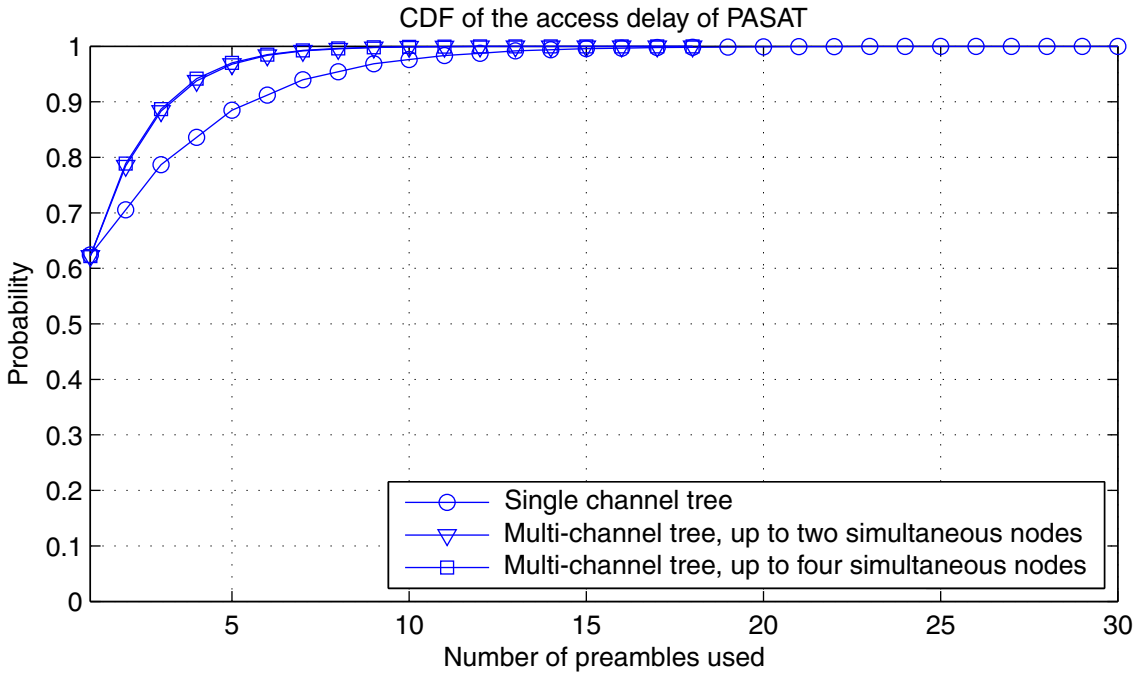


Figure 3.5: CDF of the access delay of the PASAT-Algorithm with 10000 contenders in 10 seconds

3000 contenders in 1 second is presented. Once again, multi-channel Tree Algorithms perform better than single channel TAs.

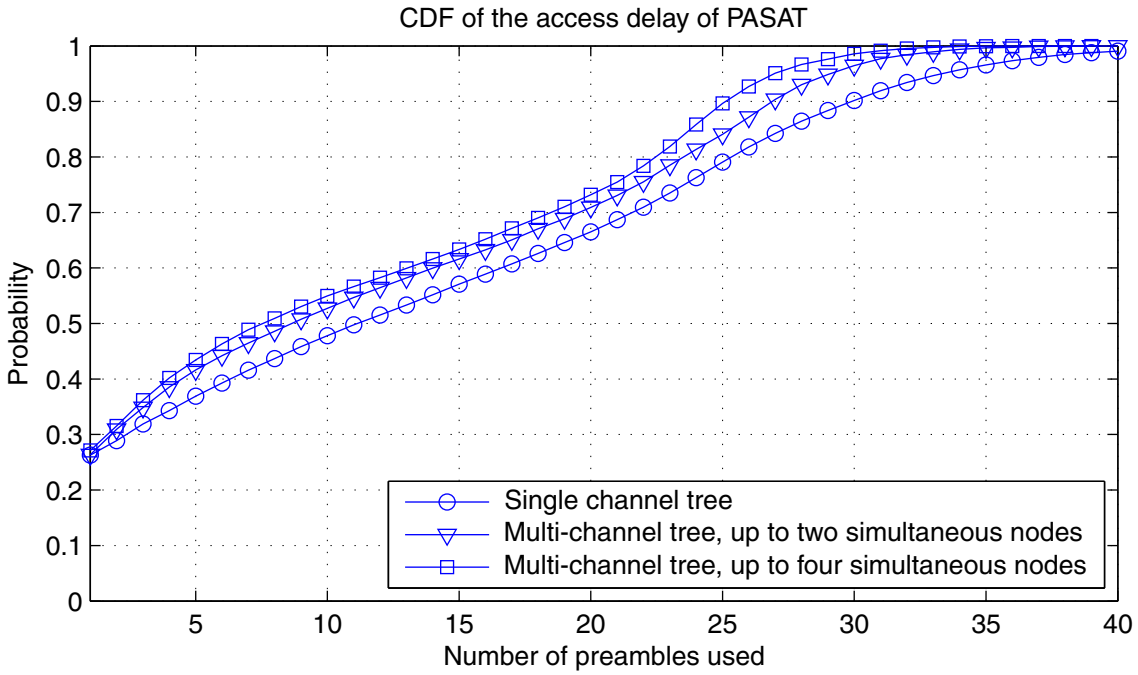


Figure 3.6: CDF of the access delay of the PASAT-Algorithm with 3000 contenders in 1 second

- Finally, we consider a short burst with a higher traffic density than in the first scenario. Namely, in Figure 3.7 the results for a burst of 600 contenders in 25 ms (a traffic density eight times higher than in the first scenario) is depicted. Even with such a high density, multi-channel Tree Algorithms still perform better.

From these simulation results, we can see that multi-channel Tree Algorithms perform better or as good as single channel Tree Algorithms in the considered scenarios. However, multi-channel implementations of Tree Algorithms have been barely studied in the literature, and hence their behavior in terms of access delay with respect to single channel Tree Algorithms is not known. If we acquired knowledge about the statistics of the access delay of multi-channel trees, we could use these algorithms in a guided manner, so that better results would be expected. Thus, in this thesis a comprehensive analytical study and simulations of multi-channel Tree Algorithms are presented to cover this lack of knowledge.

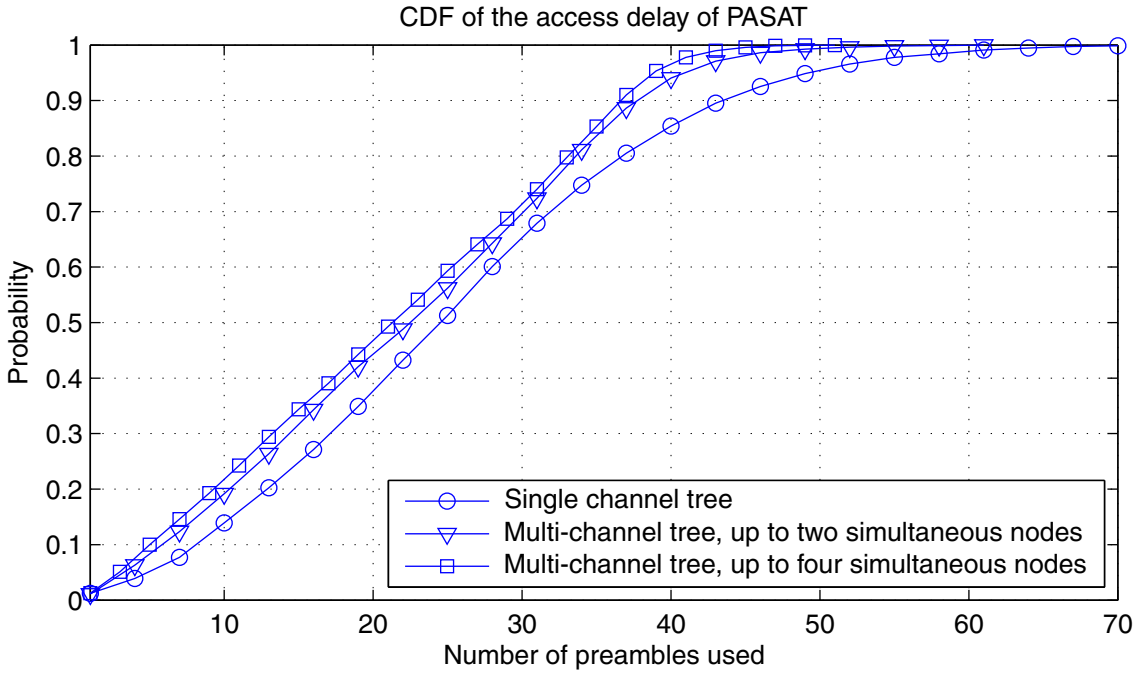


Figure 3.7: CDF of the access delay of the PASAT-Algorithm with 600 contenders in 25 milliseconds

Chapter 4

Modeling Multi-Channel Tree Algorithms

In order to properly adapt Tree Algorithms to the characteristics of the Random Access Channel of LTE, one should be able to predict the behavior of these algorithms in a multi-channel scenario. In this chapter, analytical expressions for the length of the tree and the access delay of multi-channel Tree Algorithms will be developed.

4.1 Notation and problem description

In this chapter, Tree Algorithms will be analyzed in an abstract manner, i.e. there will not be direct references to their final application, in order for the results to be used in any field other than the LTE RACH. Deterministic variables will be denoted with italic letters (such as N), whereas random variables will be denoted with calligraphic letters (such as $\mathcal{L}$). In Table 4.1 a summary of the most important variables that will be used throughout this chapter is presented.

The whole analysis of multi-channel Tree Algorithms will be performed assuming Binary Tree Algorithm and parallel traversing, as explained in section 2.3.1, unless explicitly stated. The goal of this analysis is to derive an analytical expression for the CDF of the access delay of a multi-channel Tree Algorithm. In order to do so, the CDF of the length of a multi-channel tree will be derived first, since it is required to compute the access delay. Before that, with the intention of introducing those techniques that will be applied to analyze multi-channel trees, the derivation of the length of the tree and the access delay of a single channel scenario are presented first.

Variable	Type	Definition
N	Deterministic value	Number of initial contenders
$\mathcal{L}_N$	Random variable	Length of the tree, in nodes
$\mathcal{D}_N$	Random variable	Access delay of serial traversing, in nodes
$\mathcal{T}_N$	Random variable	Length of the tree, in slots
$\mathcal{F}_N$	Random variable	Length of the tree, in contention frames
$\mathcal{X}_N$	Random variable	Number of collisions
$\mathcal{U}_N$	Random variable	Number of successful transmissions
$\mathcal{S}_N$	Random variable	Partial sum of the length of the tree, in slots
$\mathcal{K}_m$	Random variable	Number of contenders in the level m
$\mathcal{P}_x^k$	Random variable	Partition of k in x parts
Θ_N	Random variable	Access delay, in slots
Ω_N^m	Random variable	Transmission time in the level m, in slots

Table 4.1: Summary of relevant variables

4.2 Delay related statistics of single channel BTA

Before addressing the problem of obtaining an analytical expression for the CDF of the multi-channel access delay, the derivation of other delay related statistics of Tree Algorithms is presented. We will start with those of single channel trees, since they are needed to understand the analysis of multi-channel TAs. Indeed, those approaches for studying single channel TAs that are not based in recursive properties can be directly applied to study multi-channel TAs. On the contrary, those approaches based on recursive properties are not directly applicable to multi-channel trees, but sometimes the idea behind the analysis can be extended to a multi-channel environment.

In this section we will cover the derivation of theoretical expressions for the length of the tree and the access delay that any device experience. When using single channel Tree Algorithms, the terms *node* and *time slot* become synonymous, so that a complete analysis can be performed in terms of nodes. The reasoning to calculate the statistics of the access delay from parameters of the tree can be summarized as follows. First, the statistics about the length of the tree are required, since both variables are clearly related: the longer the tree, the higher the expected access delay. Once the length of the tree is characterized, the strategy to obtain the delay is to calculate the probabilities of every possible access delay within a tree of a given length, and then use the probability to have that given length to compute the probability of the access delay. Given this strategy, the statistics of the length of the tree will be obtained first, and then those of the access delay will be derived.

4.2.1 Length of the tree

The first important parameter that can be calculated for Tree Algorithms is the length of tree, i.e. the number of nodes that are needed to complete the tree. In the following, it will be denoted by $\mathcal{L}_N$, where N is the number of initial contenders.

The length of the tree is usually the most studied parameter of a TA, since it is used to compute the throughput η of the algorithm:

$$\eta = \frac{N}{\mathrm{E}\left[\mathcal{L}_N\right]}. \tag{4.1}$$

In words, the throughput is the ratio between the number of slots with successful transmission and the average number of slots used by the tree. Throughput is often used to compare different TAs, or even TAs with other algorithms like Slotted-ALOHA. Although this parameter cannot be directly translated into access delay, the relation between both is clear. When the throughput is high, one should expect low access delays and vice versa. Apart from that application, previous knowledge of the length of the tree is required to calculate the access delay, as it was mentioned before.

We will progressively derive the CDF of the length of a single channel tree in this section. First, only the average value will be computed. Finally, the approach used to obtain the average will be extended to compute the probability mass function (PMF) and hence the CDF.

Average value

In the literature, the derivation of the average and variance of the length of the tree is often addressed to characterize this random variable. Two different approaches that provide exact results are available: a recursive and a 'levelwise' approach. In addition, an approximate closed-form expression is also available in the state of the art. All these approaches will reviewed in the following.

The recursive approach, which is covered in [Mas14], takes advantage of the recursive nature of the tree. Indeed, each node in the tree can be seen as the root node of the subtree formed by its descendant nodes. Consequently, we can decompose every binary tree into two subtrees, whose root nodes will be the children of the original root node, as depicted in Figure 4.1.

Each of the two child nodes with i and $N - i$ contenders will be the origins of two new subtrees with lengths $\mathcal{L}_i$ and $\mathcal{L}_{N-i}$, respectively. As the new subtrees are not related, it is clear that these two random variables are independent. This an important property, since it can be exploited to obtain the probability-generating function (PGF), as it is explained later. This new random variables can related with the total length of the tree as follows:

$$\mathcal{L}_N = 1 + \mathcal{L}_i + \mathcal{L}_{N-i}. \tag{4.2}$$

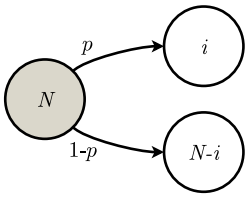


Figure 4.1: Decomposition of a binary tree

For the moment we are only interested in the average value, therefore we can focus on that.

$$L_N = \mathrm{E}\left[\mathcal{L}_N\right], \tag{4.3}$$

$$L_{N|i} = 1 + L_i + L_{N-i}, \tag{4.4}$$

where $L_{N|i}$ is the average tree length given a splitting of i and $N - i$ contenders. The probability of that splitting is equal to the probability that i out of N devices select the first node, which follows a binomial distribution:

$$P_N(i) = \binom{N}{i} p^i (1-p)^{N-i},$$

where p is the probability to choose the first node, i.e. the probability to choose '0'. In an unbiased binary tree, $p = 0.5$, so that both nodes are equally likely. It is known that the unbiased BTA performs optimally with respect to biased ($p \neq 0.5$) BTAs [AZKC$^+$08], therefore an unbiased tree will be assumed henceforth.

At this point, calculating the desired mean value for all $i \in [0, N]$ is a direct application of the law of total expectation:

$$L_N = \sum_{i=0}^{N} L_{N|i} \cdot P_N(i) \tag{4.5}$$

$$= 1 + \left(p^N + (1-p)^N\right) L_N + \sum_{i=1}^{N-1} \left(L_i + L_{N-i}\right) P_N(i). \tag{4.6}$$

After solving for L_N:

$$L_N = \frac{1 + \sum_{i=1}^{N-1} \left(L_i + L_{N-i}\right) P_N(i)}{1 - p^N - (1-p)^N}. \tag{4.7}$$

Since expression 4.7 is recursive, initial values are required in order to complete the definition. Those values are $L_0 = L_1 = 1$, as the length of the tree when the number of contenders is 0 or 1 is just one node.

The derivation of expression 4.7 is a good exercise to understand the recursive nature of tree algorithms, and the underlying idea can be applied to obtain other parameters.

Nonetheless, this is not the unique manner to derive an expression for the average length of the tree. In [KG85], the authors presented an alternative approach that is based on examining the tree level by level, what turns out to be very useful when studying multi-channel TAs, as it is explained later.

Under this approach, firstly we need to decompose our original variable $\mathcal{L}_N$ as follows:

$$\mathcal{L}_N = 1 + \sum_{m=1}^{\infty} \mathcal{L}_N^m, \tag{4.8}$$

where $\mathcal{L}_N^m$ is the number of nodes in the level m. Intuitively, this method to obtain $\mathcal{L}_N$ relies on computing the number of nodes that are in every level and then add them to obtain the final result. Regarding average, the expression follows the same structure, owing to the linear properties of the expectation operation:

$$L_N = 1 + \sum_{m=1}^{\infty} L_N^m, \tag{4.9}$$

where L_N^m is the average number of nodes in the level m. At this point, it is important to notice that the number of nodes in certain level it is equal to the number of nodes with collisions in the previous level multiplied by 2. This property of the tree is illustrated in Figure 4.2. Therefore, we only need to compute the average number of nodes with collisions in each level. In order to do that, we first calculate the probability C that one node at level m contains a collision, given N initial contenders:

$$C(N,m) \;=\; 1 - \left(1 - \frac{1}{2^m}\right)^N - N\frac{1}{2^m}\left(1 - \frac{1}{2^m}\right)^{N-1}. \tag{4.10}$$

This expression for C is obtained after realizing that the event of one node being occupied by one device is a Bernoulli trial with $p = \frac{1}{2}$. Knowing that probability, the average number of collisions X_N^m in the level m is simply:

$$X_N^m = 2^m C(N,m), \tag{4.11}$$

which is the total number of nodes multiplied by the probability for one node to contain a collision. The last step is to convert collisions into number of nodes, as follows:

$$L_N^m = 2X_N^{m-1} = 2^m C(N, m-1). \tag{4.12}$$

As it was mentioned before, this 'level-wise' approach is much more applicable to multi-channel Tree Algorithms than the previous approach, since recursion properties are in general lost when nodes are grouped.

However, none of these previous approaches allows us to see clearly how the average length of the tree increases when the number of contender increases. It would be desirable to

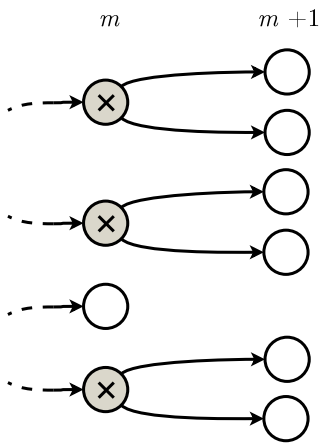


Figure 4.2: After every collision, two new nodes are generated in the tree.

have a closed-form non-recursive expression in order to distinguish the behavior of L_N with respect to N at a glance. Regarding that matter, in [JdJ00] the authors obtained the following closed-form approximation of the average length of the tree, as a result of Fourier analysis of equation 4.9:

$$L_N \simeq \frac{2N}{\ln 2}. \tag{4.13}$$

Besides the average, the variance of $\mathcal{L}_N$ can be also derived using similar arguments as those of recursive, level-wise and approaches, in [Mas14], [KG85] and [JdJ00], respectively. The derivation of the variance will not be exposed here, since its value can be computed from the whole PMF that will be derived in the following.

Probability-generating and probability mass functions

As it was explained in the beginning of this section, we need complete knowledge about the length of the tree to obtain the access delay, so that the average value is not enough. In order to completely characterize $\mathcal{L}_N$ we need to find its probability mass function (PMF), which will be denoted by $p_{\mathcal{L}_N}(l)$. This PMF is defined as:

$$\Pr\left[\mathcal{L}_N = l\right] = p_{\mathcal{L}_N}(l). \tag{4.14}$$

Despite of the multiple manners to obtain the average, deriving an analytical expression for $p_{\mathcal{L}_N}(l)$ is not trivial. Instead, it is easier to compute its probability-generating function (PGF) $G_{\mathcal{L}_N}(z)$, defined as the z-transform of the PMF:

$$G_{\mathcal{L}_N}(z) = \mathcal{Z}\left\{p_{\mathcal{L}_N}(l)\right\}. \tag{4.15}$$

The key property of the PGF is the fact that the PGF of the sum of a set of independent random variables is equal to the product of their PGFs. If we apply this property to

expression 4.2, it yields the following relation:

$$G_{\mathcal{L}_N}(z) = z^{-1}\sum_{i=0}^{N} P_N(i)G_{\mathcal{L}_i}(z)G_{\mathcal{L}_{N-i}}(z) \tag{4.16}$$

$$= \left(p^N + (1-p)^N\right) z^{-2}G_{\mathcal{L}_N}(z) + z^{-1}\sum_{i=1}^{N-1} P_N(i)G_{\mathcal{L}_i}(z)G_{\mathcal{L}_{N-i}}(z). \tag{4.17}$$

Finally, we solve the for $G_{\mathcal{L}_N}(z)$ to obtain the final recursive relation:

$$G_{\mathcal{L}_N}(z) = \frac{z^{-1}\sum_{i=1}^{N-1} P_N(i)G_{\mathcal{L}_i}(z)G_{\mathcal{L}_{N-i}}(z)}{1-\left(p^N + (1-p)^N\right) z^{-2}}. \tag{4.18}$$

Expression 4.18 provides us with a way to obtain an analytical expression of the PGF of the length of a tree. In order to obtain the PMF, a inverse z-transformation has to be applied to the result. However, expression 4.18 produces complicated PGFs even for a low number of initial contenders. In [MP93], a list of these PGFs up to $N = 5$ is presented, and it is clear that z-inverting those expressions to obtain a PMF is not a trivial process. Even if symbolic computation software is in place, the inversion of these PGFs might not be feasible or extremely time consuming if the number of contenders is high, since the number of terms in the PGF seems to grow rapidly with the number of contenders.

In order to overcome this problem, a numerical alternative based on replacing z-transforms with FFTs is suggested here. Unavoidably, some precision is going to be lost with respect to an exact closed-form expression, since the DFT works with a limited number of points. However, this loss can be rather small if the number of points used by the FFT is correctly chosen. For this case, we can define:

$$G^F_{\mathcal{L}_N}(\omega) = \mathcal{F}\{p_{\mathcal{L}_N}(l)\}. \tag{4.19}$$

The same expressions as for $G_{\mathcal{L}_N}(z)$ hold for $G^F_{\mathcal{L}_N}(\omega)$, what allows us to compute a very good approximation for the PMF of the length of the tree in little time.

In the Figure 4.3 some PMFs of the length of the tree for several values of N are shown. All of them were computed using the FFT approximation exclusively, with the exception of $N = 2$, which was also computed symbolically in order to provide a comparison between both techniques. Two interesting remarks can be extracted from these pictures. In the first place, no even-valued lengths are possible, since nodes are generated in pairs starting from the root node. In second place, the envelope of the PMF resembles a Gaussian distribution as N increases, as a consequence of the central limit theorem.

4.2.2 Access delay

Now we address the problem of obtaining a more interesting metric, the delay experienced by a contender from the initial collision until its successful transmission. This delay is

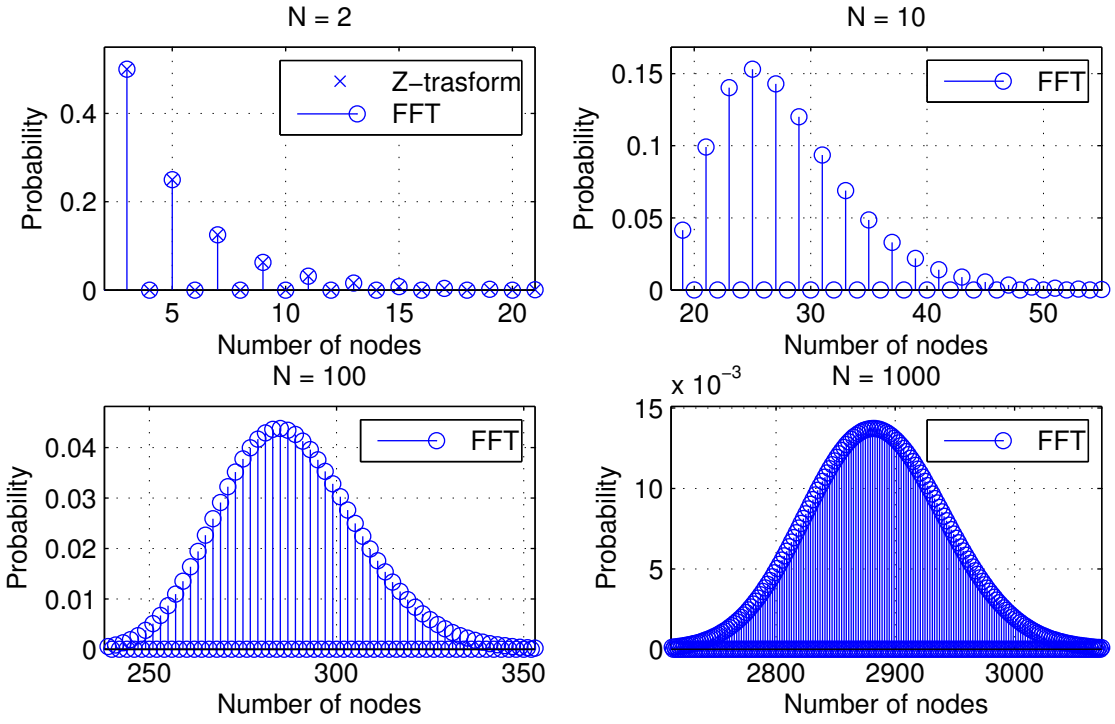


Figure 4.3: Example results of the PMF of the length of the tree

measured in time slots, which are related with nodes in the tree through an one-to-one correspondence. This means that each node in a single channel tree will be translated into a slot in the time domain, therefore this two terms will be interchangeable.

The access delay depends on the way that the tree is traversed, since this way affects the order of transmission of all nodes. In this subsection, only serial traversing will be tackled, since it is the most commonly used in single channel Tree Algorithms. The access delay experienced when applying parallel traversing will be addressed in the next section, as its analysis can be merged with that of multi-channel Tree Algorithms.

As in the analysis of the length of the tree, the derivation of the average and the probability mass function of the access delay will be tackled separately.

Average value

The access delay experienced by a device can be modeled as a random variable, which will be denoted as $\mathcal{D}_N$, given N initial contenders.

In contrast to the average value of the length of the tree, only a recursive expression of the average access delay can be found in the state of the art. In [MP93], the reasoning to obtain such recursive expression is presented, although the final expression is not derived. In this subsection, the resulting expression will be indeed presented for the sake of completeness.

In order to understand the procedure for computing the average access delay, we recover in Figure 4.4 the picture describing the generic splitting, but this time as it is seen from a single contender. That means that only $N-1$ out of N contenders will be considered.

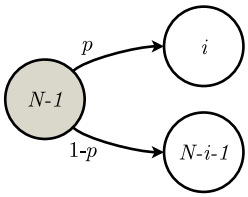


Figure 4.4: Decomposition of a binary tree as seen from a contender

This contender has two possibilities while being in the first node:

1. With probability p, the device will choose the first slot. This slot is the initial node of a new subtree, which will be resolved in the first place. Therefore, the total delay experienced by a device in this case will be the delay experienced in the subtree plus one slot:
$$\mathcal{D}_N = 1 + \mathcal{D}_{i+1}. \tag{4.20}$$

2. With probability $1-p$, the device will choose the second slot. In this case, the device will have to wait until the first subtree is totally resolved, and then its subtree will be

resolved. Therefore, it will experience a delay equal to the length of the first subtree, plus the delay experienced in the second subtree, plus one slot:

$$\mathcal{D}_N = 1 + \mathcal{L}_i + \mathcal{D}_{N-i}. \tag{4.21}$$

Hence, the random variable $\mathcal{D}_N$ can be expressed as:

$$\mathcal{D}_N = \begin{cases} 1 + \mathcal{D}_{i+1} & \text{with prob. } p. \\ 1 + \mathcal{L}_i + \mathcal{D}_{N-i} & \text{with prob. } 1-p. \end{cases} \tag{4.22}$$

Let us focus now on the mean value of $\mathcal{D}_N$:

$$D_N = \mathrm{E}\left[\mathcal{D}_N\right]. \tag{4.23}$$

Given some splitting of size i with probability $P_{N-1}(i)$, the mean $D_{N|i}$ can be expressed as:

$$D_{N|i} = 1 + pD_{i+1} + (1-p)(L_i + D_{N-i}). \tag{4.24}$$

At this point, we apply the law of total expectation in the same manner as while studying the length of the tree:

$$D_N \;=\; 1 + \sum_{i=0}^{N-1} \Big(pD_{i+1} + (1-p)(L_i + D_{N-i}) \Big) P_{N-1}(i). \tag{4.25}$$

After some manipulation, the resulting recursive expression is:

$$\begin{aligned} D_N = \; & \frac{1 + p(1-p)^{N-1} + (1-p)^N + p^{N-1}(1-p)(1+L_{N-1})}{1-(1-p)^N - p^N} \\ & + \frac{\displaystyle\sum_{i=1}^{N-2} \Big(pD_{i+1} + (1-p)(L_i + D_{N-i}) \Big) P_{N-1}(i)}{1-(1-p)^N - p^N}. \end{aligned} \tag{4.26}$$

As opposed to the analysis of the length of the tree, there is not a closed-form expression for the average access delay that we can use to predict its behavior. Instead, we can simply compute values out of expression 4.26 and plot them, as it is shown in Figure 4.5.

From Figure 4.5 we can deduce a linear relation between N and D_N, as it happened with the length of the tree. The ratio of that linear relation is unknown, but based on empirical results we might conjecture that:

$$\frac{D_N}{N} = \frac{1}{\ln 2} \approx 1.44. \tag{4.27}$$

If expression 4.27 is true, the average access delay would be half of the average length of the tree when N is large, according to equation 4.13.

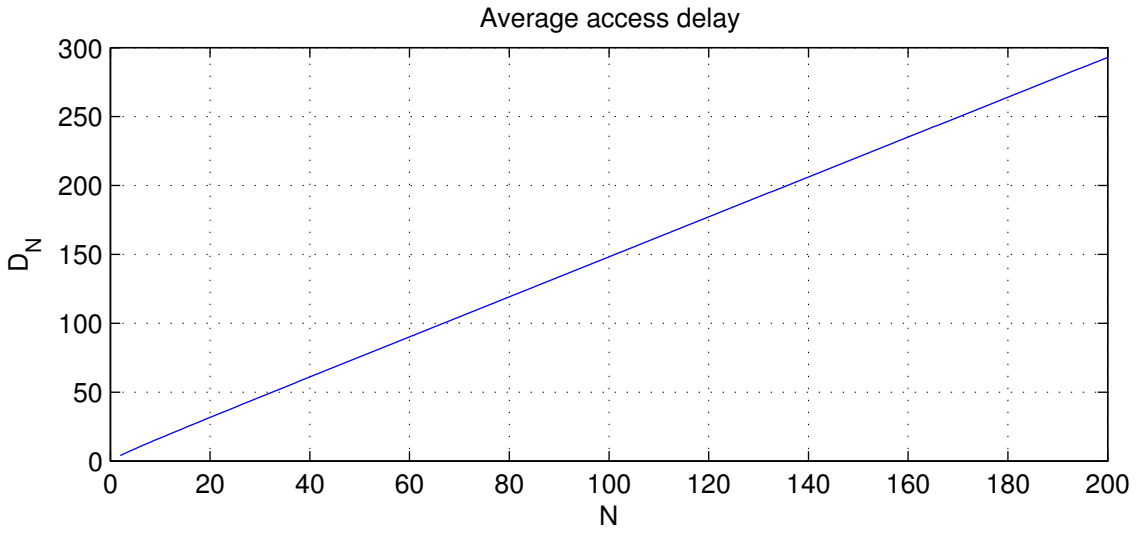


Figure 4.5: Behavior of the average access delay

Probability-generating and probability mass functions

The same ideas for calculating the PGF and PMF of the length of the tree can be applied to the access delay. We define the PMF of the access delay as:

$$\Pr\left[\mathcal{D}_N = d\right] = p_{\mathcal{D}_N}(d). \tag{4.28}$$

And the PGF as the z-transform of the above:

$$G_{\mathcal{D}_N}(z) = \mathcal{Z}\left\{p_{\mathcal{D}_N}(d)\right\}. \tag{4.29}$$

Now, we extract the PGF from equation 4.22:

$$G_{\mathcal{D}_N}(z) = z^{-1}\left(\sum_{i=0}^{N-1}\Big(p\cdot G_{\mathcal{D}_{i+1}}(z) + (1-p)G_{\mathcal{D}_{N-i}}(z)G_{\mathcal{L}_i}(z)\Big)P_{N-1}(i)\right). \tag{4.30}$$

After solving the for $G_{\mathcal{D}_N}(z)$ we obtain our final result:

$$\begin{aligned} G_{\mathcal{D}_N}(z) = {} & \frac{\left(\sum_{i=1}^{N-2}\Big(p\cdot G_{\mathcal{D}_{i+1}}(z) + (1-p)G_{\mathcal{D}_{N-i}}(z)G_{\mathcal{L}_i}(z)\Big)P_{N-1}(i)\right)z^{-1}}{1-p^N z^{-1}-(1-p)^N z^{-2}} \\ & + \frac{\Big(p^{N-1}(1-p)\mathfrak{L}_{N-1}(z) + p(1-p)^{N-1}\Big)z^{-2}}{1-p^N z^{-1}-(1-p)^N z^{-2}}. \end{aligned} \tag{4.31}$$

By using FFTs instead of Z-transforms, good approximations of the PGF and hence the PMF can be again obtained. We define:

$$G^F_{\mathcal{L}_N}(\omega) = \mathcal{F}\{p_{\mathcal{L}_N}(l)\}. \tag{4.32}$$

In the Figure 4.6, some PMFs of the access delay for different values of N are shown. We notice an interesting evolution of the shape of the PMFs as N increases. Instead of converging to a Gaussian as we might expect, PMFs resemble rather to an uniform distribution with a smooth tail for the right end. However, the explanation of this interesting shape is out of the scope of this thesis.

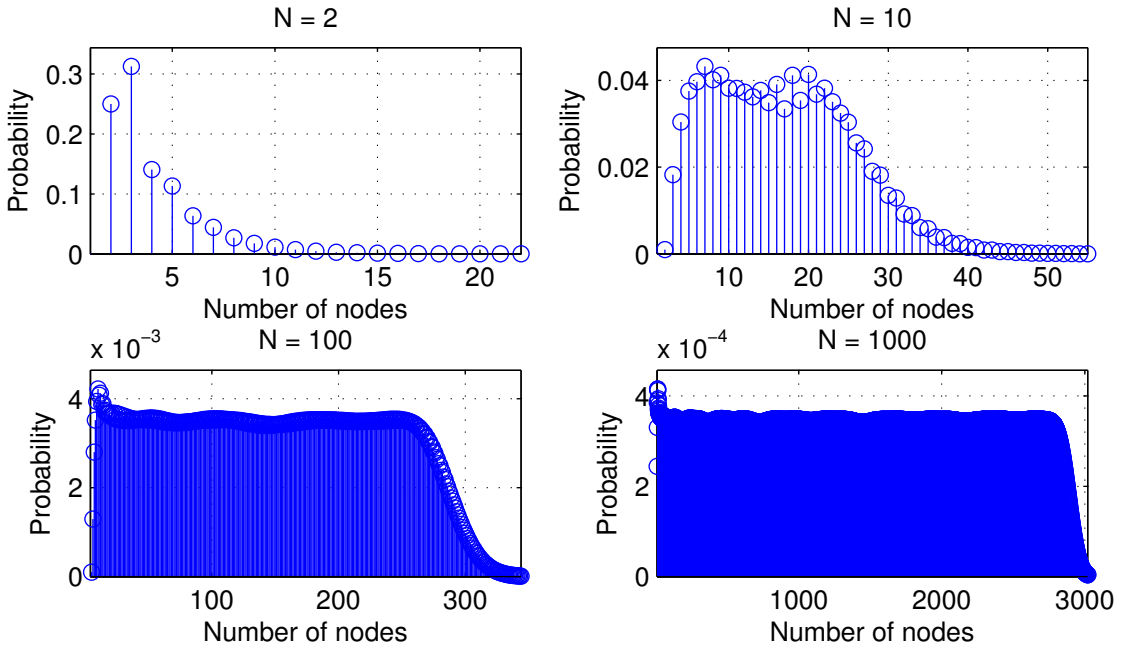


Figure 4.6: Example results of the PMF of the access delay of a single channel Tree Algorithm

4.3 Delay related parameters of multi-channel BTA

The use of orthogonal preambles in the Random Access procedure of LTE provides the opportunity to transmit several nodes of the tree at the same time, i.e. in the same time slot. As it was mentioned before, the capacity of every slot is denoted by G, which is the maximum number of contention frames (pair of nodes) that fit in one slot. The transmission of several nodes simultaneously should decrease both the length of the tree and the access delay, therefore it is an interesting scenario to study.

One might think at first that increasing G should decrease the access delay up to an arbitrarily low value. For instance, if the number of nodes that can be transmitted simultaneously is actually greater than the length of a given tree, it might be thought that that tree would be resolved in just one slot. However, this is not true, since a Tree Algorithm cannot be constructed in a single step. Instead, a tree needs the information about past

collisions that is contained in the feedback in order to evolve. Therefore, the application of Tree Algorithms on multi-channel scenarios has a bounded performance, due to their progressive nature.

Without limiting the foregoing, the access delay experienced by a contender within a tree can be indeed reduced in a multi-channel scenario. Several nodes may be transmitted together as long as that does not affect to the feedback. In other words, the limitation is simply that any node must be transmitted after all of its ascendant nodes. There are several options for grouping nodes into slots that fulfill such limitation. In the Figure 4.7, different options are depicted. Out of these options, (a) and (b) are indeed feasible, whereas (c) is not, since parent and child nodes are been transmitted together.

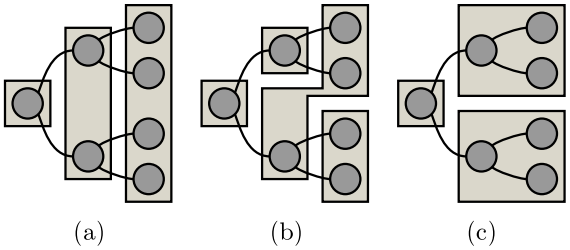


Figure 4.7: Different options for grouping nodes into slots. (a) and (b) are possible, (c) is not feasible.

The selection of a manner to group nodes into slots affects the order of the transmission of those nodes, therefore it influences the length of the tree (in terms of slots) and the access delay. This implies that the analysis have to be performed after selecting a grouping technique in order to obtain consistent results. In the present analysis, a level-wise grouping is assumed, that is, only those nodes belonging to the same level can be transmitted in the same slot. This grouping technique is the one depicted in the tree (a) of Figure 4.7. There is a twofold explanation for this selection: grouping is simple to perform and level-wise approaches can be hence applied to analyze the tree.

An example of level-wise grouping is shown in the Figure 4.8, with a maximum of four nodes per slot. From this figure, an important, aforementioned consequence of the grouping of the nodes in a multi-channel scenario can be detected: the algorithm loses the tree structure, what prevents us to use recursive approaches as for single channel scenarios.

In this section, the derivation of the statistics of the length of the tree will be presented first. These statistics are required to obtain those of the access delay, which will be addressed next.

4.3.1 Length of the tree

As it was mentioned before, the grouping of nodes into slots erases the recursive properties of Tree Algorithms, so that the recursive approach to obtain the length of the tree is not

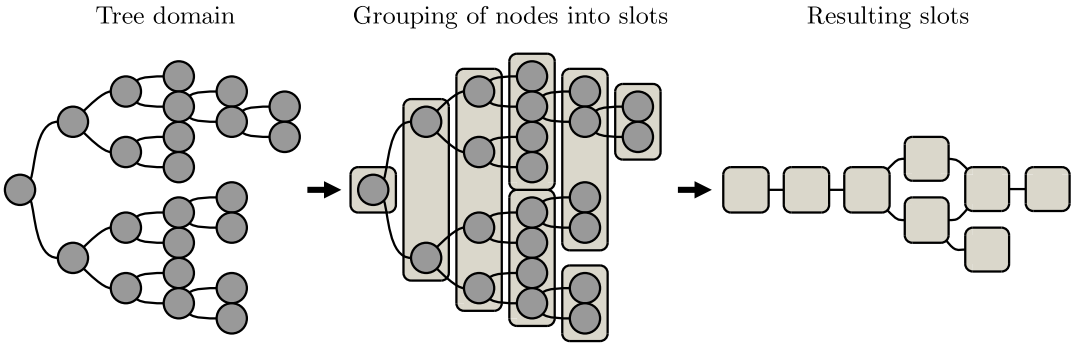


Figure 4.8: Evolution to from a single channel to a multi-channel tree

an option anymore. On the contrary, the level-wise approach can still be applied, and it will be the basis of the following analysis.

We will denote by $\mathcal{T}_N$ the random variable modeling the number of slots needed to complete a multi-channel tree. We use a different variable with respect to the single channel case, since $\mathcal{L}_N$ can still be used to model the number of nodes in a the tree.

If we define $\mathcal{T}_N^m$ as the number of slots in the level m, the total number of slots in the tree can be expressed as:

$$\mathcal{T}_N = 1 + \sum_{m=1}^{\infty} \mathcal{T}_N^m, \tag{4.33}$$

where $\mathcal{T}_N^m$ is the number of slots in the level m. The number of slots in the level m can be related with the number of nodes in such level as follows:

$$\mathcal{T}_N^m = \left\lceil \frac{\mathcal{L}_N^m}{2G} \right\rceil, \tag{4.34}$$

where G is the number of contention frames per slot, $\mathcal{L}_N^m$ is the number of nodes in the level m and $\lceil \cdot \rceil$ is the ceiling function. A graphical reason of this equation can be extracted from Figure 4.8, in which $G = 2$. If we now define $\mathcal{F}_N^m$ as the number of contention frames in the level m, expression 4.34 can be rewritten as:

$$\mathcal{T}_N^m = \left\lceil \frac{\mathcal{F}_N^m}{G} \right\rceil. \tag{4.35}$$

Furthermore, since every collision in level $m-1$ produces a new contention frame in level m, if we define $\mathcal{X}_N^m$ as the number of collisions in the level m, we can rewrite expression 4.35 as:

$$\mathcal{T}_N^m = \left\lceil \frac{\mathcal{X}_N^{m-1}}{G} \right\rceil. \tag{4.36}$$

After combining expressions 4.33 and 4.36, we obtain:

$$\mathcal{T}_N = 1 + \sum_{m=1}^{\infty} \left\lceil \frac{\mathcal{X}_N^{m-1}}{G} \right\rceil. \tag{4.37}$$

Equation 4.37 is the starting point for obtaining the average value and the PMF of $\mathcal{T}_N$, as it expresses this variable as a function of the number of collisions occurred in the tree. The number of collisions is often more tractable than nodes or slots, since it can be obtained from combinatorial analysis.

Average value

At this point we are ready to derive an expression of the average length of the tree in slots, which we will denote by T_N:

$$T_N = \mathrm{E}\left\{\mathcal{T}_{\mathcal{N}}\right\}. \tag{4.38}$$

As commented before, the analysis of multi-channel Tree Algorithms is barely covered in the literature. However, in [nSP14], an expression for computing the average length of a multi-channel Tree Algorithm can be found:

$$T_N = 1 + \sum_{m=1}^{\infty} \left\lceil \frac{\mathrm{E}\left\{\mathcal{X}_N^{m-1}\right\}}{G} \right\rceil. \tag{4.39}$$

Although expression 4.39 is presented as exact by the authors, in general equation 4.39 does not hold. We can prove this by applying the expectation operator on equation 4.37, as follows:

$$\begin{aligned} \mathrm{E}\left\{\mathcal{T}_{\mathcal{N}}\right\} &= \mathrm{E}\left\{1 + \sum_{m=1}^{\infty} \left\lceil \frac{\mathcal{X}_N^{m-1}}{G} \right\rceil \right\} && (4.40) \\ &= 1 + \sum_{m=1}^{\infty} \mathrm{E}\left\{ \left\lceil \frac{\mathcal{X}_N^{m-1}}{G} \right\rceil \right\}. && (4.41) \end{aligned}$$

Since the ceiling function is not a linear operation, in general $\mathrm{E}\left\{\lceil\cdot\rceil\right\} \neq \lceil \mathrm{E}\left\{\cdot\right\} \rceil$, therefore expression 4.39 is not exact in general. As a consequence it either might be only used as an approximation, or disregarded.

Nevertheless, an exact expression can be indeed derived. Our starting point will be to decompose the ceiling function of a fraction into a set of linear operations:

$$\left\lceil \frac{\mathcal{X}_N^{m-1}}{G} \right\rceil = \begin{cases} \frac{\mathcal{X}_N^{m-1}}{G} & \text{if } \mathcal{X}_N^{m-1} \bmod G = 0. \\ \frac{\mathcal{X}_N^{m-1}+G-1}{G} & \text{if } \mathcal{X}_N^{m-1} \bmod G = 1. \\ \vdots & \vdots \\ \frac{\mathcal{X}_N^{m-1}+1}{G} & \text{if } \mathcal{X}_N^{m-1} \bmod G = G-1. \end{cases} \tag{4.42}$$

Now, we can simply apply the law of total expectation:

$$\begin{aligned}\mathrm{E}\left\{\left\lceil\frac{\mathcal{X}_N^{m-1}}{G}\right\rceil\right\} &= \frac{1}{G}\mathrm{E}\{\mathcal{X}_N^{m-1}\}\Pr\{\mathcal{X}_N^{m-1} \bmod G = 0\} + \dots \\ &\qquad \dots + \frac{1}{G}\mathrm{E}\{\mathcal{X}_N^{m-1}+1\}\Pr\{\mathcal{X}_N^{m-1} \bmod G = G-1\} \qquad (4.43)\\ &= \frac{1}{G}\left(\mathrm{E}\{\mathcal{X}_N^{m-1}\} + \sum_{k=1}^{G-1}(G-k)\cdot\sum_{j=0}^{\infty}\Pr\{\mathcal{X}_N^{m-1} = G\cdot j + k\}\right). \qquad (4.44)\end{aligned}$$

At this point, we are left with the problem of computing $\Pr\{\mathcal{X}_N^{m-1} = x_{m-1}\}$, i.e. the probability to have x_{m-1} collisions in the level $m-1$ with N initial contenders. In order to solve this problem, we can formulate it as a bins-and-balls problem.

We know that the maximum number of nodes in level m of the tree is 2^m, since the number of nodes can, at most, duplicate from level $m-1$. Given this, we can index every node in the level m with a number i_m ranging from 0 to 2^m. Now, we can think about what needs to occur for one device to be in the node i_m. For this, two conditions are required:

1. The device must not have transmitted successfully before level m.
2. The sequence of decisions $\{\varsigma_1, \varsigma_2, \dots, \varsigma_m\}$ performed by the device along the tree must be such that $(\varsigma_1\varsigma_2\dots\varsigma_m)_2 = i_m$. For instance, to lie in the node 35 of the level 6, the sequence of decisions needs to be $\{1,0,0,0,1,1\}$ since $(100011)_2 = 35$.

Since we are only aiming at computing the probability of the number of collisions, we are able to neglect the first condition. The reason for this is simple: in order for one device to collide at level m, it must have also collided in levels $1,\dots,m-1$. Therefore, only the second condition applies.

If we look carefully into this second condition, we realize that generating any sequence $\{\varsigma_1, \varsigma_2, \dots, \varsigma_m\}$ is equivalent to draw the whole number $(\varsigma_1\varsigma_2\dots\varsigma_m)_2$ out of 2^m possibilities in one go. Therefore, we can ignore the whole tree and see the problem as if N contenders (or balls) were randomly distributed over 2^m available nodes (or bins). In this situation, we just need to compute the probability of having x_{m-1} bins with more than one ball.

Solutions to bins-and-balls problems are rather abundant in the literature. For instance, in [WBC15] the authors model the collisions of MTC devices when choosing preambles as a bins-and-balls problem with some interesting results, which will be adapted to our purpose.

Let us denote by $\mathcal{U}_N^m$ the number of nodes with successful transmissions in the level m. In the following, this will be equivalent the number of bins containing only one ball. In, [WBC15], a expression for computing the PMF of such random variable is presented:

$$\Pr\left[\mathcal{U}_N^m = u\right] = p_{\mathcal{U}_N^m}(u) = \sum_{i=u}^{\min\left(\left\lfloor\frac{N+u}{2}\right\rfloor, R\right)} \frac{\Psi_{i,i-u}^{N}\binom{R}{i}i!}{R^N}, \qquad (4.45)$$

where $R = 2^m$ is the number of 'bins' and $\Psi^m_{i,j}$ is a custom extension of the Stirling numbers of the second kind introduced by the authors. Let us tear apart expression 4.45 to fully understand it:

- The term $\Psi^N_{i,i-u}$ is the number of ways to partition N balls into i bins, such that u of them contains only one bin. In general, $\Psi^N_{i,j}$ is defined as the number of ways to partition $N \geq 1$ balls into $i \geq 1$ subsets, $j \in (0, i)$ of which contain more than one ball. These numbers are computed through the following recursion:

$$\Psi^N_{i,j} = j\Psi^{N-1}_{i,j} + (i - j + 1)\,\Psi^{N-1}_{i,j-1} + \Psi^{N-1}_{i-1,j}, \tag{4.46}$$

  with initial conditions, $\Psi^N_{N,0} = 1$, $\Psi^1_{1,0} = 1$, $\Psi^1_{1,1} = 0$ and $\Psi^{N>1}_{1,1} = 1$.

- The term $\binom{R}{i}$ is the number of ways to choose exactly i bins out of a total of R bins.
- The term $i!$ is the number of ways to arrange i used bins.
- The term R^N is the total number of possible combinations resulting from the random distribution of N balls over R bins.

Nevertheless, the number of successful transmissions $\mathcal{U}^m_N$ is not the variable whose PMF we want to obtain, but the number of collisions $\mathcal{X}^m_N$, i.e. the number of bins with more than one ball. There is not a direct translation between these variables, but we can easily adapt expression 4.45 to obtain the PMF of $\mathcal{X}^m_N$. In order to do that, we have to make the following modifications:

- We need to know the number of ways to partition N balls into i bins, such that x_m of them have more than one ball. That is simply Ψ^N_{i,x_m}, according to the definition of these numbers.
- The number of used bins will range from the the number of collisions x_m to the lowest of either the total number of bins R or $N - x_m$ bins. This last bound is derived from the fact that at least $2x_m$ balls are needed to generate x_m collisions. The remaining balls $N - 2x_m$ occupy one bin each, therefore the maximum number of used bins is $x_m + N - 2x_m = N - x_m$.

Once these modifications are applied, we obtain the desired PMF of $\mathcal{X}^m_N$:

$$\Pr\left[\mathcal{X}^m_N = x_m\right] = p_{\mathcal{X}^m_N}(x_m) = \sum_{i=x_m}^{\min(N-x_m,R)} \frac{\Psi^N_{i,x_m}\binom{R}{i} i!}{R^N}. \tag{4.47}$$

Finally, combining expressions 4.41, 4.44 and 4.47 we can compute the actual value of T_N.

However, there is a problem when computing values for $p_{\mathcal{X}^m_N}(x_m)$. As N or R increase, terms Ψ^N_{i,x_m}, $\binom{R}{i}$, $i!$ and R^N quickly get very large, so that the value of the fraction $\frac{\Psi^N_{i,x_m}\binom{R}{i} i!}{R^N}$ rapidly loses precision. In order to overcome this, a recursive computation of this value is

proposed. Let us denote this fraction by $\xi_{i,x_m}^{R,N}$:

$$\xi_{i,x_m}^{R,N} \triangleq \frac{\Psi_{i,x_m}^{N}\binom{R}{i}i!}{R^N}. \tag{4.48}$$

Every term of $\xi_{i,x_m}^{R,N}$ can be written in a recursive manner:

$$i! = i \cdot (i-1)!, \tag{4.49}$$

$$\binom{R}{i} = \left(\frac{R+1}{i} - 1\right) \cdot \binom{R}{i-1}, \tag{4.50}$$

$$R^N = R \cdot R^{N-1}. \tag{4.51}$$

After combining expressions 4.49, 4.50, 4.51 and 4.46, we obtain the following recursive expression for $\xi_{i,x_m}^{R,N}$, which is much more computationally suitable:

$$\xi_{i,x_m}^{R,N} = \frac{1}{R}\left[x_m \xi_{i,x_m}^{R,N-1} + (i - x_m + 1)\xi_{i,x_m-1}^{R,N-1} + (R - i + 1)\xi_{i-1,x_m}^{R,N-1}\right]. \tag{4.52}$$

More details about the derivation of this recursion can be found in the Appendix A. With this newly defined recursion, the PMF of $\mathcal{X}_N^m$ can be written as:

$$p_{\mathcal{X}_N^m}(x_m) = \sum_{i=x_m}^{\min(N-x_m,R)} \xi_{i,x_m}^{R,N}. \tag{4.53}$$

At last, expressions 4.41, 4.44, 4.47 and 4.48 can be merged into the final expression of the average of the length of the tree:

$$T_N = 1 + \frac{1}{G}\sum_{m=1}^{\infty}\left(\mathrm{E}\{\mathcal{X}_N^{m-1}\} + \sum_{k=1}^{G-1}(G-k)\cdot\sum_{j=0}^{\infty} p_{\mathcal{X}_N^{m-1}}(G\cdot j + k)\right). \tag{4.54}$$

Some computed results of the average length of a multi-channel tree are shown in the Figure 4.9, which also depicts the average length of a single channel tree for comparison. It can be observed from this figure that the higher G the lower average length for the same number of contenders, as expected.

At this point, one might think that the average length for $G = 1$ should be close to half the length of the single channel tree. In general, the average length T_N for a given G should be ideally close to $\frac{L_N}{2G}$. In order to check this conjecture, we can use equation 4.11 to rewrite 4.53 as follows:

$$\begin{aligned} T_N &= 1 + \frac{1}{G}\sum_{m=1}^{\infty}\mathrm{E}\{\mathcal{X}_N^{m-1}\} + \frac{1}{G}\sum_{m=1}^{\infty}\left(\sum_{k=1}^{G-1}(G-k)\cdot\sum_{j=0}^{\infty} p_{\mathcal{X}_N^{m-1}}(G\cdot j + k)\right) \\ &= \frac{L_N}{2G} + \frac{G-1}{G} + \frac{1}{G}\sum_{m=1}^{\infty}\left(\sum_{k=1}^{G-1}(G-k)\cdot\sum_{j=0}^{\infty} p_{\mathcal{X}_N^{m-1}}(G\cdot j + k)\right). \end{aligned} \tag{4.55}$$

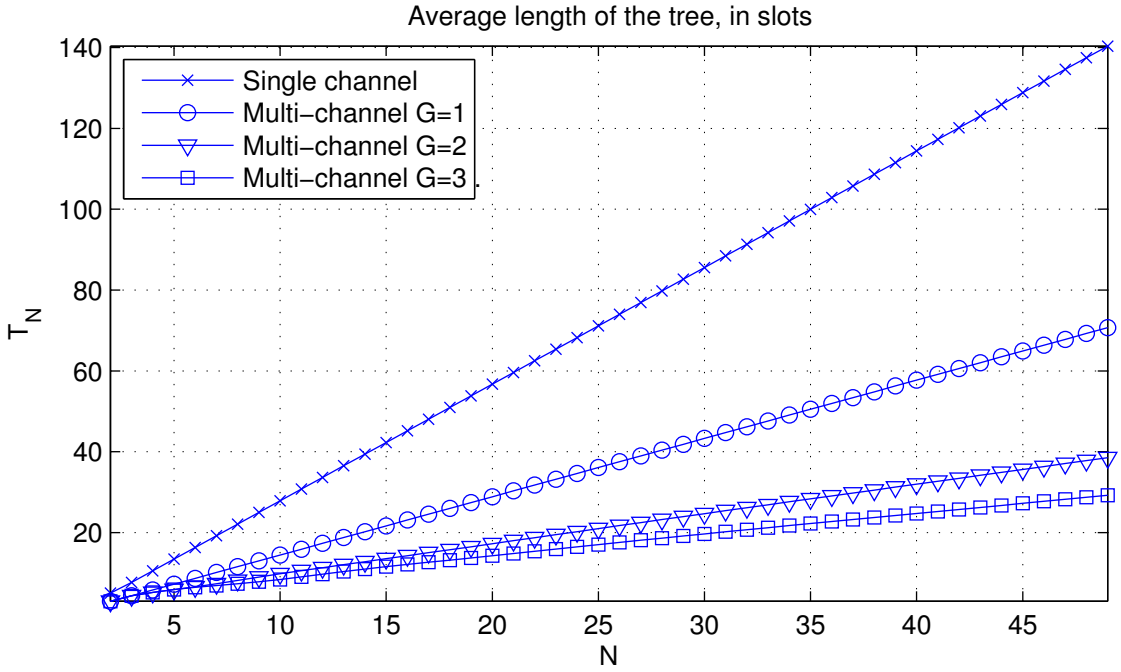


Figure 4.9: Average length of single channel and multi-channel trees, for several G.

From equation 4.55, the relation between T_N and L_N can be inferred. We see that $T_N - \frac{L_N}{2G}$ yields a positive offset, which according to Figure 4.10 increases along with G. This offset can be understood as the efficiency loss that occurs when the number of nodes that lie in one level is lower than the size of the slot, what gets more probable when G is large.

Probability mass function

In our path to get the CDF of the access delay of a multi-channel tree, the second step is to obtain the PMF of the length of the tree, once we have calculated the average value. In order to do so, let us recall equation 4.33, which is the basis of our analysis.

$$\mathcal{T}_N = 1 + \sum_{m=1}^{\infty} \mathcal{T}_N^m. \tag{4.33}$$

This equation shows us that it is possible to express $\mathcal{T}_N$ as an infinite sum of related random variables. If the set of variables $\{\mathcal{T}_N^m\}_1^\infty$ ($m \in \{1, \ldots, \infty\}$) were independent from one another, we might obtain the PMF of $\mathcal{T}_N$ as the convolution of the PMFs of $\{\mathcal{T}_N^m\}_1^\infty$:

$$\Pr\left[\mathcal{T}_N = t\right] = p_{\mathcal{T}_N}(t) = \delta\left[t-1\right] * p_{\mathcal{T}_N^1}(t) * p_{\mathcal{T}_N^2}(t) * p_{\mathcal{T}_N^3}(t) * \ldots, \tag{4.56}$$

where $\delta\left[t-1\right]$ is the Kroenecker delta. Nonetheless, it is clear that variables $\{\mathcal{T}_N^m\}_1^\infty$ are indeed dependent. In order to notice the dependence, one may think of the influence of

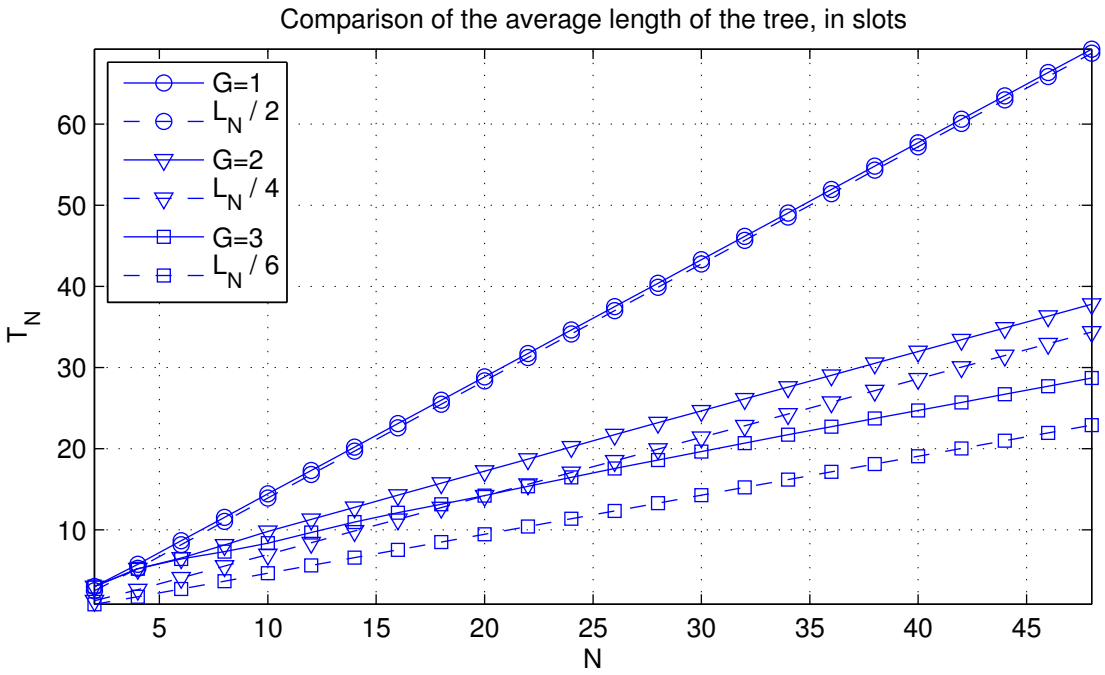


Figure 4.10: Comparison between T_N and $\frac{L_N}{2G}$

the number of slots in one level over the next level. For instance, given some $\mathcal{T}_N^{m-1} = t'$ in the level $m-1$, the number of slots $\mathcal{T}_N^m$ in the level m cannot be more than $2t'$ even if $p_{\mathcal{T}_N^m}(t) > 0$ for $t > 2t'$.

In the case of dependent variables, a more general approach to calculate the PMF of the sum is required. Namely, the joint PMF of all variables is needed, since it contains every influence from one variable to another. However, a joint PMF of an infinite set of variables cannot be defined. Therefore, we have to find a maximum number of variables that we will consider in our analysis, such that the difference between the result yielded by the finite set and the actual result is negligible.

We will denote by M the cardinality of the set $\{\mathcal{T}_N^m\}_1^M$ $(m \in \{1, \ldots, M\})$, containing the variables that we will use to compute an approximate PMF of T_N. As a consequence, M will be the maximum level that we will consider at analyzing the PMF of the tree.

Since whichever node trespassing level M will not be taken into account, we want to set M as high as possible. On the other hand, the greater M the bulkier the operations will be. In order to choose an optimum M, we would need to compute the probability for a tree to reach the level M.

Let us denote by $\mathcal{M}_N$ the number of levels required to complete a tree started with N

initial contenders. In [JdJ00], the authors provide the PMF of this random variable:

$$\Pr\left[\mathcal{M}_N = m\right] = p_{\mathcal{M}_N}(m) = \frac{(2^m)!}{(2^m - N)!(2^m)^N} - \frac{(2^{m-1})!}{(2^{m-1} - N)!(2m-1)^N}. \tag{4.57}$$

We can choose an optimum M for a required accuracy based on this expression. Fortunately, M grows slowly as we increase either the required accuracy or the number of contenders, since the number of nodes in every level grows exponentially, and so do the number of opportunities to successfully transmit. For instance, only $M = 35$ is required to guarantee that 99.9% of trees will be finished if $N = 10000$.

Once we have limited our variables to a finite set, we can define their joint PMF as follows:

$$\Pr\left[\mathcal{T}_N^1, \ldots, \mathcal{T}_N^M = t_1, \ldots, t_M\right] = p_{\mathcal{T}_N^1, \ldots, \mathcal{T}_N^M}(t_1, \ldots, t_M). \tag{4.58}$$

Then, we can denote by $\mathcal{S}_N^M$ the sum of all M random variables in $\{\mathcal{T}_N^m\}_1^M$:

$$\mathcal{S}_N^M = \sum_{m=1}^{M} \mathcal{T}_N^m. \tag{4.59}$$

Given the joint PMF of $\{\mathcal{T}_N^m\}_1^M$, the PMF of $\mathcal{S}_N^M$ can be expressed as:

$$\Pr\left[\mathcal{S}_N^M = s\right] = p_{\mathcal{S}_N^M}(s) = \sum_{t_1=0}^{s} \sum_{t_2=0}^{s-t_1} \cdots \sum_{t_{M-1}=0}^{s-\sum_{i=1}^{M-2} t_m} p_{\mathcal{T}_N^1, \ldots, \mathcal{T}_N^M}(t_1, t_2, \ldots, t_{M-1}, s - \textstyle\sum_{i=1}^{M-1} t_i). \tag{4.60}$$

If we had this $p_{\mathcal{S}_N^M}(s)$, we could convert it directly to the PMF of $\mathcal{T}_N$, as follows:

$$\mathcal{T}_N \cong 1 + \mathcal{S}_N^M, \tag{4.61}$$

$$p_{\mathcal{T}_N}(t) \cong \delta[t-1] * p_{\mathcal{S}_N^M}(t). \tag{4.62}$$

Therefore, in order to obtain $p_{\mathcal{T}_N}(t)$ we need to derive first the joint PMF of $\{\mathcal{T}_N^m\}_1^M$. However, the derivation of this PMF is rather difficult, since we are facing a problem with $M \sim 35$ variables that are all dependent from one another. Therefore, attempting to model the exact dependence from each level to the remaining levels is likely to be cumbersome and analytically intractable.

In order to overcome this challenge, a Markovian simplification is proposed. This means that we will assume that the Markov property holds for our set of variables:

$$\begin{aligned} \Pr\left[\mathcal{T}_N^m = t_m | \mathcal{T}_N^M = t_M, \ldots, \mathcal{T}_N^{m+1} = t_{m+1}, \mathcal{T}_N^{m-1} = t_{m-1}, \ldots, \mathcal{T}_N^1 = t_1\right] \\ \cong \Pr\left[\mathcal{T}_N^m = t_m | \mathcal{T}_N^{m-1} = t_{m-1}\right] \end{aligned} \tag{4.63}$$

In words, this property implies that the number of slots in a given level is only influenced by the number of slots in the very previous level. This is not true in general, since both number of slots and distribution of contenders in the level $m-1$ would be needed to calculate the statistics of the number of slots in the level m. The distribution of contenders is the result of what happened in the tree since the root node, therefore $\mathcal{T}_N^m$ is indeed influenced by previous levels apart from $\mathcal{T}_N^{m-1}$. However, it is clear that what happens in one level heavily affects what happens in the next one, therefore a Markovian approximation should reflect the most of the dependence among levels. In fact, according to the simulation results presented in the next chapter, the statistics derived from this approximation are almost identical to empirical statistics.

The first benefit of applying Markov property is the simple form of the joint PMF of $\{\mathcal{T}_N^m\}_1^M$:

$$p_{\mathcal{T}_N^1,\ldots,\mathcal{T}_N^M}(t_1,\ldots,t_M) = p_{\mathcal{T}_N^1}(t_1)p_{\mathcal{T}_N^2|\mathcal{T}_N^2}(t_2|t_1)\ldots p_{\mathcal{T}_N^M|\mathcal{T}_N^{M-1}}(t_M|t_{M-1}). \tag{4.64}$$

Therefore, the problem is now reduced to find $p_{\mathcal{T}_N^m|\mathcal{T}_N^{m-1}}(t_m|t_{m-1})$, which is the probability to get $\mathcal{T}_N^m$ slots in the level m given $\mathcal{T}_N^{m-1}$ in the level $m-1$. In order to facilitate the derivation of $p_{\mathcal{T}_N^m|\mathcal{T}_N^{m-1}}(t_m|t_{m-1})$, we can translate the number of slots $\mathcal{T}_N^m$ into the number of nodes $\mathcal{L}_N^m$ and the number of collisions $\mathcal{X}_N^m$, as it was stated in equations 4.34 and 4.36. The conversion among PMFs within one level is obtained as follows:

$$p_{\mathcal{T}_N^m}(t_m) = \begin{cases} p_{\mathcal{L}_N^m}(0) = p_{\mathcal{X}_N^{m-1}}(0) & t_m = 0. \\ \displaystyle\sum_{i=1}^{2G} p_{\mathcal{L}_N^m}\left(2G\cdot(t_m-1)+i\right) = \sum_{i=0}^{G} p_{\mathcal{X}_N^{m-1}}\left(G\cdot(t_m-1)+i\right) & t_m > 0. \end{cases} \tag{4.65}$$

Thus, our problem is further reduced to find $p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1}}(x_m|x_{m-1})$, that is, the probability to get x_m collisions in the level m given x_{m-1} collisions in the level $m-1$. In order to calculate this probability, knowing the number of contenders in every collision in the level $m-1$ is required. Indeed, the probability to produce e.g. two new collisions is higher if the parent collision occurred with eight contenders than with four contenders. A priori, we cannot know the number of contenders involved in the given x_{m-1} collisions, but we can consider every different possibility and then apply the law of total probability.

Let us define $\mathcal{K}_m$ as the total number of devices which have been involved in collisions in the level m. With this new variable, we can calculate the following conditional probability:

$$\Pr\left[\mathcal{X}_N^m = x_m|\mathcal{X}_N^{m-1} = x_{m-1}, \mathcal{K}_{m-1} = k_{m-1}\right] = p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1}}(x_m|x_{m-1},k_{m-1}), \tag{4.66}$$

which is the probability to generate x_m collisions in the level m given x_{m-1} collisions and k_{m-1} contenders in the level $m-1$. Calculating this probability should be easier than simply $p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1}}(x_m|x_{m-1})$. We can relate both probabilities by applying the law of total

probability:

$$p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1}}(x_m|x_{m-1}) = \sum_{k_{m-1}=0}^{N} p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1}}(x_m|x_{m-1},k_{m-1}) p_{\mathcal{K}_{m-1}|\mathcal{X}_N^{m-1}}(k_{m-1}|x_{m-1}). \tag{4.67}$$

As expected, expression 4.67 has split our problem into two sub-problems:

1. $p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1}}(x_m|x_{m-1},k_{m-1})$ is the aforementioned probability to generate x_m collisions in the level m given x_{m-1} collisions and k_{m-1} contenders in the level $m-1$.
2. $p_{\mathcal{K}_{m-1}|\mathcal{X}_N^{m-1}}(k_{m-1}|x_{m-1})$ is the probability to produce x_{m-1} collisions while having k_{m-1} contenders.

Let us tackle the second sub-problem first. Again, we can model this problem with balls and bins. In this case, we know the number of bins that contain two or more balls, which is x_{m-1}. We also know that k_{m-1} balls out of N are 'contenders', that is, every ball shares bin with at least other ball.

In order to compute the desired probability, we just need to compute the number of ways to generate x_{m-1} collisions with exactly k_{m-1} contenders, and then divide it by the total number of ways to produce x_{m-1} collisions for any possible $\mathcal{K}_{m-1}$. Let us denote the former by $\Gamma_{k_{m-1}}^{x_{m-1}}$, the resulting expression is:

$$p_{\mathcal{K}_{m-1}|\mathcal{X}_N^{m-1}}(k_{m-1}|x_{m-1}) = \frac{\Gamma_{k_{m-1}}^{x_{m-1}}}{\sum_{k_{m-1}} \Gamma_{k_{m-1}}^{x_{m-1}}}. \tag{4.68}$$

The reasoning to compute $\Gamma_{k_{m-1}}^{x_{m-1}}$ is the following. We have N balls, k_{m-1} of which are 'contenders', what means that $N - k_{m-1}$ balls are alone in their occupied bin. Therefore, we have a total of $x_{m-1} + N - k_{m-1}$ occupied bins: x_{m-1} with collisions and $N - k_{m-1}$ with single balls, as it is depicted in Figure 4.11. Knowing this, we can compute the number of ways to arrange N balls into $x_{m-1} + N - k_{m-1}$ bins such that x_{m-1} of them have more than one balls as $\Psi_{N-k_{m-1}+x_{m-1},x_{m-1}}^{N}$. Finally, we just need to compute the number of ways to select $x_{m-1} + N - k_{m-1}$ bins out of 2^{m-1} possible bins as $\binom{2^{m-1}}{N-k_{m-1}+x_{m-1}}$ and the number of ways to arrange $x_{m-1} + N - k_{m-1}$ bins as $(x_{m-1} + N - k_{m-1})!$. As a result, our final expression is:

$$\Gamma_{k_{m-1}}^{x_{m-1}} = \Psi_{N-k_{m-1}+x_{m-1},x_{m-1}}^{N} \binom{2^{m-1}}{N - k_{m-1} + x_{m-1}} (N - k_{m-1} + x_{m-1})! \tag{4.69}$$

Combining expressions 4.68 and 4.69 we obtain our final result for $p_{\mathcal{K}_{m-1}|\mathcal{X}_N^{m-1}}(k_{m-1}|x_{m-1})$.

Now we are left with the computation of $p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1}}(x_m|x_{m-1},k_{m-1})$. The approach to obtain this probability relies on seeing the problem as a number theory problem. Namely, we will cope with integer partitions.

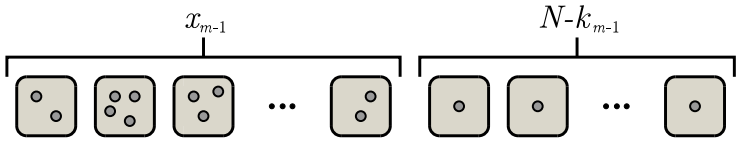


Figure 4.11: Collisions and successful transmissions as occupied bins.

Let us consider the following example, where $x_{m-1} = 4$ and $k_{m-1} = 10$. There are five different ways to decompose 12 contenders into 4 collisions, which are the five different partitions of 10 in 4 parts, such that every part is greater than 1. Namely, these partitions are:

$$2 + 2 + 2 + 6 = 12,$$
$$2 + 2 + 3 + 5 = 12,$$
$$2 + 2 + 4 + 4 = 12,$$
$$2 + 3 + 3 + 4 = 12,$$
$$3 + 3 + 3 + 3 = 12.$$

At this point, it is easy to see why it is interesting to decompose k_{m-1} into partitions. Given a certain partition of contenders in the level $m-1$, it is immediate to compute the probability of x_m collisions in the level m. It is only needed to compute the probability to generate 0, 1 or 2 new collisions for every collision in the level $m-1$, which is now trivial since we know the number of contenders in each one.

We can incorporate partitions by using again the law of total probability, but we first need to compute the probability of each partition, i.e. the probability of each distribution of contenders. Let us denote by $\Pi_{x_{m-1}}^{k_{m-1}}$ the set of partitions of k_{m-1} in x_{m-1} parts greater than 1. Each element of $\Pi_{x_{m-1}}^{k_{m-1}}$ is a partition, denoted by π_i, where $i \in \{1, \ldots, |\Pi_{x_{m-1}}^{k_{m-1}}|\}$. We also define $\mathcal{P}_{x_{m-1}}^{k_{m-1}}$ as the random variable modeling the selected partition of k_{m-1} contenders in x_{m-1} parts greater than 1. We want to derive an expression for:

$$\Pr\left[\mathcal{P}_{x_{m-1}}^{k_{m-1}} = \pi_i\right] = p_{\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(\pi_i). \tag{4.70}$$

Let κ_j^i be a part of the partition π_i, for $j \in \{1, \ldots, x_{m-1}\}$. The following relations hold:

$$\kappa_j^i > 1, \tag{4.71}$$

$$\sum_j \kappa_j^i = k_{m-1}, \tag{4.72}$$

$$\pi_i = (\kappa_1^i, \kappa_2^i, \ldots, \kappa_{x_{m-1}}^i). \tag{4.73}$$

In addition, let us define $\#_a^i$ as the number of occurrences of the number a within the partition π_i. We can formally define this new variable as follows:

$$\#_a^i = \sum_j \delta\left[\kappa_j^i - a\right], \tag{4.74}$$

where $\delta[\cdot]$ is the Kroenecker delta. With these variables, we can compute $p_{\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(\pi_i)$ as follows:

$$p_{\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(\pi_i) = \frac{k_{m-1}!}{\Psi_{x_{m-1},x_{m-1}}^{k_{m-1}}} \prod_{j=1}^{x_{m-1}} \frac{1}{\kappa_j^i!\#_j^i!}. \tag{4.75}$$

The derivation of equation 4.75 is explained in the Appendix B. Once we have this probability, we apply again the law of total probability:

$$p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1}}(x_m|x_{m-1},k_{m-1}) = \sum_i p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1},\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(x_m|x_{m-1},k_{m-1},\pi_i)\cdot p_{\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(\pi_i). \tag{4.76}$$

At this point, we have added information about the number and the distribution of the contenders that occupy some given collisions. At last, that information suffices to compute the PMF of the number of collisions in the level m. In order to do so, we define $\mathcal{Y}_n$ as the number of child collisions of a parent collision of n contenders. Since we are analyzing a Binary Tree Algorithm, the sample space of $\mathcal{Y}_n$ is simply $\{0,1,2\}$, i.e. at most two collisions can be children of one initial collision. The PMF of $\mathcal{Y}_n$ is the result of a simple combinatorial problem:

$$\Pr\left[\mathcal{Y}_n = y_n\right] = p_{\mathcal{Y}_n}(y_n) = \begin{cases} 0.5\cdot\delta[y_n] + 0.5\cdot\delta[y_n-1] & n=2. \\ (n+1)\left(\frac{1}{2}\right)^{n-1}\delta[y_n-1] + \left(1-(n+1)\left(\frac{1}{2}\right)^{n-1}\right)\delta[y_n-2] & n>2. \end{cases} \tag{4.77}$$

Given a certain distribution (partition) of contenders, we can use $p_{\mathcal{Y}_n}(y_n)$ to compute the probability that some collision in level $m-1$ generates 0, 1 or 2 collisions in level m, as illustrated in Figure 4.12. Furthermore, since the subtrees generated by the parent collisions are not related, variables $\{\mathcal{Y}_n\}$ are independent. Therefore we can compute the PMF of the sum of all $\{\mathcal{Y}_n\}$ as the convolution of all of them:

$$p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1},\mathcal{K}_{m-1},\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(x_m|x_{m-1},k_{m-1},\pi_i) = \overset{x_{m-1}}{\underset{j=1}{\circledast}} p_{\mathcal{Y}_{\kappa_j^i}}(y_{\kappa_j^i}), \tag{4.78}$$

where:

$$\overset{N}{\underset{j=1}{\circledast}} f(x_j) = f(x_1) * \ldots * f(x_N). \tag{4.79}$$

Finally, after combining expressions 4.65 – 4.78, we are able to compute the joint PMF $p_{\mathcal{T}_N^1,\ldots,\mathcal{T}_N^M}(t_1,\ldots,t_M)$, whose definition is repeated here for clearness:

$$p_{\mathcal{T}_N^1,\ldots,\mathcal{T}_N^M}(t_1,\ldots,t_M) = p_{\mathcal{T}_N^1}(t_1)p_{\mathcal{T}_N^2|\mathcal{T}_N^2}(t_2|t_1)\ldots p_{\mathcal{T}_N^M|\mathcal{T}_N^{M-1}}(t_M|t_{M-1}). \tag{4.64}$$

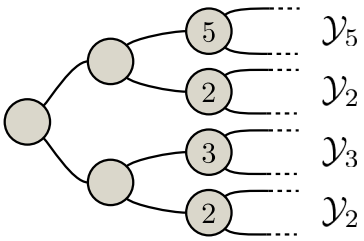


Figure 4.12: Example distribution of contenders in for $m = 2$ and $N = 12$

Each of the conditional PMFs on the right-hand side reflect the dependence relation between each pair of levels. As they are bivariate, they admit a graphical representation, so that we can observe this dependence. In fact, this dependence is better seen through the conditional probability of collisions, $p_{\mathcal{X}_N^m|\mathcal{X}_N^{m-1}}(x_m|x_{m-1})$, since collisions provide better granularity. In Figure 4.13, two selected examples of this PMF are depicted: $p_{\mathcal{X}_{30}^6|\mathcal{X}_{30}^5}(x_6|x_5)$ and $p_{\mathcal{X}_{50}^6|\mathcal{X}_{50}^5}(x_6|x_5)$. From the pictures, one can notice how the average and variance of $\mathcal{X}_{30}^6$ and $\mathcal{X}_{50}^6$ increases along with $\mathcal{X}_{30}^5$ and $\mathcal{X}_{50}^5$, respectively, although the relation is not linear.

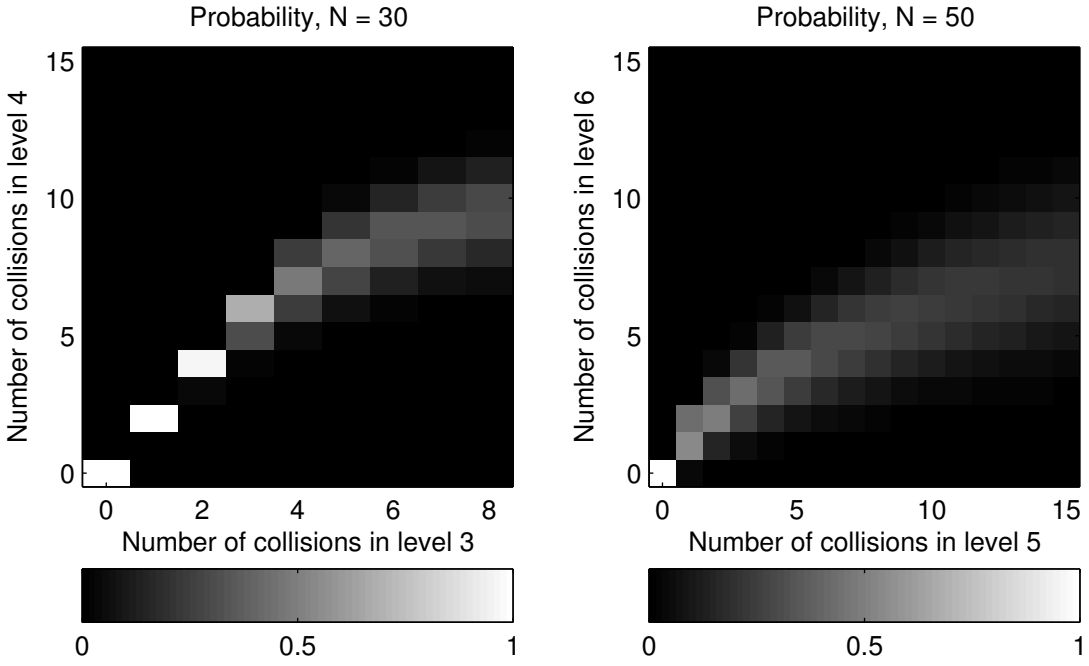


Figure 4.13: Example of conditional PMFs of collisions

The last step in our path to get $p_{\mathcal{T}_N}(t_N)$ is the conversion between the joint PMF and the PMF of the sum, which was presented in equation 4.60. However, this equation cannot be used directly. Although it is exact, equation 4.60 is computationally unsuitable, since summations will be converted into nested for-loops. Hence, the number of total iterations will grow exponentially as we increase s or M, leading to an exponential grow of the

computation time. Therefore, we need to find an alternative manner to obtain $p_{\mathcal{S}_N^M}(s)$.

Once more, we can benefit from the Markov property to achieve that goal. Let us define $\mathcal{S}_N^m$ as the partial sum of the number of slots per level:

$$\mathcal{S}_N^m = \sum_{i=1}^{m} \mathcal{T}_N^i. \tag{4.80}$$

We can now use this new random variable to reformulate our objective:

$$p_{\mathcal{S}_N^M}(s_M) = p_{\mathcal{T}_N^M + \mathcal{S}_N^{M-1}}(s_M) = \sum_{t_M} p_{\mathcal{T}_N^M, \mathcal{S}_N^{M-1}}(t_M, s_M - t_M). \tag{4.81}$$

In expression 4.81, $p_{\mathcal{S}_N^M}(s)$ is written as a function of the bivariate PMF $p_{\mathcal{T}_N^M, \mathcal{S}_N^{M-1}}(t, s_{M-1})$. Obtaining the PMF of the sum of two variables is easy, since only a simple summation is required. If we manage to express the statistics of every partial sum $\mathcal{S}_N^m$ as a function of two variables, we should be able to efficiently compute $p_{\mathcal{S}_N^M}(s)$. Combining Markov and associative properties, we can accomplish that as follows:

$$\begin{aligned}
p_{\mathcal{T}_N^m, \mathcal{S}_N^{m-1}}(t_m, s_{m-1}) &= \sum_{t_{m-1}} p_{\mathcal{T}_N^m, \mathcal{T}_N^{m-1}, \mathcal{S}_N^{m-1}}(t_m, t_{m-1}, s_{m-1}) \\
&= \sum_{t_{m-1}} p_{\mathcal{T}_N^m | \mathcal{T}_N^{m-1}, \mathcal{S}_N^{m-1}}(t_m | t_{m-1}, s_{m-1}) p_{\mathcal{T}_N^{m-1}, \mathcal{S}_N^{m-1}}(t_{m-1}, s_{m-1}) \\
&= \sum_{t_{m-1}} p_{\mathcal{T}_N^m | \mathcal{T}_N^{m-1}}(t_m | t_{m-1}) p_{\mathcal{T}_N^{m-1}, \mathcal{S}_N^{m-1}}(t_{m-1}, t_{m-1} + s_{m-2}) \\
&= \sum_{t_{m-1}} p_{\mathcal{T}_N^m | \mathcal{T}_N^{m-1}}(t_m | t_{m-1}) p_{\mathcal{T}_N^{m-1}, \mathcal{S}_N^{m-2}}(t_{m-1}, s_{m-1} - t_{m-1}).
\end{aligned} \tag{4.82}$$

We can see how $p_{\mathcal{T}_N^m, \mathcal{S}_N^{m-1}}(t_m, s_{m-1})$ is now written as a function of $p_{\mathcal{T}_N^m | \mathcal{T}_N^{m-1}}(t_m | t_{m-1})$, which is already known, and $p_{\mathcal{T}_N^{m-1}, \mathcal{S}_N^{m-2}}(t_{m-1}, s_{m-2})$, which is the 'previous step'. Indeed, $p_{\mathcal{S}_N^M}(s)$ can be now computed recursively, starting with the following initial conditions:

$$p_{\mathcal{S}_N^1}(s_1) = p_{\mathcal{T}_N^1}(t_1) = \delta_{t_1,1}, \tag{4.83}$$

$$p_{\mathcal{T}_N^2, \mathcal{S}_N^1}(t_2, s_1) = p_{\mathcal{T}_N^2}(t_2) \cdot p_{\mathcal{S}_N^1}(s_1), \tag{4.84}$$

$$p_{\mathcal{S}_N^2}(s_2) = \sum_{t_2} p_{\mathcal{T}_N^2, \mathcal{S}_N^1}(t_2, s_2 - t_2). \tag{4.85}$$

In order to obtain $p_{\mathcal{T}_N^2}(t_2)$, expression 4.65 needs to be applied. After this last step, we have finally reached the end of our calculation, the PMF of the number of slots in a multi-channel tree, $p_{\mathcal{T}_N}(t)$, is at last obtained after applying expression 4.62.

Since we have used an approximate method (the Markovian approach) to calculate $p_{\mathcal{T}_N}(t)$, we should check how good it is before using it to predict the behavior of Tree Algorithms. In order to do so, we can compare the results that this method yields in two extreme cases:

1. When $G = 1$. In this case, a slot contains only a contention frame, therefore the recursive properties of the tree are preserved. That means that we can still use a recursive approach to compute the PGF and hence the PMF, as in equation 4.18. Indeed, it is trivial that:

$$p_{\mathcal{T}_N}(t) = p_{\mathcal{L}_N}(2(t-1)+1) \qquad \text{when } G = 1. \tag{4.86}$$

   In fact, a similar relation also holds for the PMF of the access delay. A comparison of the result of both approaches is depicted in Figure 4.14. One can notice that the PMF obtained from the Markovian approach exhibits a slightly larger variance than that of the recursive approach, which is presumed to be exact. This is a consequence of the Markovian approximation, but it is less noticeable when comparing CDFs. Indeed, the maximum difference between CDFs is less than 2.5%.

2. When $G \to \infty$. In this case, all nodes belonging to the same level at transmitted at once, therefore the PMF of the number of slots converges to the PMF of the number of levels needed to complete the tree.

$$p_{\mathcal{T}_N}(t) = p_{\mathcal{M}_N}(t) \qquad \text{when } G \to \infty. \tag{4.87}$$

   In Figure 4.15, a comparison between the Markov approach and the PMF and CDF of the levels reached by the tree is presented. This time, the coincidence is almost exact, what invites us to think that the Markovian approximation improves when G increases.

Finally, in Figure 4.16 several CDFs for different values of G are shown. It can be observed that the larger G, the lower the average and variance of $p_{\mathcal{T}_N}(t)$. This Figure confirms the intuition that the length of the tree decreases as we increase the number of transmitted nodes per slot. Indeed, it matches the average depicted in Figure 4.9, which was computed independently.

4.3.2 Access delay

Finally, we address the central goal of this thesis, which is the derivation of the cumulative distribution function (CDF) of the access delay for a multichannel tree. After the analysis of single channel trees and the length of multi-channel trees, we have all the tools that we require in this section.

Let Θ_N^m be the random variable modeling the access delay of one device that transmits at level m. In other words, Θ_N^m is the number of slots transmitted until the successful transmission of a certain device. In addition, let us define Ω_N^m as the position in which the slot containing that device was transmitted. We can decompose Θ_N^m as follows:

$$\Theta_N^m = \mathcal{S}_N^{m-1} + \Omega_N^m. \tag{4.88}$$

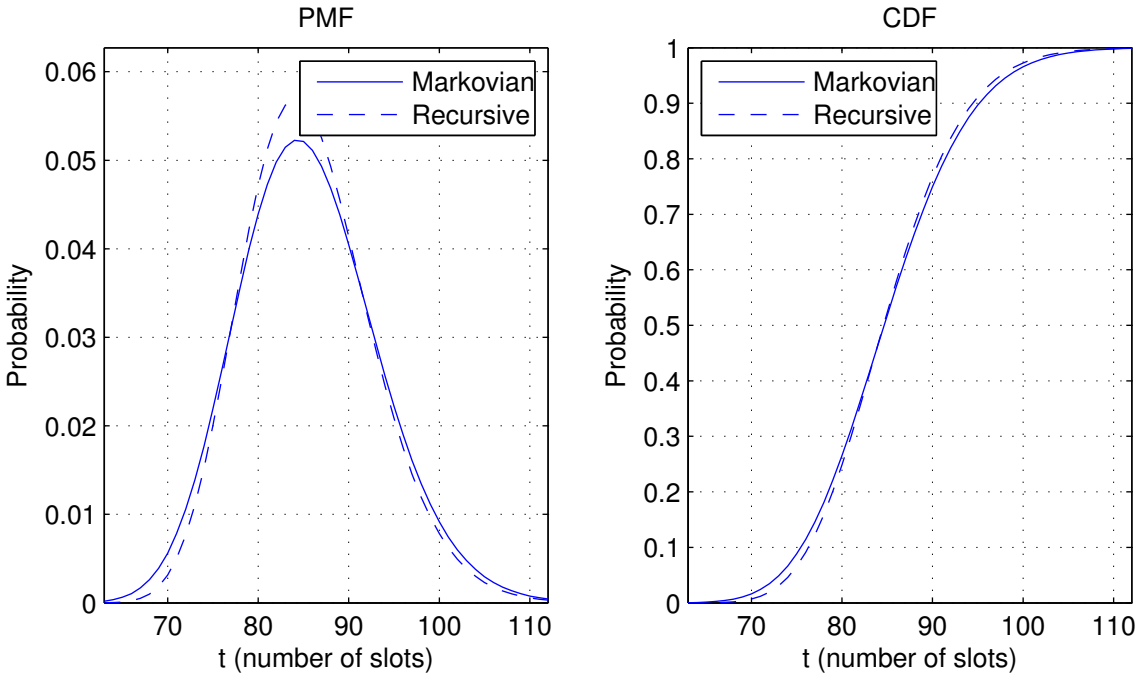


Figure 4.14: Comparison between Markovian and recursive approaches, for $N = 60$ and $G = 1$

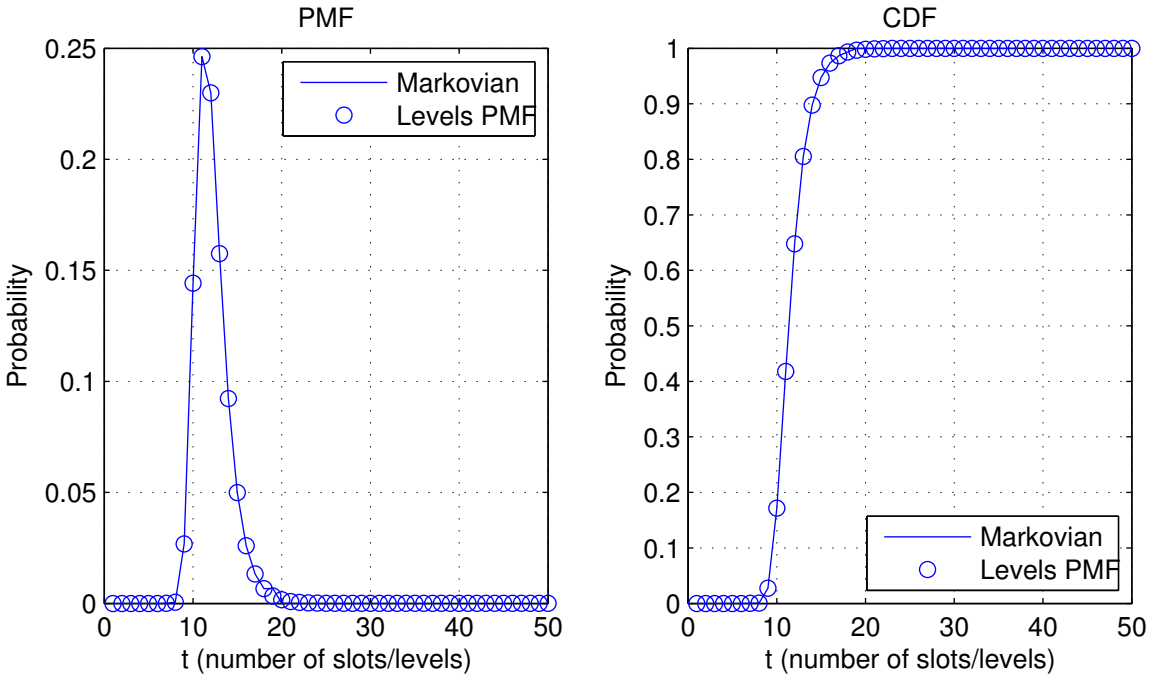


Figure 4.15: Comparison between Markovian approach and PMF and CDF of the levels, for $N = 60$ and $G = 25$

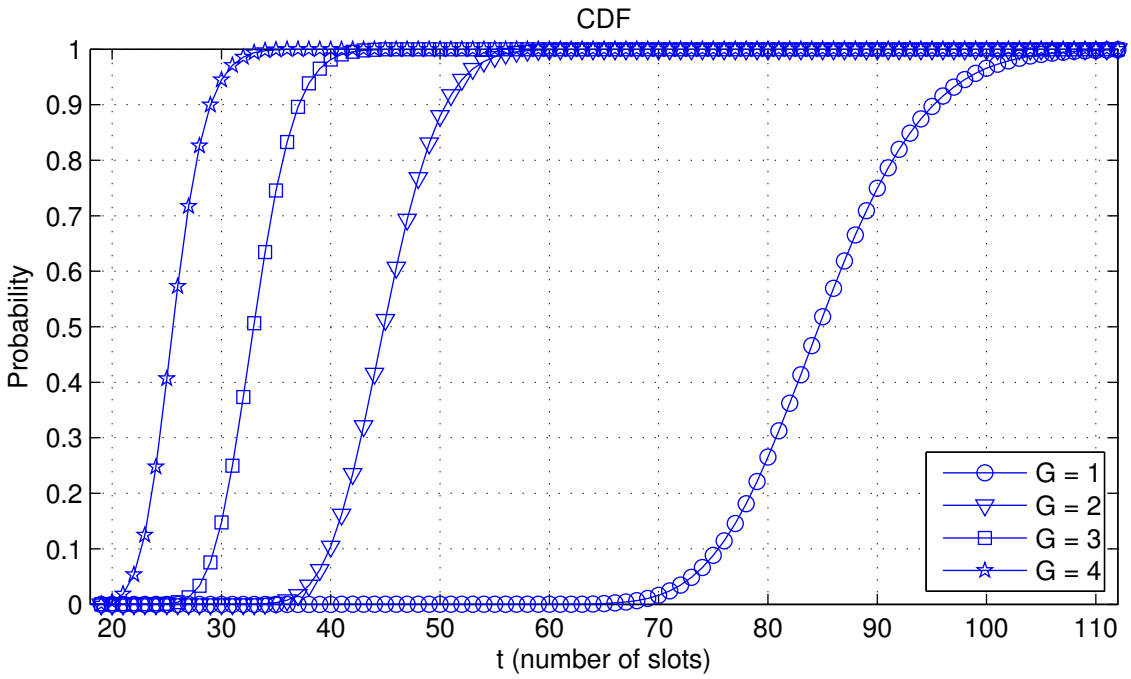


Figure 4.16: CDFs of the length of a multi-channel tree with different values of G, for $N = 60$

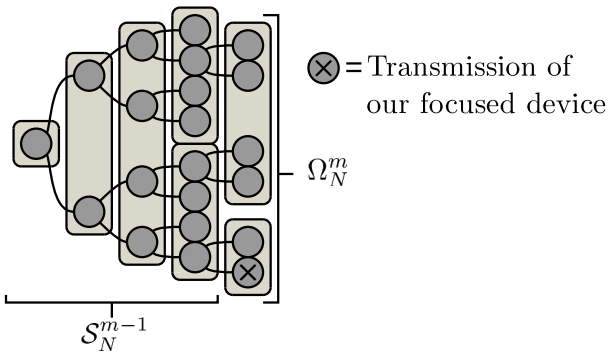


Figure 4.17: Illustration of the calculation of the access delay

The meaning of this equation is illustrated in Figure 4.17. In this figure, our focused device has transmitted in the second slot of the fourth level, so that we can compute the access delay that it has experienced by counting the number of slots up to level 3 and then the number of slots within the level 4. In this way, we decompose the problem into two tractable variables. Under this approach, we want to derive an expression for:

$$\Pr\left[\Theta_N^m = \theta_m\right] = p_{\Theta_N^m}(\theta_m). \tag{4.89}$$

Again we face the problem of obtaining the PMF of the sum of two dependent random variables. As in the previous section, this is accomplished through the joint PMF:

$$p_{\Theta_N^m}(\theta_m) = \sum_{\omega_m=0}^{\theta_m} p_{\Omega_N^m,\mathcal{S}_N^{m-1}}(\omega_{m-2}, \theta_m - \omega_m). \tag{4.90}$$

Therefore, the problem is to derive the joint PMF of $\mathcal{S}_N^{m-1}$ and Ω_N^m. The law of total probability can be applied to introduce $\mathcal{T}_N^m$, which provides information about the number of slots in the level m. This should be useful to infer the probability to transmit at position $\Omega_N^m = \omega_m$.

$$p_{\Omega_N^m,\mathcal{S}_N^{m-1}}(\omega_m, s_{m-1}) = \sum_{t_m} p_{\Omega_N^m|\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|t_m, s_{m-1}) p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(t_m, s_{m-1}). \tag{4.91}$$

The second term in the right-hand side of this equation is already known, as it was derived in 4.82. In fact, the presence of $\mathcal{S}_N^{m-1}$ justifies the analysis of the length of the tree that was performed in the previous section. Therefore, we need to calculate only the first term, which is the probability to transmit in position ω_m out of t_m slots in the level m, given also that the length of the tree so far is s_{m-1}.

If the tree is unbiased, as we have assumed for the whole analysis, all nodes within a tree are equally like, since the tree is statistically symmetrical. Therefore, if $G = 1$, every slot is equally likely, since they all contain two nodes. Thus, the desired probability in that case is:

$$p_{\Omega_N^m|\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|t_m, s_{m-1}) = p_{\Omega_N^m|\mathcal{L}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|2t_m, s_{m-1}) = \frac{1}{t_m} \qquad \text{when } G = 1. \tag{4.92}$$

However, if $G > 1$ the number of nodes in one slot is not fixed, but it ranges from 2 to $2G$ nodes. Namely, given l_m nodes in the level m that generate t_m slots, there will be $t_m - 1$ slots of size $2G$ and one (the last one) of size $l_m - 2G(t_m - 1)$. This difference in the size of the slots produces an asymmetry in the tree: it is less probable two transmit in the last slot than in the rest of them. In order to cope with this asymmetry, let us introduce the

number of nodes $\mathcal{L}_N^m$ in the level m into the problem:

$$p_{\Omega_N^m|\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|t_m,s_{m-1}) =$$

$$= \sum_{l_m} p_{\Omega_N^m|\mathcal{L}_N^m,\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|l_m,t_m,s_{m-1}) p_{\mathcal{L}_N^m|\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(l_m|t_m,s_{m-1}) \tag{4.93}$$

$$= \sum_{l_m} p_{\Omega_N^m|\mathcal{L}_N^m}(\omega_m|l_m) p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}|\mathcal{L}_N^m}(t_m,s_{m-1}|l_m) \frac{p_{\mathcal{L}_N^m}(l_m)}{p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(t_m,s_{m-1})}. \tag{4.94}$$

In the first term of expression 4.94, the simplification $p_{\Omega_N^m|\mathcal{L}_N^m,\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|l_m,t_m,s_{m-1}) = p_{\Omega_N^m|\mathcal{L}_N^m}(\omega_m|l_m)$ was applied, since it is clear that the probability to transmit in the slot ω_m is only influenced by the number of nodes l_m in the level m. In the second term, the Bayes' theorem was applied in order to change the order of the variables.

The probability to transmit in the slot ω_m of level m given a total number of nodes l_m is simple to compute, since it is just the number of nodes in one slot divided by the total number of slots. However, since the last slot may have a different size with respect to the others, two cases have to be considered:

$$p_{\Omega_N^m|\mathcal{L}_N^m}(\omega_m|l_m) = \begin{cases} \frac{2G}{l_m} & \omega_m < t_m, \\ \frac{l_m - 2G(t_m-1)}{l_m} & \omega_m = t_m, \\ 0 & \text{otherwise}, \end{cases} \qquad \text{where } t_m = \left\lceil \frac{l_m}{2G} \right\rceil. \tag{4.95}$$

Now we are left with calculating the probability to have t_m slots in the level m and a total of s_{m-1} slots from level 1 to level $m-1$, given that we have l_m nodes in the level m. From equation 4.34 we know that the number of slots is completely determined given the number of nodes, therefore we only allow for $t_m = \lceil \frac{l_m}{2G} \rceil$. This condition leads to the following expression:

$$p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}|\mathcal{L}_N^m}(t_m,s_{m-1}|l_m) = \begin{cases} \frac{p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(t_m,s_{m-1})}{p_{\mathcal{T}_N^m}(t_m)} & t_m = \left\lceil \frac{s_m}{2G} \right\rceil. \\ 0 & \text{otherwise}. \end{cases} \tag{4.96}$$

Combining 4.94 and 4.96 yields the following expression:

$$p_{\Omega_N^m|\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(\omega_m|t_m,s_{m-1}) = \sum_{l_m} p_{\Omega_N^m|\mathcal{L}_N^m}(\omega_m|l_m) \frac{p_{\mathcal{L}_N^m}(l_m)}{p_{\mathcal{T}_N^m}\left(\left\lceil \frac{s_m}{2G} \right\rceil\right)}. \tag{4.97}$$

Finally, if we merge expressions 4.91 and 4.97 we obtain the following result after some basic manipulation:

$$\begin{aligned} p_{\Omega_N^m,\mathcal{S}_N^{m-1}}(\omega_m,s_{m-1}) &= \sum_{t_m} \left(\sum_{l_m} p_{\Omega_N^m|\mathcal{L}_N^m}(\omega_m|l_m) \frac{p_{\mathcal{L}_N^m}(l_m)}{p_{\mathcal{T}_N^m}\left(\left\lceil \frac{s_m}{2G} \right\rceil\right)} \right) p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}(t_m,s_{m-1}) \\ &= \sum_{l_m} p_{\Omega_N^m|\mathcal{L}_N^m}(\omega_m|l_m) p_{\mathcal{T}_N^m,\mathcal{S}_N^{m-1}}\left(\left\lceil \frac{s_m}{2G} \right\rceil, s_{m-1}\right) \frac{p_{\mathcal{L}_N^m}(l_m)}{p_{\mathcal{T}_N^m}\left(\left\lceil \frac{s_m}{2G} \right\rceil\right)}. \end{aligned} \tag{4.98}$$

At this point, we just need to combine equations 4.98 and 4.90 to obtain the final PMF of the access delay. In Figure 4.18, example PMFs and CDFs are shown for $N = 60$ and $G = 1$. The strange shape of the PMF reflects the level-wise traversal, since every node belonging to an early level have the same probabilities to be occupied by a device. Finally, in Figure 4.19, several CDFs for different values of G are depicted. As it happened with the length of the tree, the average and variance of the access delay decreases when G increases. From this plot, one may calculate which is the optimum G for a given delay constraint, so that the number of nodes per slot is as low as possible.

In the light of this result, we confirm our hypothesis that multi-channel Tree Algorithms can yield a lower access delay with respect to single channel Tree Algorithms, in exchange for additional resources. Thus, Tree Algorithms may be applied to multi-channel scenarios where contention is expected, such as the LTE Random Access Channel, in order to provide reliability to delay sensitive devices. Depending on the application, different multi-channel or single channel configurations for a Tree Algorithm can be selected in order to meet the delay requirements with an optimal use of resources.

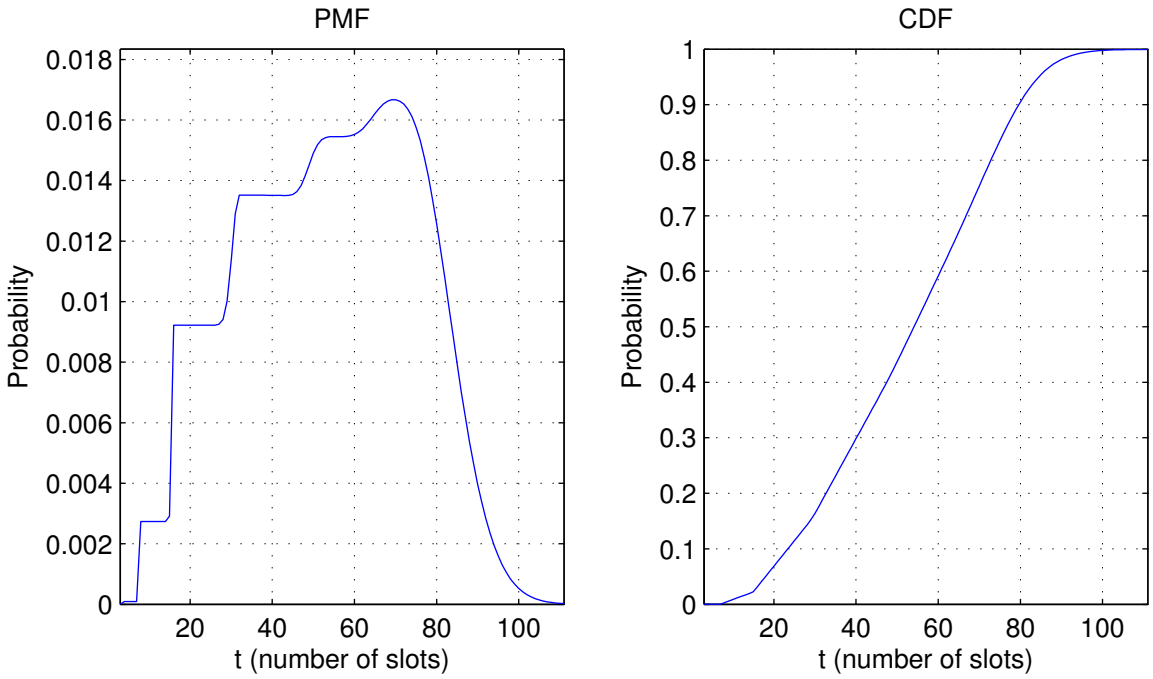


Figure 4.18: Example of PMF and CDF of the access delay, for $N = 60$ and $G = 1$

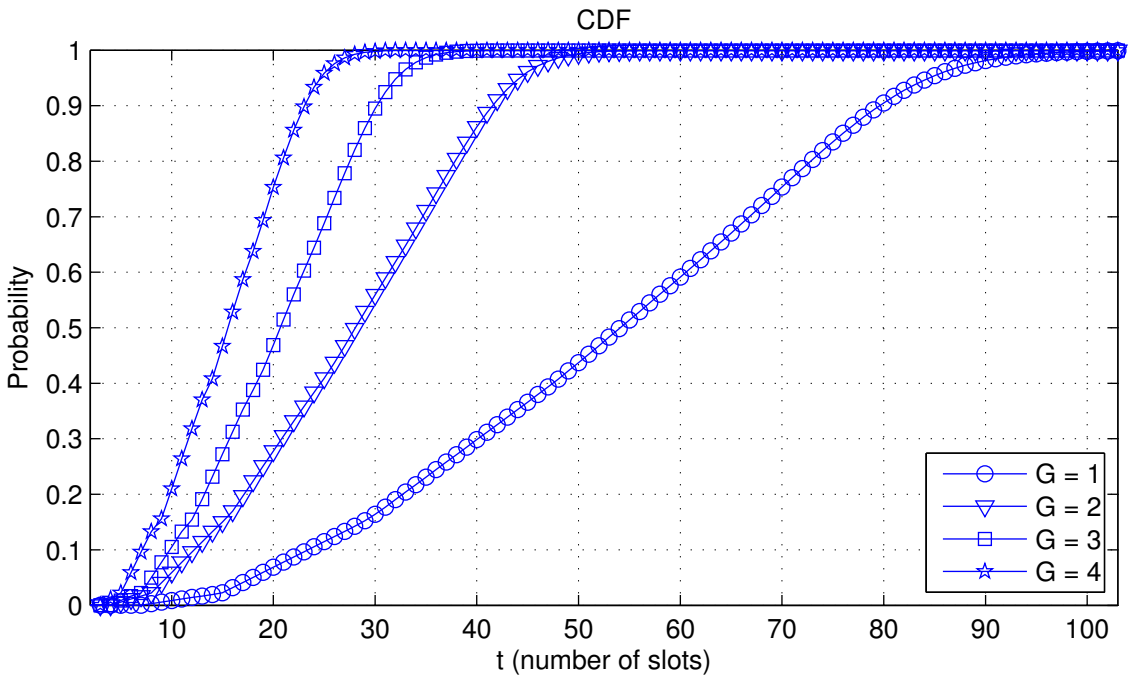


Figure 4.19: Comparison of CDFs of the access delay, for $N = 60$

Chapter 5

Simulations

In the previous chapter, an analytical model for the length of the tree and the access delay was derived. In order to check the accuracy of the model, simulations were performed and their results were compared with the predicted values. In this chapter, the simulator design and the simulation results are presented.

5.1 Simulator design

A custom simulator was built in order to fit the characteristics of Tree Algorithms. When designing the simulator, two different options were considered:

1. A device-centered approach, where every device and the eNodeB would be represented as an object by the code of the simulator. In this case, the succession of time slots would be simulated and objects would interact with one another as in a real system.
2. A matrix-modeled approach, where the whole tree is represented as a matrix of decisions, and hence any parameter of the tree can be directly obtained from it. In order to understand this approach, let us consider the tree depicted in the Figure 5.1. This tree, started with $N = 3$ contenders, is totally determined by the splitting

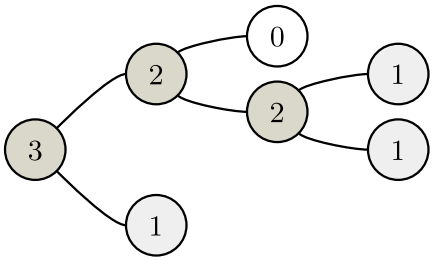


Figure 5.1: Basic example

decisions of its contenders. As one can infer from the tree, one device selected 1 after the first collision and two devices selected 0 after the first collision. Then, both collided devices selected 1 before splitting after the third collision, when they finally selected different nodes. These decisions are summarized in the matrix $\mathbf{C}$ as follows:

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & - & - \\ 0 & 1 & 1 \end{bmatrix}. \tag{5.1}$$

In this matrix, every row represents the sequence of decisions of one device, whereas every column stands for the vector of decisions within one level. Matrix $\mathbf{C}$ totally describes the tree, as it compiles the decisions of the devices, which is the only information needed to generate any tree. This implies that every delay-related parameter may be also extracted from $\mathbf{C}$. Thus, this approach relies on randomly generating and analyzing these matrices.

Although the first approach provides a more intuitive manner to perform simulations, the second one should have a better performance, since it reduces whole process of generating a tree into a simple matrix. As a consequence, the second approach was selected and MATLAB was chosen as the framework to develop the simulator, in order to cope with matrix operations efficiently.

5.1.1 Structure of the simulator

The objective of the simulator is to extract the total length of the tree L and the vector of access delays $\boldsymbol{\theta}$ of each device. For example, given the decisions matrix presented in 5.1, the expected result would be:

$$L = 7, \qquad \boldsymbol{\theta} = \begin{bmatrix} 3 & 6 & 7 \end{bmatrix}.$$

In order to do extract L and $\boldsymbol{\theta}$ from $\mathbf{C}$, the simulator is structured in three phases, as described below:

1. In the first phase, the *tree generation phase*, the decisions matrix $\mathbf{C}$ is generated along with some auxiliary vectors.
2. In the second phase, the *offsets phase*, the position of every successful transmission within each level is computed.
3. In the third phase, the *processing phase*, the length of the tree and the access delays are computed.

In the following sections, a detailed description of each phase is presented.

Tree generation phase

The first step under the matrix-modeled simulator is to randomly generate a decisions matrix. This matrix $\mathbf{C}$ has dimensions $N \times M$ where N is the number of contenders and M is the number of levels required to complete the tree. However, when a tree is randomly generated, M cannot be known a priori. Therefore, C has to be generated progressively by concatenating new vectors of decisions in every level until every row is different. An example of this process is shown below:

$$\mathbf{C} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \longrightarrow \quad \mathbf{C} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \longrightarrow \quad \mathbf{C} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{5.2}$$

Once every row in $\mathbf{C}$ is different, the tree is known to be finished, since it is guaranteed that every device has followed a different path. If we compare the decisions matrix in 5.1 with the obtained in 5.2, we may notice that both describe the same tree, although they are different. Indeed, the matrix in 5.2 contains splitting decisions beyond the level in which the second device successfully transmitted. However, this extra decisions are useless, hence they can be safely skipped without affecting the final result.

Apart from the decisions matrix, two additional vectors (a vector with the level of successful transmission of each device and a vector with the number of collisions per level) and a decimal version of the decisions matrix are returned from this generation phase. Furthermore, some other temporary matrices and vectors will be used throughout this phase. A compilation of all of these variables is presented below:

- $\mathbf{C}$ is the aforementioned decisions matrix. It is formally defined as:

$$\mathbf{C} = [C_{i,j}], \qquad C_{i,j} = \text{node selected by the } i\text{-th device in the } j\text{-th level.} \tag{5.3}$$

- $\mathbf{D}$ is a matrix with the same dimensions as $\mathbf{C}$ whose elements are defined as follows:

$$\mathbf{D} = [D_{i,j}], \qquad D_{i,j} = (C_{i,1} \dots C_{i,j})_2. \tag{5.4}$$

  For example, using the matrix $\mathbf{C}$ as defined in 5.2, its related $\mathbf{D}$ would be:

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 5 \\ 0 & 1 & 3 \end{bmatrix}.$$

- $\mathbf{s}$ is a column vector containing the level of successful transmission of each device.

$$\mathbf{s} = [s_i], \qquad s_i = \text{level of successful transmission of the } i\text{-th device.} \tag{5.5}$$

- $\mathbf{x}$ is a row vector containing the number of collisions in each level.

$$\mathbf{x} = [x_j], \qquad x_j = \text{number of collisions in the } j\text{-th level.} \tag{5.6}$$

- $\mathbf{0}_{m\times n}$ is a zero matrix of m rows and n columns.
- $\mathbf{1}_{m\times n}$ is a matrix of ones with m rows and n columns.
- $\mathfrak{R}_{m\times n}$ is a matrix of m rows and n columns whose elements are randomly chosen between 0 and 1. For example:

$$\mathfrak{R}_{2\times 4} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

- $\mathfrak{T}_{m\times n}$ is a matrix of m rows and n columns whose elements are:

$$t_{ij} = 2^{m-i}. \tag{5.7}$$

  For example:

$$\mathfrak{T}_{3\times 2} = \begin{bmatrix} 4 & 4 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}.$$

- $\mathfrak{M}$ stands for the largest positive floating-point number, i.e. $\mathfrak{M} \approx 1.8 \cdot 10^{308}$.
- $\mathfrak{E}(\mathbf{a})$ is a function of the vector $\mathbf{a}$ that returns the set with the distinct elements in $\mathbf{a}$. For example:

$$\mathbf{a} = \begin{bmatrix} 2 & 8 & 2 & 8 & 4 & 3 \end{bmatrix},$$
$$\mathfrak{E}(\mathbf{a}) = \{2, 3, 4, 8\}.$$

- $\mathfrak{U}(\mathbf{a})$ is a function of the vector $\mathbf{a}$ that returns the set with the elements that appears *only once* in $\mathbf{a}$. For example:

$$\mathbf{a} = \begin{bmatrix} 2 & 8 & 2 & 8 & 4 & 3 \end{bmatrix},$$
$$\mathfrak{U}(\mathbf{a}) = \{3, 4\}.$$

- $\mathfrak{V}(\mathbf{a})$ is a function of the vector $\mathbf{a}$ that returns a set with the *index* of the elements that appears only once in $\mathbf{a}$. For example:

$$\mathbf{a} = \begin{bmatrix} 2 & 8 & 2 & 8 & 4 & 3 \end{bmatrix},$$
$$\mathfrak{V}(\mathbf{a}) = \{5, 6\}.$$

In addition to this variables, two special notations for matrix operations will be used:

1. The cardinality of the set A, i.e the number of elements in A, will be denoted by $\#A$.
2. The horizontal concatenation between matrices (or vectors) $\mathbf{A}$ and $\mathbf{B}$ will be denoted by $[\mathbf{A} \quad \mathbf{B}]$.

At this point we can proceed with the description of the algorithm behind the tree generation phase of the simulation, which is presented in Algorithm 1. In a nutshell, the algorithm uses a while loop to generate vectors of random decisions until all rows in $\mathbf{C}$ are different. At the same time, it keeps track of the number of collisions in each level and also saves the level of successful transmission of each device.

Algorithm 1 Tree generation phase of the simulator

function TREE_GENERATION_PHASE(N)
  $m \leftarrow 0$ ▷ Level index
  $\mathbf{C} \leftarrow \mathbf{0}^{N\times 0}, \quad \mathbf{d} \leftarrow \mathbf{0}^{N\times 0}, \quad \mathbf{D} \leftarrow \mathbf{0}^{N\times 0}$
  $\mathbf{s} \leftarrow \mathfrak{M} \cdot \mathbf{1}^{N\times 1}$
  $\mathbf{x} \leftarrow \mathbf{1}^{1\times 1}$
  while $\#\mathfrak{U}(\mathbf{d}) < N$ **do** ▷ While collisions exist
    $m \leftarrow m + 1$
    $\mathbf{C} \leftarrow \begin{bmatrix}\mathbf{C} & \mathfrak{R}^{N\times 1}\end{bmatrix}$ ▷ Generate random decisions
    $\mathbf{d} \leftarrow \mathbf{C} \times \mathfrak{T}^{m\times 1}$ ▷ Translate sequence of decisions into a decimal number
    $\mathbf{D} \leftarrow [\mathbf{D} \quad \mathbf{d}]$
    $s_i = \min(s_i, m) \quad \forall i \in \mathfrak{V}(\mathbf{d})$ ▷ Check for new successful transmissions
    if $\#\mathfrak{U}(\mathbf{d}) < N$ **then**
      $\mathbf{x} \leftarrow [\mathbf{x} \quad \#\mathfrak{E}(\mathbf{d}) - \#\mathfrak{U}(\mathbf{d})]$ ▷ Register collisions
    end if
  end while
return $\{\mathbf{C}, \mathbf{D}, \mathbf{s}, \mathbf{x}\}$
end function

In order to illustrate the operation of this algorithm, let us consider an example run for $N = 6$. The output might be as follows:

$$\mathbf{C} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1 & 3 & 7 & 15 & 31 & 63 \\ 1 & 2 & 4 & 8 & 17 & 35 \\ 0 & 0 & 0 & 1 & 3 & 7 \\ 1 & 2 & 5 & 10 & 21 & 42 \\ 0 & 0 & 0 & 1 & 3 & 6 \end{bmatrix},$$
$$\mathbf{s} = \begin{bmatrix} 2 \\ 3 \\ 6 \\ 3 \\ 6 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} 1 & 2 & 2 & 1 & 1 & 1 \end{bmatrix}. \tag{5.8}$$

If we draw the sequence of decisions contained in $\mathbf{C}$, we end up with the tree depicted in

the Figure 5.2. We can check in this picture that **s** and **x** correctly describe the number of collisions and the level of successful transmissions.

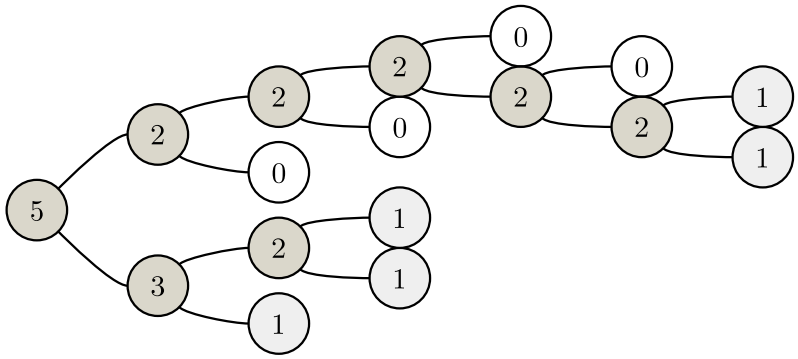


Figure 5.2: Tree defined by **C**

Offsets phase

At the end of the generation phase, we have created a random tree and extracted access delay metrics from it. With the information that we have, we can compute the total length of the tree, since we have the number of collisions in the tree. However, in order to calculate the access delay, we need to locate the successful transmissions within each level. In order to do so, in the offsets phase we will inspect again the tree to translate the information contained in **D** and **s** into a delay-related variable.

In summary, the operation of the algorithm of the offsets phase is as follows. For every device, it identifies those devices that were still active in the level in which the selected device successfully transmitted. Then, it detects empty nodes that could be in such level by taking advantage of the fact that nodes always come in pairs. Finally, the algorithm locates the selected node within the level, which will be used to compute its access delay. A detailed description of this algorithm is shown in Algorithm 2.

After the execution of this second phase, the offsets returned by the function Offsets for the example values of **D** and **s** shown in 5.8 would be the following:

$$\boldsymbol{\omega} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 4 \\ 1 \end{bmatrix}. \tag{5.9}$$

The meaning of this vector 5.9 is simple to understand if we observe Figure 5.2. Out of the five devices, two of them transmitted in the fourth node of their successful level and the rest transmitted in the third, second and first place. Once we know these delays, we

Algorithm 2 Calculate offsets

function OFFSETS_PHASE(N, $\mathbf{D}$, $\mathbf{s}$)
  $\boldsymbol{\omega} \leftarrow \mathbf{0}^{N\times 1}$
  for $i = 1..N$ **do**
    $A \leftarrow \{j \mid s_j \leq s_i\}$ ▷ Indexes of active contenders
    $Q_{ne} \leftarrow \{D_{j,s_i} \mid j \in A\}$ ▷ Nodes occupied by contenders
    $Q_{even} \leftarrow \{q \in Q_{ne} \mid q \mod 2 = 0\}$
    $Q_{odd} \leftarrow \{q \in Q_{ne} \mid q \mod 2 = 1\}$
    $Q^{req}_{even} \leftarrow \{q + 1 \mid q \in Q_{odd}\}$ ▷ This ensures that nodes appear in pairs
    $Q^{req}_{odd} \leftarrow \{q - 1 \mid q \in Q_{even}\}$
    $Q \leftarrow Q_{even} \cup Q_{odd} \cup Q^{req}_{even} \cup Q^{req}_{odd}$ ▷ Occupied and empty nodes
    $\omega_i \leftarrow \#\{q \in Q \mid q \leq D_{i,s_i}\}$ ▷ Offset of present contender
  end for
return $\boldsymbol{\omega}$
end function

know the value of the random variable Ω^m_N, and hence we are ready to compute the access delay.

Processing phase

Finally, after the completion of generation and offsets phases, we can derive the total length of the tree and the access delay of every device from $\mathbf{x}$ and $\boldsymbol{\omega}$. This is done in the processing phase, where the multi-channel aspect of the tree comes into play.

In brief, the total number of slots required by the tree is computed by adding the number of collisions divided by G for all levels. The access delay is calculated as the sum of the number of slots until the level of successful transmission and the offset experienced by each device, in slots. A more detailed description of the algorithm used in the processing phase can be found in Algorithm 3.

The final results of the example presented in 5.8 and 5.9 for $G = 1$ and $G = 2$ are the following:

$$G = 1 \longrightarrow L = 9,\ \boldsymbol{\theta} = \begin{bmatrix} 4 \\ 6 \\ 9 \\ 6 \\ 9 \end{bmatrix}. \qquad G = 2 \longrightarrow L = 7,\ \boldsymbol{\theta} = \begin{bmatrix} 3 \\ 4 \\ 7 \\ 4 \\ 7 \end{bmatrix}. \tag{5.10}$$

Algorithm 3 Calculate length and delay

function PROCESSING_PHASE(N, G, $\mathbf{x}$, $\boldsymbol{\omega}$)
  $\boldsymbol{\theta} \leftarrow \mathbf{0}^{N\times 1}$
  $L \leftarrow \sum_i \left\lceil \frac{x_i}{G} \right\rceil$
  for $i = 1..N$ **do**
    $\theta_i \leftarrow \sum_{k=1}^{i} \left\lceil \frac{x_k}{G} \right\rceil + \left\lceil \frac{\left\lceil \frac{\omega_k}{2} \right\rceil}{G} \right\rceil$
  end for
return $\{L, \boldsymbol{\theta}\}$
end function

5.1.2 Selection of the number of runs

In the section 4.3.2, an analytic expression of the PMF $p_{\Theta_N^m}(\theta_i)$ of the access delay was derived, namely in equations 4.98 and 4.90. From this PMF, computing the CDF $F_{\Theta_N^m}(\theta_m)$ is straightforward:

$$F_{\Theta_N^m}(\theta_m) = \sum_{\theta_i=1}^{\theta_m} p_{\Theta_N^m}(\theta_i). \tag{5.11}$$

From the results of the simulations, we can build an empirical CDF after n simulation runs. Let us denote by $\boldsymbol{\theta}^{runs}$ the vector of $N \cdot n$ elements containing the access delay of every device for all runs. We can define the empirical CDF of this data as:

$$\hat{F}^n_{\Theta_N^m}(\theta_m) = \frac{1}{Nn}\#\{\theta_i^{runs} \mid \theta_i^{runs} \leq \theta_m, \} \qquad \forall i \in \{1, \ldots, Nn\}. \tag{5.12}$$

When $n \to \infty$, the empirical CDF almost surely converges to the actual CDF of Θ_N^m, according to the strong law of large numbers:

$$\hat{F}^n_{\Theta_N^m}(\theta_m) \xrightarrow{a.s.} \tilde{F}_{\Theta_N^m}(\theta_m). \tag{5.13}$$

Since we have used approximations to compute $F_{\Theta_N^m}(\theta_m)$, the actual distribution $\tilde{F}_{\Theta_N^m}(\theta_m)$ may differ from $F_{\Theta_N^m}(\theta_m)$. In fact, simulations will be used to measure the goodness of fit of $F_{\Theta_N^m}(\theta_m)$ over the actual data, in order to confirm or reject its usefulness.

Therefore, we need a good estimation of $\tilde{F}_{\Theta_N^m}(\theta_m)$ in order to compare it with the analytical approximation. This implies that the number of runs n needs to be high, although the higher n the higher the time and resources that our simulator will require. Thus, we would like to figure out an optimum value of n, barely high enough to meet some required confidence level.

In order to do so, we will use the Dvoretzky–Kiefer–Wolfowitz (DKW) inequality [Kos], defined as follows:

$$\alpha = \Pr\left[\max\left|\hat{F}^{n}_{\Theta^{m}_{N}}(\theta_m) - \tilde{F}_{\Theta^{m}_{N}}(\theta_m)\right| > \varepsilon\right] \leq 2e^{-2n'\varepsilon^2}, \qquad \text{for every } \varepsilon > 0. \tag{5.14}$$

In words, this inequality states that the probability α that the maximum difference between the theoretical and the empirical CDFs is higher than ε is upper bounded by a function of the number of samples n' and ε. Therefore, for some selected values of α and ε, we can compute the number of required samples as follows:

$$n' = \frac{1}{2\varepsilon^2}\ln\left(\frac{2}{\alpha}\right). \tag{5.15}$$

For instance, if we want our empirical CDF to differ at most $\varepsilon = 1\%$ from the actual CDF with a probability of $1 - \alpha = 99\%$ in, the required number of samples will be:

$$n' = 26492. \tag{5.16}$$

If we are only interested in computing the CDF of the access delay, the number of required runs n is equal to the number of required samples n' divided by the number of contenders N, as in every run N different access delays are yielded by the simulator. However, if we are also interested in the CDF of the length of the tree $\hat{F}_{\mathcal{T}_N}(t)$, the number of runs has to be equal to the number of required samples $n = n'$, since only one tree is resolved in each run.

Once we know the number of required runs, we can put the three simulation phases together to obtain the complete simulation procedure, which is described in Algorithm 4.

Algorithm 4 Complete simulation procedure

Require: $N \geq 2$, $G \geq 1$, α, ε

$n_{max} \leftarrow \frac{1}{2\varepsilon^2}\ln\left(\frac{2}{\alpha}\right)$ ▷ Compute the required number of runs
$\mathbf{L}^{runs} \leftarrow \mathbf{0}_{1\times n_{max}}$
$\boldsymbol{\theta}^{runs} \leftarrow \mathbf{0}_{1\times N n_{max}}$
for $n = 1..n_{max}$ **do**
    $\{\mathbf{C}, \mathbf{s}, \mathbf{x}\} \leftarrow$ GENERATION_PHASE(N)
    $\boldsymbol{\omega} \leftarrow$ OFFSETS_PHASE($N, \mathbf{s}$)
    $\{L, \boldsymbol{\theta}\} \leftarrow$ PROCESSING_PHASE(N,G,$\mathbf{x}$,$\boldsymbol{\omega}$)
    $L^{runs}_{n} \leftarrow L$ ▷ Accumulate samples of the tree length
    $\left[\theta^{runs}_{1,n(N-1)+1} \ldots \theta^{runs}_{1,nN}\right] \leftarrow \boldsymbol{\theta}$ ▷ Accumulate samples of the access delay
end for

5.2 Simulation results

The simulator described in the previous section was implemented in MATLAB. The selected parameters were $N = 60$ contenders and $n = 26500$ runs, as explained in the previous section.

In the Figure 5.3, the theoretical and the empirical PMFs of the length of the tree for $G = \{1, ..., 8\}$ are plotted together for comparison. For $G = 1$, we see a slight but noticeable difference between the approximate model and the actual results, as it was indeed predicted in Figure 4.14. Nevertheless, the accuracy of the analytical model seems to improve rapidly when G increases.

In the Figure 5.4, the theoretical and the empirical PMFs of the access delay for $G = \{1, ..., 8\}$ are also plotted together. In this case, the fit seems better than for the length of the tree. Nonetheless, for $G = 1$ the predicted and the actual result differ slightly, although it is barely noticeable. For the remaining values of G, it can be observed that the model becomes more accurate when G increases. Thus, one may conclude the Markovian approximation is valid and yields accurate approximations.

Finally, in Figures 5.5 and 5.6, the CDFs of the length of the tree and the access delay are presented, respectively. The accuracy of the approximate model is confirmed in these figures, since the difference between the predicted and the simulated probabilities seems negligible.

Apart from the validity of the model, conclusions about the values of the access delay may be drawn as well, now that the predicted values are backed with simulations. Regarding access delay, we see how the maximum access delay might be lowered up to 20 slots, which in the case of the LTE RACH would be equal to 20 RAOs, or 100 ms assuming one RAO every 5 ms. We can compare this value with the results of a single channel Tree Algorithm, which is obtained after multiplying by two the result for $G = 1$, as it was stated in equation 4.86. According again to Figure 5.6, a device in a single channel Tree Algorithm could experience a delay up to 200 slots, or 1 s when applied to LTE RACH. Hence, a tenfold reduction of the access delay can be achieved if $G = 8$, and larger reductions are possible is $G \geq 8$. Nevertheless, the higher G the lower the number of trees that can be executed in parallel if the number of channels is limited, therefore the optimum value of G needs to be carefully chosen depending on the application.

Although visual inspection of the the aforementioned figures seems to approve the validity the model, some measures are still necessary to be aware of the significance of the errors. In order to measure the goodness of fit between the approximate model and the simulation results, the Kolmogorov–Smirnov statistic [MJ51] will be applied. This statistic is often employed to perform the Kolmogorov–Smirnov test, which is used to check whether an empirical CDF matches an hypothetical theoretical CDF. Although this appears similar to our situation, it would be pointless to use the complete Kolmogorov-Smirnov test in the present case, since we already know that our theoretical CDF is just an approximation to

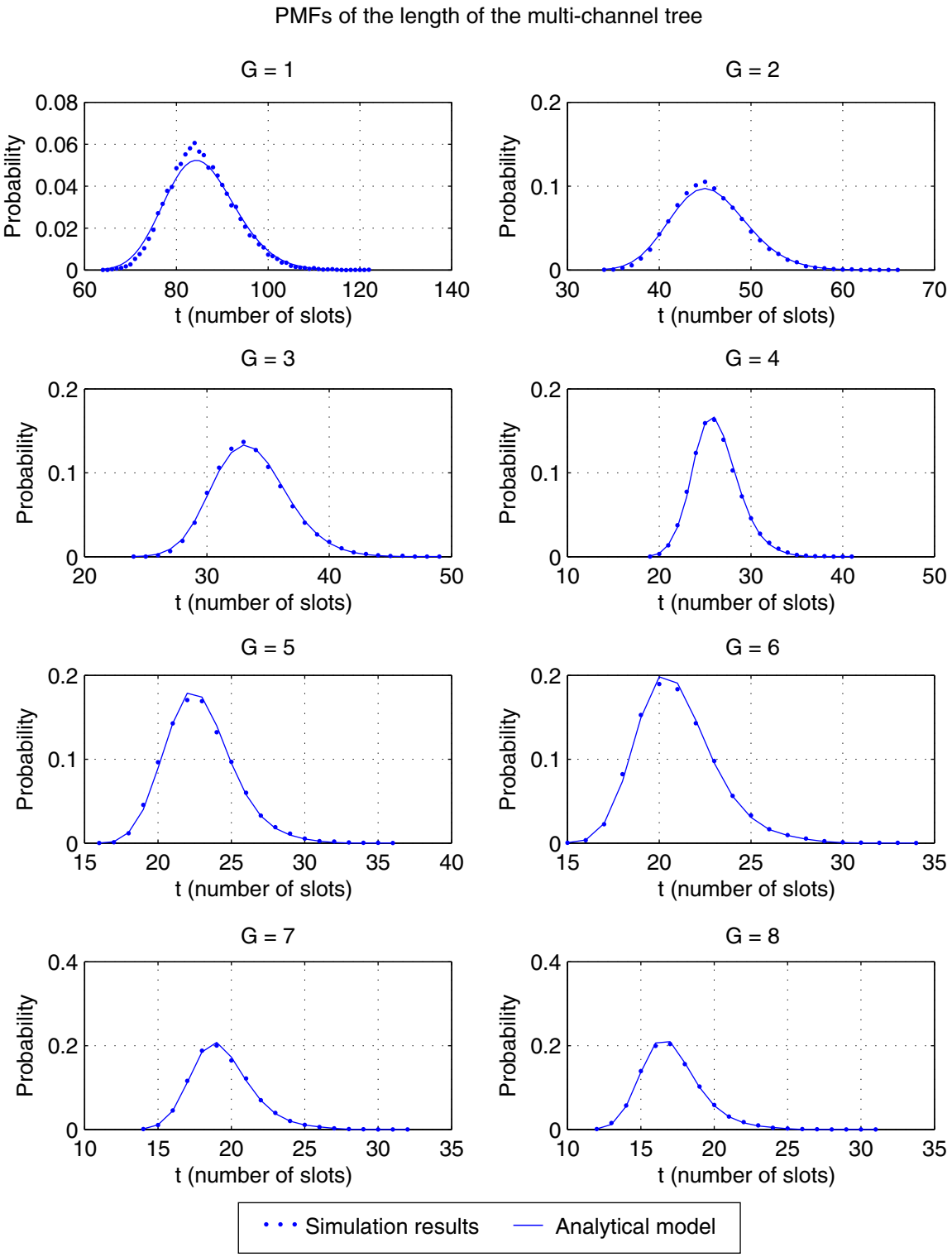


Figure 5.3: PMF of the simulation results of the length of the tree

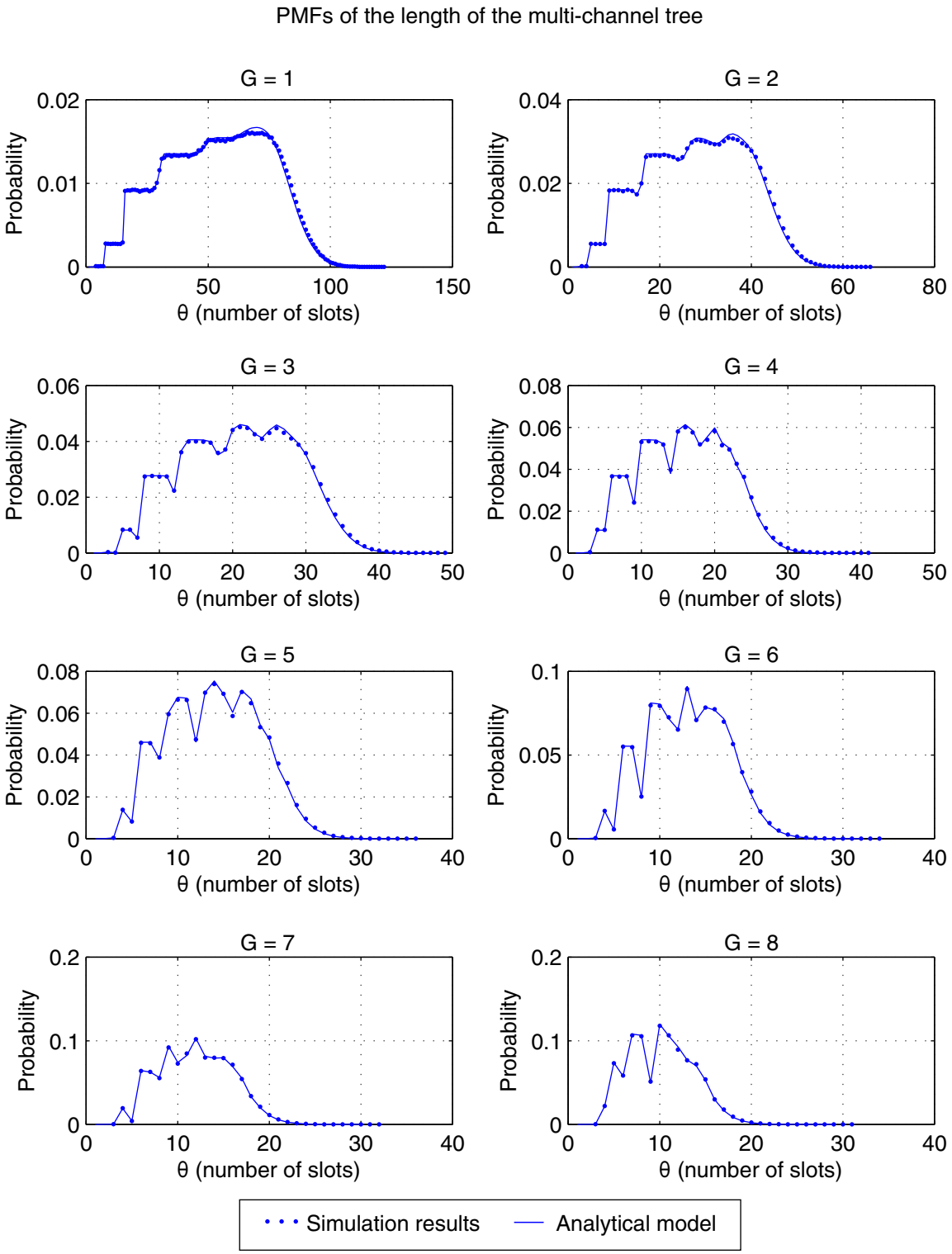


Figure 5.4: PMFs of the simulation results of the access delay.

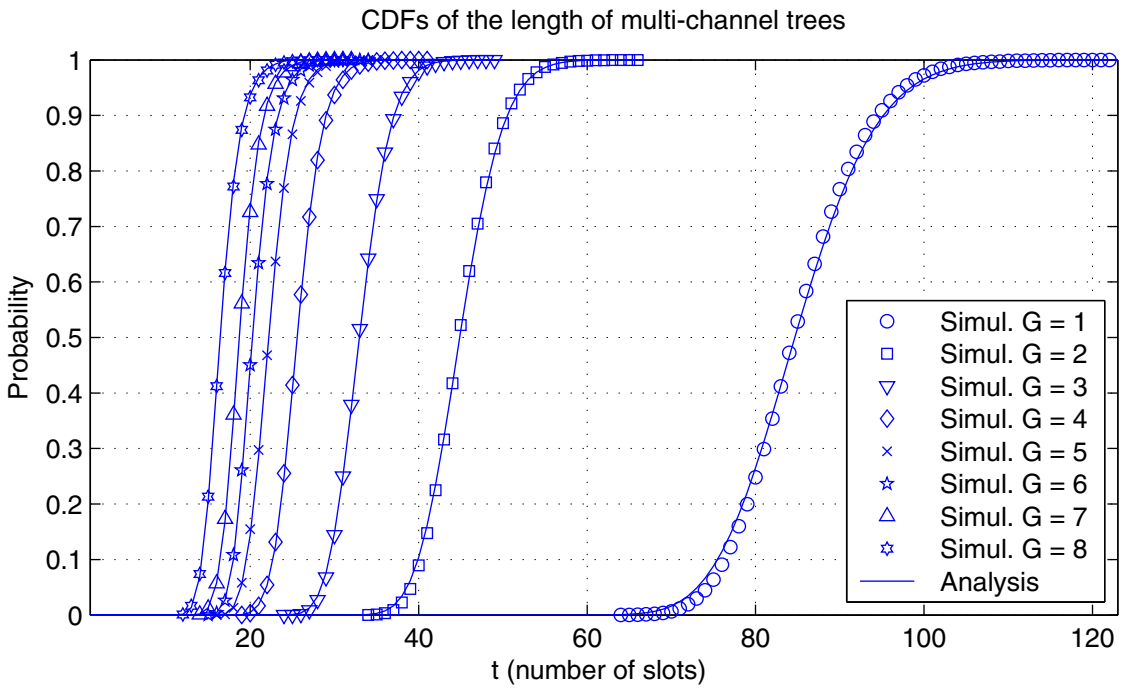


Figure 5.5: CDFs of the simulation results and analytical models of the length of the tree

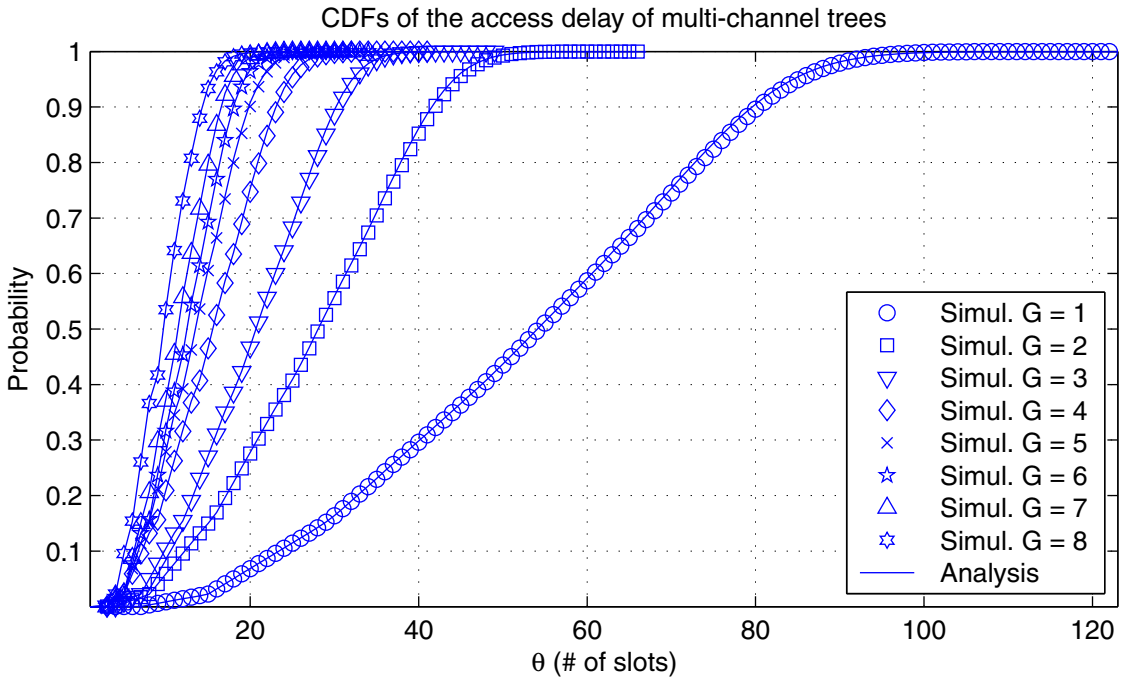


Figure 5.6: CDFs of the simulation results and analytical models of the access delay

the actual CDF. Therefore, the test will surely fail given a high enough number of samples. Nonetheless, the Kolmogorov–Smirnov statistic alone can be still used as a metric of the goodness of fit.

Given an empirical CDF of the access delay $\hat{F}^n_{\Theta^m_N}(\theta_m)$ and an analytical CDF $F_{\Theta^m_N}(\theta_m)$, the Kolmogorov–Smirnov statistic $\mathfrak{D}$ is defined as:

$$\mathfrak{D} = \max \left| \hat{F}^n_{\Theta^m_N}(\theta_m) - F_{\Theta^m_N}(\theta_m) \right| . \tag{5.17}$$

In words, $\mathfrak{D}$ is simply the maximum difference between the empirical and the theoretical CDFs. As a consequence, in our case it will also be the maximum error that we could expect from our approximation, This statistic may seem biased, since only the worst point of the CDFs is considered, regardless of the goodness of the remaining points. However, as we are modeling an algorithm that copes with delay-sensitive devices, we are indeed interested in the maximum error of our prediction rather than in the average or some other 'smoother' statistic.

A table with values of $\mathfrak{D}$ for different values of G is presented in Table 5.2. One can observe from this table that the maximum difference between the predicted and the actual probability of any access delay is lower than 1% for $G > 1$. For most applications, this margin of error should be acceptable, therefore we can finally conclude that our model is accurate, even though it is still an approximation.

G	1	2	3	4	5	6	7	8
$\mathfrak{D}$	0.0108	0.0090	0.0096	0.0081	0.0082	0.0057	0.0042	0.0061

Table 5.1: Kolmogorov–Smirnov statistic for different values of G

Chapter 6

Conclusions and Outlook

In this thesis, the problem of guaranteeing reliability of mobile networks under massive MTC arrivals is addressed. In order to do so, a new approach for improving the access delay statistics of the current specification of the Random Access Channel in LTE is presented.

The proposed approach is based on the application of a type of collision resolution algorithms, called Tree Algorithms, to the MAC layer of the Random Access Channel in LTE. Preliminary simulations showed promising results for this approach with respect to the current performance of LTE, specially for multi-channel implementations of Tree Algorithms. Nonetheless, knowledge about the behavior of Tree Algorithms in multi-channel scenarios is required to provide a guided use of the proposed approach. Therefore, an analytical model of multi-channel Tree Algorithms is required.

In order to derive this analytical model, the analysis of single channel is presented first, along with necessary extensions. Then, an analytical model of the statistics of the length of a multi-channel tree is obtained. With the intention of overcoming the analytical complexity, an approximation based on the Markov property is used.

Based on the model of the length of the tree, an analytical model of the statistics of the access delay of a multi-channel Tree Algorithm is finally derived. According to that model, multi-channel Tree Algorithms may outperform single channel Tree Algorithms, in exchange for more resource usage, as it would be intuitively expected. Hence, this model provides a way to cope with delay sensitive applications by means of multi-channel implementations.

Finally, in order to confirm the validity of the model, simulations of multi-channel Tree Algorithms are performed. The results of the simulations accurately match those of the analytical model, with a maximum error in the CDF lower than 1%.

Future work could include the implementation of different configurations of multi-channel Tree Algorithms for different traffic classes. Furthermore, guaranteed reliability might be provided via collision estimation, so that the implementation of Tree Algorithms would be

optimally selected according to that estimation. Lastly, Ternary or Q-ary multi-channel Tree Algorithms could be tested, instead of the Binary Tree Algorithm that was analyzed in this work.

Appendix A

Derivation of a recursive expression for $\xi_{i,x_m}^{R,N}$

In equation 4.48, we defined $\xi_{i,x_m}^{R,N}$ as follows:

$$\xi_{i,x_m}^{R,N} \triangleq \frac{\Psi_{i,x_m}^{N}\binom{R}{i}i!}{R^N}. \tag{4.48}$$

Every term in equation 4.48 can be written as a recursion.

$$\Psi_{i,j}^{N} = j\Psi_{i,j}^{N-1} + (i-j+1)\,\Psi_{i,j-1}^{N-1} + \Psi_{i-1,j}^{N-1}, \tag{4.46}$$

$$i! = i\cdot(i-1)!, \tag{4.49}$$

$$\binom{R}{i} = \left(\frac{R+1}{i} - 1\right)\cdot\binom{R}{i-1}, \tag{4.50}$$

$$R^N = R\cdot R^{N-1}. \tag{4.51}$$

Hence, if we combine all these recursions we may obtain a general recursive expression for $\xi_{i,x_m}^{R,N}$. In order for the derivation of this recursion to be easier to understand, let us proceed progressively. At first, we will compute the recursion of the combination of equations 4.46 and 4.49, denoted by $A_{i,j}^{N}$.

$$\begin{aligned} A_{i,j}^{N} = \Psi_{i,j}^{N}\cdot i! &= j\Psi_{i,j}^{N-1}\cdot i! + (i-j+1)\,\Psi_{i,j-1}^{N-1}\cdot i! + i\Psi_{i-1,j}^{N-1}\cdot(i-1)! \\ &= jA_{i,j}^{N-1} + (i-j+1)\,A_{i,j-1}^{N-1} + iA_{i-1,j}^{N-1}. \end{aligned} \tag{A.1}$$

The next step is to combine A.1 and 4.50, which we will denote by $B_{i,j}^{R,N}$:

$$\begin{aligned} B_{i,j}^{R,N} = A_{i,j}^{N}\cdot\binom{R}{i} &= jA_{i,j}^{N-1}\cdot\binom{R}{i} + (i-j+1)\,A_{i,j-1}^{N-1}\cdot\binom{R}{i} \\ &\quad + i\left(\frac{R+1}{i}-1\right)\cdot\binom{R}{i-1}A_{i-1,j}^{N-1} \\ &= jB_{i,j}^{R,N-1} + (i-j+1)\,B_{i,j-1}^{R,N-1} + (R-i+1)\,B_{i-1,j}^{R,N-1}. \end{aligned} \tag{A.2}$$

Finally, after combining A.2 and 4.51 we obtain our final expression:

$$\begin{aligned}\xi^{R,N}_{i,j} = \frac{B^{R,N}_{i,j}}{R^N} &= \frac{j}{R}\frac{B^{R,N-1}_{i,j}}{R^{N-1}} + \frac{i-j+1}{R}\frac{B^{R,N-1}_{i,j-1}}{R^{N-1}} + \frac{R-i+1}{R}\frac{B^{R,N-1}_{i-1,j}}{R^{n-1}} \\ &= \frac{1}{R}\left[j\xi^{R,N-1}_{i,j} + (i-j+1)\xi^{R,N-1}_{i,j-1} + (R-i+1)\xi^{R,N-1}_{i-1,j}\right]. \qquad \text{(A.3)}\end{aligned}$$

In order to completely define this recursion, we have to describe the initial conditions:

- If $N = i$ and $j = 0$:

$$\Psi^N_{N,0} = 1,$$

$$\xi^{R,N}_{N,0} = \frac{\Psi^N_{N,0}\binom{R}{N}N!}{R^N} = \frac{R!}{R^N(R-N)!} = \prod_{i=0}^{N-1}\frac{R-i}{R}.$$

- If $N = i$ and $j > 0$:

$$\Psi^N_{N,j>0} = 0,$$

$$\xi^{R,N}_{N,j>0} = \frac{\Psi^N_{N,j>0}\binom{R}{N}N!}{R^N} = 0.$$

- If $N = 1$, $i = 1$ and $j = 0$:

$$\Psi^1_{1,0} = 1,$$

$$\xi^{R,1}_{1,0} = \frac{\Psi^1_{1,0}\binom{R}{1}1!}{R^1} = 1.$$

- If $N = 1$, $i = 1$ and $j = 1$:

$$\Psi^1_{1,1} = 0,$$

$$\xi^{R,1}_{1,1} = \frac{\Psi^1_{1,1}\binom{R}{1}1!}{R^1} = 0.$$

- If $N > 1$, $i = 1$ and $j = 1$:

$$\Psi^{N>1}_{1,1} = 1,$$

$$\xi^{R,N>1}_{1,1} = \frac{\Psi^{N>1}_{1,1}\binom{R}{1}1!}{R^N} = R^{1-N}.$$

With this, the derivation of the recursive expression for $\xi^{R,N}_{i,x_m}$ is complete.

Appendix B

Derivation of the probability of a given partition

In equation 4.75, the probability to obtain a partition π_i from a random distribution of k_{m-1} balls over x_{m-1} bins was presented as:

$$p_{\mathcal{P}_{x_{m-1}}^{k_{m-1}}}(\pi_i) = \frac{k_{m-1}!}{\Psi_{x_{m-1},x_{m-1}}^{k_{m-1}}} \prod_{j=1}^{x_{m-1}} \frac{1}{\kappa_j^i!\#_j^i!}. \tag{4.75}$$

In this Appendix, the derivation of this expression will be tackled. First, let us simplify the notation, so that we denote the number of balls by a and the number of bins by b. The probability to obtain a partition π of a with b parts is:

$$p_{\mathcal{P}_b^a}(\pi) = \frac{a!}{\Psi_{b,b}^a} \prod_{j=1}^{b} \frac{1}{\kappa_j!\#_j!}, \tag{B.1}$$

where $\pi = (\kappa_1, \kappa_2, \ldots, \kappa_b)$ and $\#_c$ is defined as:

$$\#_c = \sum_j \delta\left[\kappa_j - c\right], \tag{B.2}$$

where $\delta\left[\cdot\right]$ is the Kroenecker delta. In words, $\#_c^i$ is the number of occurrences of c within the partition π_i.

Once all variables are defined, we can start with the derivation. Let us start by computing the number of ways $W_a^{\kappa_1}$ to choose κ_1 balls out of a total of a balls:

$$W_a^{\kappa_1} = \binom{a}{\kappa_1}. \tag{B.3}$$

After this selection is done, the number of ways $W_a^{\kappa_2}$ to choose κ_2 balls out of a total of $a - \kappa_1$ balls is:

$$W_{a-\kappa_1}^{\kappa_2} = \binom{a-\kappa_1}{\kappa_2}. \tag{B.4}$$

In general, the number of ways $W_a^{\kappa_n}$ to choose κ_n balls out of a total of $a - \sum_{i=1}^{n-1} \kappa_i$ balls is:

$$W_a^{\kappa_n} = \binom{a - \sum_{i=1}^{n-1} \kappa_i}{\kappa_n}. \tag{B.5}$$

If every part κ_n in the partition π is different, the total number of ways W^π to generate such partition is simply:

$$W^\pi = \prod_{j=1}^{b} W_a^{\kappa_j} = \prod_{j=1}^{b} \binom{a - \sum_{i=1}^{j-1} \kappa_i}{\kappa_j}. \tag{B.6}$$

After some basic manipulation based on the definition of the binomial coefficient, we can rewrite B.6 as:

$$W^\pi = a! \prod_{j=1}^{b} \frac{1}{\kappa_j!}. \tag{B.7}$$

Nevertheless, if some parts have the same value, e.g. $10 = 4+4+2$, the number of ways to select those parts would be counted multiple times, yielding an incorrect result. In order to solve this issue, we have to correct by the number of ways to arrange those repeated values:

$$W^\pi = a! \prod_{j=1}^{b} \frac{1}{\kappa_j! \#_j!}. \tag{B.8}$$

Finally, the probability of partition π is obtained by dividing the number of ways W^π to generate that specific partition by the total number of ways to generate any partition, which is given by $\Psi_{b,b}^a$, i.e. the number of ways to arrange a balls in b groups, b of which have more than one ball. Therefore, we have reached our final result:

$$p_{\mathcal{P}_b^a}(\pi) = \frac{a!}{\Psi_{b,b}^a} \prod_{j=1}^{b} \frac{1}{\kappa_j! \#_j!}. \tag{B.1}$$

List of Figures

List of Tables

Appendix C

Notation und Abkürzungen

3GPP	3rd Generation Partnership Project
ACB	Access Class Barring
BTA	Binary Tree Algorithm
CDF	Cumulative Distribution Function
DTA	Dynamic Tree Algorithm
EAB	Extended Access class Barring
eNodeB	evolved Node B or E-UTRAN Node B
FFT	Fast Fourier Transform
HTC	Human Type Communications
LTE	Long Term Evolution
MAC	Medium Access Control
MTC	Machine Type Communications
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PGF	Probability-Generating Function
PMF	Probability Mass Function
PRACH	Physical Random Access Channel
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QTA	Q-ary Tree Algorithm
SIB	System Information Block
RACH	Random Access Channel
RAO	Random Access Opportunity
RAR	Random Access Response
RRC	Radio Resource Control
TA	Tree Algorithm
UE	User Equipment

Bibliography

[3GP10] 3GPP. MTC simulation results with specific solutions. TR R2-104662, 3rd Generation Partnership Project (3GPP), August 2010.

[3GP14] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation. TS 136.211, 3rd Generation Partnership Project (3GPP), October 2014.

[3GP15a] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification. TS 136.321, 3rd Generation Partnership Project (3GPP), April 2015.

[3GP15b] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures. TS 136.213, 3rd Generation Partnership Project (3GPP), February 2015.

[3GP15c] 3GPP. Service accessibility. TS 22.011, 3rd Generation Partnership Project (3GPP), June 2015.

[3GP16] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification. TS 36.331, 3rd Generation Partnership Project (3GPP), June 2016.

[AHK16] Md Shipon Ali, Ekram Hossain, and Dong In Kim. Lte/lte-a random access for massive machine-type communications in smart cities. *arXiv preprint arXiv:1608.03042*, 2016.

[AZKC+08] Jesus Alonso-Zárate, Elli Kartsakli, Alex Cateura, Christos Verikoukis, and Luis Alonso. Contention-based collision-resolution medium access control algorithms. In Hongyi Wu and Yi Pan, editors, *Medium Access Control in Wireless Networks*, Wireless networks and mobile computing, pages 40–64. Nova Science Publishers, 2008.

[Cap79a] John Capetanakis. Generalized tdma: The multi-accessing tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, 1979.

[Cap79b] John Capetanakis. Tree algorithms for packet broadcast channels. *IEEE transactions on information theory*, 25(5):505–515, 1979.

[CCCW15] R. G. Cheng, J. Chen, D. W. Chen, and C. H. Wei. Modeling and analysis of an extended access barring algorithm for machine-type communications in lte-a networks. *IEEE Transactions on Wireless Communications*, 14(6):2956–2968, June 2015.

[ChLL11] J. P. Cheng, C. h. Lee, and T. M. Lin. Prioritized random access with dynamic access barring for ran overload in 3gpp lte-a networks. In *2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pages 368–372, Dec 2011.

[Cox14] C. Cox. *An Introduction to LTE: LTE, LTE-Advanced, SAE, VoLTE and 4G Mobile Communications*. Wiley, 2014.

[CW10] Y. Chen and W. Wang. Machine-to-machine communication in lte-a. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–4, Sept 2010.

[DPS13] E. Dahlman, S. Parkvall, and J. Skold. *4G: LTE/LTE-Advanced for Mobile Broadband*. Elsevier Science, 2013.

[DSEAB+] Icaro Da Silva, Salah Eddine El Ayoubi, Orange Mauro Boldi, Ömer Bulakci, Panagiotis Spapis, Malte Schellmann, Jose F Monserrat, Thomas Rosowski, Gerd Zimmermann, Deutsche Telekom, et al. 5g ran architecture and functional design.

[DSMW13] S. Duan, V. Shah-Mansouri, and V. W. S. Wong. Dynamic access class barring for m2m communications in lte networks. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 4747–4752, Dec 2013.

[Eri11] Ericsson. More than 50 billion connected devices. *White Paper*, Feb 2011.

[FA15] L. Ferdouse and A. Anpalagan. A dynamic access class barring scheme to balance massive access requests among base stations over the cellular m2m networks. In *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*, pages 1283–1288, Aug 2015.

[FI13] G. Farhadi and A. Ito. Group-based signaling and access control for cellular machine-to-machine communication. In *Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th*, pages 1–6, Sept 2013.

[Gal78] Robert G Gallager. Conflict resolution in random access broadcast networks. In *Proc. of the AFOSR Workshop in Communication Theory and Applications*, pages 74–76, 1978.

[JdJ00] Augustus JEM Janssen and MJ de Jong. Analysis of contention tree algorithms. *IEEE Transactions on Information Theory*, 46(6):2163–2172, 2000.

[JKK+14] H. S. Jang, S. M. Kim, K. S. Ko, J. Cha, and D. K. Sung. Spatial group based random access for m2m communications. *IEEE Communications Letters*, 18(6):961–964, June 2014.

[KG85] Michael Kaplan and Eugene Gulko. Analytic properties of multiple-access trees. *IEEE Transactions on Information Theory*, 31(2):255–263, 1985.

[Kos] MR Kosorok. Introduction to empirical processes and semiparametric inference. 2008.

[LAAZ14] A. Laya, L. Alonso, and J. Alonso-Zarate. Is the random access channel of lte and lte-a suitable for m2m communications? a survey of alternatives. *IEEE Communications Surveys Tutorials*, 16(1):4–16, First 2014.

[LLCC14] T. M. Lin, C. H. Lee, J. P. Cheng, and W. T. Chen. Prada: Prioritized random access with dynamic access barring for mtc in 3gpp lte-a networks. *IEEE Transactions on Vehicular Technology*, 63(5):2467–2472, Jun 2014.

[LLJK11] Anthony Lo, Yee Wei Law, Martin Jacobsson, and Michal Kucharzak. Enhanced lte-advanced random-access mechanism for massive machine-to-machine (m2m) communications. In *27th World Wireless Research Forum (WWRF) Meeting*, pages 1–5. Dusseldorf, Germany Dusseldorf, Germany, 2011.

[LLKC12] Shao-Yu Lien, Tzu-Huan Liau, Ching-Yueh Kao, and Kwang-Cheng Chen. Cooperative access class barring for machine-to-machine communications. *IEEE Transactions on Wireless Communications*, 11(1):27–32, January 2012.

[Mas14] James L. Massey. Collision-resolution algorithms and random-access communications. In G. Longo, editor, *Multi-User Communication Systems*, CISM International Centre for Mechanical Sciences. Springer Vienna, 2014.

[MF85] Peter Mathys and Philippe Flajolet. Q-ary collision resolution algorithms in random-access systems with free or blocked channel access. *IEEE Transactions on Information Theory*, 31(2):217–243, 1985.

[MJ51] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[MP93] Mart L Molle and George C Polyzos. Conflict resolution algorithms and their performance analysis. *Tec. Rep. Department of Computer Science and Engineering, University of California at San Diego, LaJolla*, 1993.

[nPSP15] G. C. Madue no, N. K. Pratas, Ĉ Stefanović, and P. Popovski. Massive m2m access with reliability guarantees in lte systems. In *2015 IEEE International Conference on Communications (ICC)*, pages 2997–3002, June 2015.

[nSP14] G. C. Madue no, Č Stefanović, and P. Popovski. Efficient lte access with collision resolution for massive m2m communications. In *Globecom Workshops (GC Wkshps), 2014*, pages 1433–1438, Dec 2014.

[OBB+14] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, H. Tullberg, M. A. Uusitalo, B. Timus, and M. Fallgren. Scenarios for 5g mobile and wireless communications: the vision of the metis project. *IEEE Communications Magazine*, 52(5):26–35, May 2014.

[PTSP12] N. K. Pratas, H. Thomsen, Č Stefanović, and P. Popovski. Code-expanded random access for machine-type communications. In *2012 IEEE Globecom Workshops*, pages 1681–1686, Dec 2012.

[Rug81] Gabriel Ruget. Some tools for the study of channel-sharing algorithms. In *Multi-user communication systems*, pages 201–231. Springer, 1981.

[SLD+15] Mahyar Shirvanimoghaddam, Yonghui Li, Mischa Dohler, Branka Vucetic, and Shulan Feng. Probabilistic rateless multiple access for machine-to-machine communication. *IEEE Transactions on Wireless Communications*, 14(12):6815–6826, 2015.

[Tan03] A.S. Tanenbaum. *Computer Networks*. Number p. 3 in Computer Networks. Prentice Hall PTR, 2003.

[TM78] Boris Solomonovich Tsybakov and Viktor Alexandrovich Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Informatsii*, 14(4):32–59, 1978.

[TM80] Boris Solomonovich Tsybakov and Viktor Alexandrovich Mikhailov. Random multiple packet access: part-and-try algorithm. *Problemy Peredachi Informatsii*, 16(4):65–79, 1980.

[TR311] 3GPP TR 37.868: Technical Specification Group Radio Access Network; Study on RAN Improvements for Machine-type Communications, 2011.

[WBC15] Chia-Hung Wei, Giuseppe Bianchi, and Ray-Guang Cheng. Modeling and analysis of random access channels with bursty arrivals in ofdma wireless networks. *IEEE Transactions on Wireless Communications*, 14(4):1940–1953, 2015.