
ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



Master Thesis

Speed Dating despite Jammers

Dominic Pascal Meier
meierdo@student.ethz.ch

Prof. Dr. Roger Wattenhofer
Distributed Computing Group

Advisors: Yvonne Anne Oswald & Stefan Schmid

Dept. of Computer Science
Swiss Federal Institute of Technology (ETH) Zurich
Summer 2008

Abstract

We study the fundamental problem of how devices, operating over a wireless medium, can discover potential communication partners. For the devices, the spectrum is divided into m frequency channels. As multiple devices share these channels, they have to cope with collisions while they are trying to find other devices in the system. Moreover, we assume an adversarial jammer to block some of the available channels. The devices cannot distinguish between involuntary collisions, jamming, and empty channels, i.e. channels on which no device is currently transmitting its information.

We construct algorithms efficiently solving the described discovery problem even if the number of jammed channels is unknown to the devices. Moreover, we discuss several jammer models and compare our algorithms to the inquiry algorithm of Bluetooth in simulations of different scenarios.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 4 |
| 2 | Related Work | 6 |
| 3 | Model | 8 |
| 3.1 | Setting | 8 |
| 3.2 | Jammer | 9 |
| 3.3 | Objective | 10 |
| 4 | Algorithms | 11 |
| 4.1 | Deterministic Algorithms | 11 |
| 4.2 | Randomized Algorithms | 12 |
| 4.2.1 | 3 Classes | 14 |
| 4.2.2 | $\log(m)$ Classes | 16 |
| 4.2.3 | $m/2$ Classes | 17 |
| 4.2.4 | k Classes | 18 |
| 4.3 | Comparison | 20 |
| 4.4 | Classes vs. Probability Distributions | 21 |
| 4.4.1 | Uniform Distribution | 22 |
| 4.4.2 | Binomial Distribution | 23 |
| 4.4.3 | Poisson Distribution | 24 |
| 4.4.4 | Geometric Distribution | 24 |

| | |
|--|-----------|
| CONTENTS | 3 |
| 4.4.5 Exponential Distribution | 25 |
| 4.4.6 Conclusion | 26 |
| 4.5 Distribution Of The Number Of Blocked Channels | 26 |
| 5 Multiplayer | 28 |
| 6 Jammer Models | 30 |
| 6.1 Pure Senders | 30 |
| 6.1.1 Static Jammer | 30 |
| 6.1.2 Oblivious Jammer | 31 |
| 6.1.3 Byzantine Jammer | 31 |
| 6.1.4 Budget Jammer | 32 |
| 6.2 Listening Senders | 32 |
| 6.2.1 Ability To Notice Senders | 33 |
| 6.2.2 Ability To Notice Senders And Receivers | 33 |
| 7 Simulations | 35 |
| 8 Conclusion | 40 |
| A Coupon Collector Problem | 42 |

1

Introduction

Not only in computer science but also in real life one of the most important tasks, if not the most important task at all, is to process information. However, as it is often infeasible or even impossible to generate all information of interest by oneself, it is indispensable to exchange information with other entities, such as humans or devices of any kind.

Focusing on computer science, artificial devices have emerged to entities which are able to autonomously communicate with each others. Moreover, interactions between devices have become a daily matter, often without humans to even realize them. As an example, we would like to mention sensor nodes, i.e. tiny devices which gather and share information about their environment.

Especially for such devices it is of course infeasible to transmit information through wires of any kind. Also in applications for which the devices have to be able to move around, the only possibility to share information is to transmit it wirelessly. As the number of such applications is growing quickly, wireless communication gets more and more important.

However, there are several problems with wireless communication, e.g. reliability and security matters. Another problem is that wireless communication takes place over a shared medium. Hence, if multiple devices are communicating at the same time and in the same range, they may disturb each others by creating collisions. Bluetooth, a protocol especially designed for wireless communication between different devices, faces this problem by spreading its communication over multiple frequencies (i.e. 79 channels of 1MHz width each) and quickly hopping between these channels. Doing so, high traffic load which is caused by other devices on some channels affects the communication for short periods and relatively seldomly only. Moreover, the technique

of hopping between multiple frequencies also serves as a countermeasure to security issues, as adversaries can hardly follow a whole transmission.

Of course, collisions may not only result from other devices trying to communicate themselves, but also from malicious devices, i.e. adversarial jammers. Much less research has been conducted so far on this kind of intentionally produced collisions than on involuntary collisions as described above, even though there exist several examples for jammers which are applied in practice. For example, theaters and libraries sometimes use jammers to prevent mobile phones from establishing a connection to the radio network, thus forcing them to be quiet as no incoming and outgoing calls can be made.

A problem which arises at the beginning of every communication is that the devices might be unaware of the presence of their communication partners. Hence, there has to be some mechanism to detect other devices and, of course, such a mechanism has to find other devices quickly, provided that there are indeed potential communication partners around.

Much effort has been put into the research on communication despite jammers, however, the also extremely important task of discovering other devices despite jammers in order to be able to start to communicate has been widely neglected.

In this thesis we address the problem of connection establishment under the influence of jammers. That is, we derive efficient algorithms to be used by the devices to elect the channels on which they try to find potential communication partners. The term ‘efficient’ means that even if we do not know the number of blocked channels, we aim at being competitive to the optimal algorithm which has knowledge of this value. Hence, the expected number of attempts to find another device shall be very small for only a few jammers and the performance of the algorithms shall only degrade gracefully for larger numbers of jammers.

The remainder of this thesis is organized as follows: Chapter 2 lists related work before we precisely state in Chapter 3 the model we use for our considerations. In Chapter 4, which is also the core chapter of the thesis, we derive efficient algorithms for the aforementioned discovery problem for two devices. Chapter 5 then discusses strategies to follow in the multiplayer case. To validate the effectiveness of our algorithms, also in comparison with the inquiry algorithm from Bluetooth, we run simulations which we describe in Chapter 7. The thesis is concluded by final remarks in Chapter 8.

2

Related Work

A lot of research concerning jamming in wireless networks (e.g. [1, 3, 9, 10, 18, 19, 20, 21]) has already been conducted. Both directions, from the point of view of efficient jamming (e.g. [3, 9, 10]) as well as from the point of view of efficient countermeasures (e.g. [1, 3, 10, 18]), have been widely studied.

Most of this work considers communication in the presence of jammers. However, the fundamental task of discovering communication partners in the presence of jammers has been broadly neglected. In this thesis, we only focus on the discovery problem.

Closely related to our work, Strasser et al. [16] propose a frequency hopping scheme to establish some shared secret key over wireless frequency channels while a jammer is present. As this hopping scheme is uncoordinated, the devices do not have to have an a priori knowledge of each others. Also very closely related to this thesis, in [5], Gilbert et al. examine the gossip problem in a setting which is equivalent to the setting used here.

Assumptions On Jammers

The most common assumptions concerning jammers in theoretical studies are that messages are corrupted at random [13] or that the jammer has bounded energy and thus is able to jam a given number of messages only [6, 7]. In our thesis, neither the first nor the second restriction is assumed. However, as we require the jammer not to simultaneously jam all channels, we are closer to the second approach than to the first one.

Detection Of Jammers

In [20], Xu et al. state that it is not easy to detect sophisticated jammers. However, they propose two protocols which are more effective than simple methods, like e.g. signal strength measurements, yet not satisfactory. In this thesis we do not try to detect jammers at all. Instead, we aim at being efficient no matter how many channels are jammed.

Strategies Against Jammers

Widely used strategies against jammers are physical layer technologies such as e.g. spread spectrum [11, 12, 15]. This technique is indeed useful, however, a narrow spreading is still applied in many popular protocols today. Hence, defenses on the MAC layer have been proposed (e.g. [2]).

3

Model

3.1 Setting

We are given m consecutively numbered frequency channels and n devices, which are able to send and receive information over these channels. Note that we mainly consider the setting with $n = 2$ devices and discuss the multiplayer case, where $n > 2$, only in Chapter 5. We assume the channels to be broad enough so that sending on a channel does not influence transmission over nearby channels. Furthermore, the channels shall be lossless and deterministic.

We assume the devices to have unique IDs and synchronized clocks, i.e. we do not consider clock drifts. Furthermore, we assume every device to be within the transmission range of all other devices, i.e. we treat the single-hop case only.

As we are interested in how long it takes for the devices to get to know each other, we will count the number of time slots from the entrance of the second (last, respectively) device to the system until all devices have either direct or indirect knowledge about all other devices. Note that in the beginning the devices have no information about which (with respect to their IDs) or even how many other devices there are. For every time slot, each device decides whether it sends or receives in this slot and on which channel it performs this action.

Note that we do not restrict the slots in their length. That is, the devices shall have the possibility to transmit their ID and, for example, some hopping sequence for further communication in only one slot.

As mentioned above, we consider the devices to have succeeded as soon as they

manage to meet on a channel during a time slot. In the two player case this means that both devices decide for the same channel in the same time slot, but one of them sends its information whereas the other one tries to receive it. In the multiplayer case such a meeting has to happen at least $n - 1$ times.

Note that in reality connection establishment most often needs at least two time slots since usually some handshaking policy is applied. However, we do not take this into account here but instead, as we do not restrict the duration of the time slots, we assume that both, an inquiry as well as a reply, may take place in the same slot.

3.2 Jammer

As already mentioned in the introduction in Chapter 1, every device that is using the restricted frequency spectrum available for wireless activity should take into account that other devices could use the same frequency at the same time which might result in collisions and possibly cause the device to fail in its mission.

However, such collisions can not only occur accidentally but also intentionally by malicious devices, called *jammers*. For our considerations we concentrate on the second one. That is, we are given a jammer which has the possibility to simultaneously jam $0 \leq t < m$ channels. Alternatively, one can also think of t jammers, each of which can jam exactly one channel. However, with the centrally organized approach we are given the freedom of not taking into account the communication overhead for the individual jammers so that they can avoid jamming the same channel more than once.

Although we will discuss further jammer models in Chapter 6, we will focus on the following model until then:

Definition 3.2.1 (Worst Case Jammer). *Let us assume that the jammer knows the algorithm of the devices, i.e. if this algorithm is deterministic then it is aware of the channels that will be chosen and otherwise it can calculate the devices' probability distribution over the channels. The jammer then decides to jam exactly those t channels which are favored by any device. Once decided, the jammer will always remain on these channels.*

Observation 3.2.2. *The jammer defined above is a worst case jammer in the sense that exchanging a jammed channel with one that came out not to be jammed would increase the probability of success for the devices.*

If a channel is jammed, then it is totally useless for any transmission of the devices. Moreover, we assume that it is impossible for a device, which is listening on channel c_i but not receiving a message, to distinguish between the following cases:

1. c_i is not jammed but there is no device sending on c_i

2. c_i is jammed but there is no device sending on c_i anyway
3. c_i is jammed and there is indeed another device sending on c_i

3.3 Objective

Having defined the setting as well as the jammer we now want to formally state our objective. For our algorithms we will assume t to be unknown to the devices. It is obvious that an optimal algorithm A_{opt} which has knowledge of t is at least as good as our unaware algorithms. Hence, for every t , we aim at being competitive compared to A_{opt} .

Let $s_A(t)$ be the number of time slots needed by two devices, which apply algorithm A , to successfully meet if those t channels with the highest probability to be chosen by the devices are jammed. The penalty Q_A of A is then measured by

$$Q_A = \max_t \frac{\mathbf{E}[s_A(t)]}{\mathbf{E}[s_{A_{opt}}(t)]}$$

which shall be minimized.

4

Algorithms

We now investigate different algorithms for efficient connection establishment. We start with some considerations about deterministic algorithms to then focus on randomized ones.

4.1 Deterministic Algorithms

Fully deterministic algorithms bear two substantial problems. First, a jammer which we assume to have knowledge of the algorithm can of course block any attempt of a device to transmit its information, given that it can change the channels it jams. Even in the multiplayer case there are $n \geq 2t + 2$ devices required in order to guarantee only one successful meeting. Furthermore, the jammer can totally prevent t devices from getting to know any other device. However, this only holds if the jammer is synchronized with the execution of the algorithm.

The second problem is that two devices which happen to start the execution of their algorithms at the same time will never succeed in meeting. Note that this might be unrealistic when we look at a handheld computer which tries to connect to a printer, but it may very well occur concerning for example a sensor network which has been spread over some terrain and is activated by a global signal.

As we assume the devices to have an ID it is clear that they could use this to generate some pseudo random hopping sequence. However, as the jammer we consider, i.e. the worst case jammer as described by Definition 3.2.1, is static¹ and as an algo-

¹It is static in the sense that it chooses the t channels only once to then remain on them.

rithm that uses some pseudo random hopping sequence ideally is behaving like a real random algorithm, we will not go into details here and instead refer to the following section.

4.2 Randomized Algorithms

For each round, the devices have to decide for a channel as well as for an action to perform on this channel. In the two player case, to which we restrict our considerations in this chapter, the following lemma states how the latter decision should be made.

Lemma 4.2.1. *In the two player case, any device shall choose to send (to listen, respectively) with probability 0.5 in every round.*

Proof. Assume that two devices d_1 and d_2 which choose the same channel for a time slot decide to send with probability p_s and, respectively, to listen with probability $p_l = 1 - p_s$. Their meeting is successful if and only if d_1 sends while d_2 listens or vice versa. Hence, $\Pr[\text{success}] = 2p_s p_l = 2(p_s - p_s^2)$ which is maximized by setting $p_s = 0.5$. \square

For the rest of this chapter we therefore assume that the devices indeed choose to send and, respectively, receive with probability 0.5 each.

To start with the examination of randomized algorithms which are used to decide for a channel to perform the respective action, we state the following observation:

Observation 4.2.2. *Every algorithm that uses some random generator to decide for one of the available channels per time slot can be expressed as a probability distribution over these channels.*

We first assume that t , the number of jammed channels, is known to the devices and look for the best possible strategy they then have. After that, we loosen this assumption and examine the substantially more challenging case.

Lemma 4.2.3. *Let m denote the number of channels and assume t , the number of those channels which are jammed, to be known. If $t \leq m/2$ then the optimal algorithm decides for one of the first² $2t$ channels uniformly at random. Otherwise, any channel is chosen with probability $1/m$.*

Proof. Let p_{c_k} be the probability for channel c_k to be chosen by any device and let c_1, c_2, \dots, c_d be the channels which are taken into consideration by the devices, i.e. $p_{c_{d+1}} = p_{c_{d+2}} = \dots = p_{c_m} = 0$. Assume the devices' probability distribution over these channels is not uniform and $p_{c_1} \geq p_{c_2} \geq \dots \geq p_{c_d}$. Note that d has to be at least equal to $t + 1$ as otherwise the devices will never succeed in meeting.

²Note that we have assumed the channels to be consecutively numbered in Section 3.1.

As the sequence p_{c_i} is decreasing it is clear that the jammer will block the channels c_i, \dots, c_t and as it is not uniform it holds that

$$\underbrace{\sum_{i=t+1}^d p_{c_i}}_{\text{unjammed channels}} < \frac{d-t}{d} < \underbrace{\sum_{i=1}^t p_{c_i}}_{\text{jammed channels}}$$

where $\frac{d-t}{d}$ is equal to the sum of the channel probabilities of the channels c_{t+1}, \dots, c_d if the probability distribution over all d channels was uniform. That is, we can cut off some probability from those channels with probability greater than $1/d$ and distribute it over the other channels to increase the total probability of success (cf Figure 4.1 for an example).

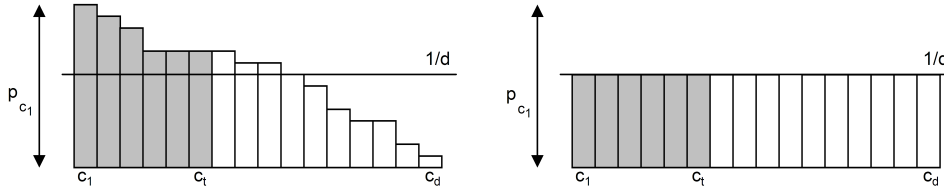


Figure 4.1: Redistribution of the channel probabilities

Note that it is of course possible to perform this redistribution in such a manner that the order $p_{c_1} \geq p_{c_2} \geq \dots \geq p_{c_d}$ still holds. It is clear that, assuming we indeed conserve this order, it holds that we have reached the uniform probability distribution over all channels as soon as $\sum_{i=t+1}^d p_{c_i} = \frac{d-t}{d}$. Moreover, we cannot further redistribute the probabilities without decreasing the overall probability of success as the jammer always blocks the t most probable channels.

It remains to show that $d = 2t$ maximizes the probability of success. As the first t channels are jammed and the probability for every considered channel to be chosen is $p_{c_i} = 1/d$, the probability of success is

$$\Pr[\text{success}] = \frac{1}{2} \sum_{i=t+1}^d \frac{1}{d^2} = \frac{d-t}{2d^2}.$$

In order to maximize this term we take its derivative and ask it to be equal to 0.

$$\begin{aligned} \frac{1}{2d^2} - \frac{d-t}{d^3} &\stackrel{!}{=} 0 \\ d &\stackrel{!}{=} 2(d-t) \\ d &\stackrel{!}{=} 2t \end{aligned}$$

Note that if $2t > m$ the devices can of course not consider $2t$ channels and instead have to confine themselves to m channels. \square

Corollary 4.2.4. *The expected running time of the optimal algorithm A_{opt} as described by Lemma 4.2.3 is*

$$\begin{aligned} 8t & \text{ for } t \leq \frac{m}{2} \\ \frac{2m^2}{m-t} & \text{ for } \frac{m}{2} < t \end{aligned}$$

For the rest of this chapter we assume t to be unknown to the devices. It seems reasonable though that the devices estimate the number of jammed channels in order to be able to choose their strategy accordingly. As we aim at being competitive with the optimal algorithm for every possible $0 \leq t \leq m - 1$ it is clear that is not sufficient to estimate t only once. Instead, the devices will make such an estimation for every round.

4.2.1 3 Classes

Applying the algorithm A_3 , the two devices d_1 and d_2 estimate t to either be $\hat{t} = 1$, $\hat{t} = \frac{\sqrt{m}}{2}$ or $\hat{t} = \frac{m}{2}$ with equal probability each. That is, in each round they choose a channel to send/receive out of one of the classes $\{c_1, c_2\}$, $\{c_1, \dots, c_{\sqrt{m}}\}$ or $\{c_1, \dots, c_m\}$.

It is clear that if $t > 1$ then the devices will fail in every round in which at least one of them guesses $\hat{t} = 1$. However, if there is indeed only one channel jammed, then they will successfully meet very quickly.

Theorem 4.2.5. *Let m denote the number of channels and let $t < m$ be the number of those channels which are jammed. The expected running time of A_3 is in*

$$\begin{aligned} \mathcal{O}(1) & \text{ for } t < 2 \\ \mathcal{O}(\sqrt{m}) & \text{ for } 2 \leq t < \sqrt{m} \\ \mathcal{O}\left(\frac{m^2}{m-t}\right) & \text{ for } \sqrt{m} \leq t \end{aligned}$$

Proof. The exact probability for a successful meeting, given the number of channels the devices take into account, is given in Table 4.1. Note that we omit the factor 0.5, which results from the fact that one of the devices has to send while the other one has to listen, for readability reasons.

| | | d_1 | | |
|-------|------------|--|--|--|
| | | 2 | \sqrt{m} | m |
| d_2 | 2 | $\frac{1}{2}$ if $t < 2$ 0 if $t \geq 2$ | $\frac{1}{\sqrt{m}}$ if $t < 2$ 0 if $t \geq 2$ | $\frac{1}{m}$ if $t < 2$ 0 if $t \geq 2$ |
| | \sqrt{m} | $\frac{1}{\sqrt{m}}$ if $t < 2$ 0 if $t \geq 2$ | $\frac{\sqrt{m}-t}{m}$ if $t < \sqrt{m}$ 0 if $t \geq \sqrt{m}$ | $\frac{\sqrt{m}-t}{m\sqrt{m}}$ if $t < \sqrt{m}$ 0 if $t \geq \sqrt{m}$ |
| | m | $\frac{1}{m}$ if $t < 2$ 0 if $t \geq 2$ | $\frac{\sqrt{m}-t}{m\sqrt{m}}$ if $t < \sqrt{m}$ 0 if $t \geq \sqrt{m}$ | $\frac{m-t}{m^2}$ |

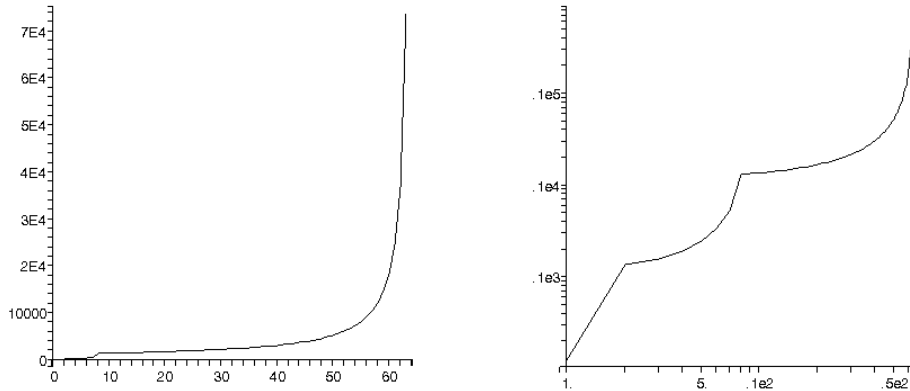
Table 4.1: Success probabilities of A_3 ; factor 0.5 omitted

Summarized, we have the following success probabilities:

$$\begin{aligned} \frac{1}{18} \left(\frac{\sqrt{m}-t}{m} + 2\frac{\sqrt{m}-t}{m\sqrt{m}} + \frac{m-t}{m^2} + 0.5 + \frac{2}{\sqrt{m}} + \frac{2}{m} \right) & \text{ for } t < 2 \\ \frac{1}{18} \left(\frac{\sqrt{m}-t}{m} + 2\frac{\sqrt{m}-t}{m\sqrt{m}} + \frac{m-t}{m^2} \right) & \text{ for } 2 \leq t < \sqrt{m} \\ \frac{1}{18} \left(\frac{m-t}{m^2} \right) & \text{ for } \sqrt{m} \leq t \end{aligned}$$

This immediately gives the expected running time as stated by Theorem 4.2.5. \square

A graphical representation of the expected running time of A_3 as a function of t is provided in Figure 4.2. Note that the second plot has logarithmic scales.

Figure 4.2: Plots of the expected running time of A_3 for $m = 64$

The penalty of A_3 is $Q_{A_3} = \frac{9m}{4(\sqrt{m}-1)}$ which is reached for $t = \sqrt{m}$. For $m = 64$, this results in $Q_{A_3} \approx 21$.

4.2.2 $\log_2(m)$ Classes

We now refine the algorithm from Section 4.2.1. That is, instead of having only 3 guess classes we now investigate the case with $\log_2(m)$ classes. For convenience reasons we assume m to be equal to 2^j for some integer value j .

Each device guesses $\hat{t} = 2^i$ for some integer value i . As t is bounded by $1 \leq t < m$, i is bounded by $0 \leq i < \log_2(m)$ which indeed yields $\log_2(m)$ guess classes. Note that choosing i to be equal to $\log_2(m) - 1$, i.e. guessing $\hat{t} = 2^{\log_2(m)-1} = m/2$, results in considering all m channels.

Theorem 4.2.6. *Let the number of jammed channels be $m = 2^j$ for some integer value j , let $t < m$ denote the number of those channels which are jammed, and let $x = \lfloor \log_2(t) \rfloor$. The expected running time of the $\log_2(m)$ classes algorithm $A_{\log_2(m)}$ is*

$$\frac{2}{\ln(2)^2} \cdot \frac{m^2 \ln(m)^2}{2m \log_2(m) + 3m + t - 2x - 2^{1-x}tm - 3 \cdot 2^{-x} + t4^{-x}m^2}$$

Proof. Let i_1 be the chosen i of the device d_1 in some round and, respectively, i_2 be the according value of d_2 (cf Figure 4.3). Assume $i_1 \geq i_2$ and $2^{i_2+1} \geq t$. The probability that the two devices successfully meet in this round is

$$\Pr[\text{meeting}] = \frac{2^{i_2+1} - t}{2^{i_1+1}2^{i_2+1}} = \frac{1}{2^{i_1+1}} - \frac{t}{2^{i_1+i_2+2}}.$$

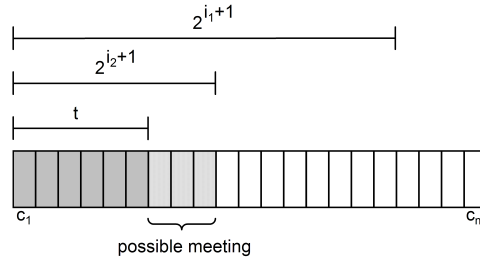


Figure 4.3: Considered channels of the devices d_1 and d_2

Note that if at least one of the devices chooses $i \leq \log_2(t) - 1$, i.e. it guesses $\hat{t} \leq t/2$, then the devices have no chance to succeed in that round.

Let $p(i_1, i_2, t) = \frac{1}{2^{i_1+1}} - \frac{t}{2^{i_1+i_2+2}}$. The overall success probability in some round is given by

$$\frac{0.5}{\log_2(m)^2} \left(\sum_{i_1=\lfloor \log_2(t) \rfloor}^{\log_2(m)-1} \sum_{i_2=\lfloor \log_2(t) \rfloor}^{i_1-1} p(i_1, i_2, t) + \sum_{i_1=\lfloor \log_2(t) \rfloor}^{\log_2(m)-1} \sum_{i_2=i_1}^{\log_2(m)-1} p(i_2, i_1, t) \right) =$$

$$\left(\ln(2)^2 \left(2m \log_2(m) + 3m + t - 2 \lfloor \log_2(t) \rfloor - 2^{1-\lfloor \log_2(t) \rfloor} tm \right) \right)$$

$$- 3 \cdot 2^{-\lfloor \log_2(t) \rfloor} + t 4^{-\lfloor \log_2(t) \rfloor} m^2)) / (2m^2 \ln(m)^2)$$

The expected running time as stated by Theorem 4.2.6 follows by inverting the value for the overall success probability. \square

Again, a graphical representation of the expected running time of $A_{\log_2(m)}$ for $m = 64$ is provided in Figure 4.4. Note that the second plot has logarithmic scales.

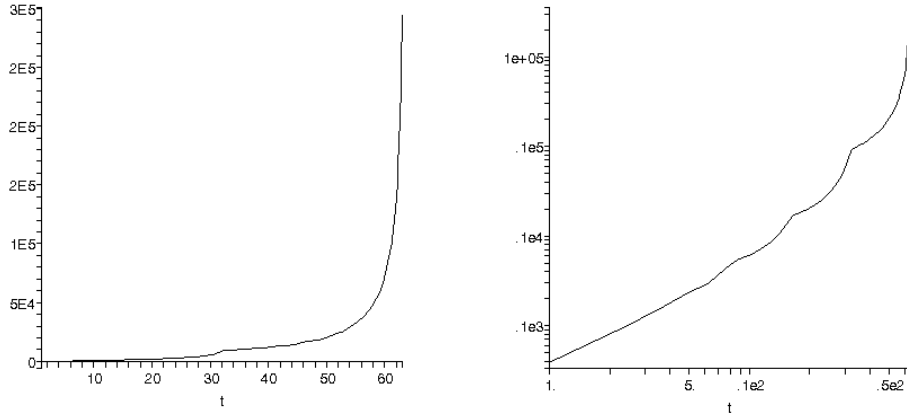


Figure 4.4: Plots of the expected running time of $A_{\log_2(m)}$

$Q_{A_{\log_2(m)}}$, the penalty of $A_{\log_2(m)}$, is equal to $\log_2(m)^2$ which is reached for $t \in [m/2, \dots, m - 1]$. In the example of $m = 64$ this results in $Q_{A_{\log_2(m)}} = 36$.

4.2.3 $m/2$ Classes

As a natural next step we now consider $m/2$ classes, i.e. the devices guess t to be in $[1, \dots, m/2]$. Note that guessing $\hat{t} = m/2$ results in considering all m channels.

Similar to $A_{\log_2(m)}$, it holds for $A_{m/2}$ that if device d_1 guesses $t = x$ while d_2 guesses $t = y$ and $x \geq y$, then the probability that the two devices successfully meet in this round is

$$\Pr[\text{meeting}] = \frac{1}{2x} \frac{1}{2y} (2y - t) = \frac{1}{4x} - \frac{t}{8xy}.$$

It is clear that both devices have to guess $\hat{t} > t/2$ in order to have a chance to successfully meet in that round.

Let $p(x, y, t) = \frac{1}{4x} - \frac{t}{8xy}$ and assume m and t to be even. The overall success probability in some round is given by

$$\frac{0.5}{(m/2)^2} \left(\sum_{x=t/2+1}^{m/2} \sum_{y=t/2+1}^{x-1} p(x, y, n) + \sum_{x=t/2+1}^{m/2} \sum_{y=x}^{m/2} p(y, x, n) \right).$$

For $\Psi(x) = \ln(\frac{x}{2}) - \frac{1}{x}$, this can be approximated by

$$2m^3 / \left(2t(\Psi(t+2) - \Psi(m+2) - 1) + 2m(\Psi(t+4) - \Psi(m+4) + 1) \right. \\ \left. + mt \left(\Psi(t+2) - \Psi(m+2)(1 - \Psi(t+2)) - \Psi(t+2)^2 - \Psi(m)(1 + \Psi(m+2) - \Psi(t+2)) \right. \right. \\ \left. \left. + \Psi(t+4) - 2 \right) + m^2(\Psi(m) - \Psi(m+4) + 2) \right).$$

An approximation of the expected running time of $A_{m/2}$ is given by the reciprocal of this term. Its graphical representation for $m = 64$ is shown in Figure 4.5.

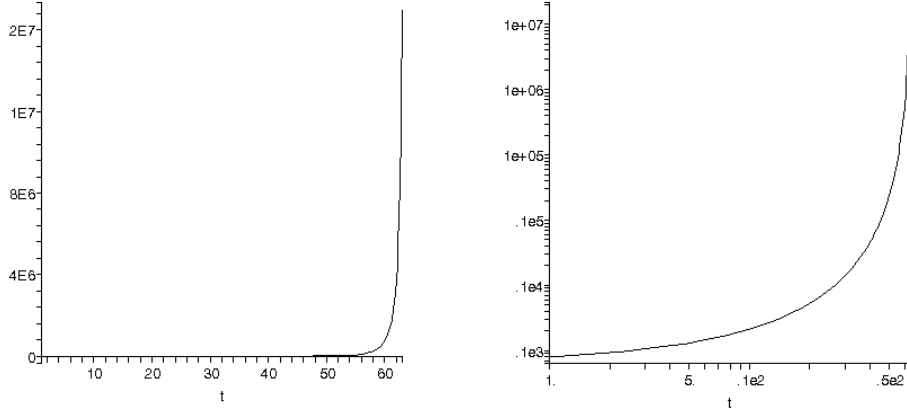


Figure 4.5: Plots of the expected running time of $A_{m/2}$

The penalty of $A_{m/2}$ is $Q_{A_{m/2}} = m^2/4$ which is reached for $t = m - 1$. For $m = 64$, this results in $Q_{A_{m/2}} = 1024$.

4.2.4 k Classes

After having discussed three algorithms with a fixed number of classes, we now investigate a more generic algorithm A_k . That is, the devices shall choose one out of k classes per round. More precisely, each device chooses some $1 \leq i \leq k$ and will then consider the channels $c_1, \dots, c_{m^{i/k}}$ in that round.

Theorem 4.2.7. *Let m denote the number of channels and let $t < m$ be the number of those channels which are jammed. Let $x = \left\lfloor \frac{k \cdot \ln(t)}{\ln(m)} \right\rfloor$ and $y = m^{-\frac{x}{k}}$ for some integer value k . The expected running time of A_k is*

$$\frac{2k^2 m (m^{1/k} - 1)^2}{m^{\frac{1}{k}} (2x - 2k - 1) - \frac{t}{m} + 2k - 2x - 1 + y \left(m^{\frac{k+1}{k}} + 2t + m - tmy \right)}$$

Proof. Let the device d_1 choose i_1 and let d_2 choose i_2 in the same round. Let $i_1 \geq i_2 > \frac{k \cdot \ln(t)}{\ln(m)}$. The second inequality comes from $m^{i_2/k} > t$ which we require as otherwise the devices will fail in this round for sure. The probability that the two devices successfully meet in this round is

$$\Pr[\text{meeting}] = \frac{m^{i_2/k} - t}{m^{i_1/k} m^{i_2/k}} = \frac{1}{m^{i_1/k}} - \frac{t}{m^{(i_1+i_2)/k}}.$$

Let $p(i_1, i_2, t) = \frac{1}{m^{i_1/k}} - \frac{t}{m^{(i_1+i_2)/k}}$. The overall success probability in some round is given by

$$\begin{aligned} & \frac{0.5}{k^2} \left(\sum_{i_1 = \lfloor \frac{k \cdot \ln(t)}{\ln(m)} \rfloor + 1}^k \sum_{i_2 = \lfloor \frac{k \cdot \ln(t)}{\ln(m)} \rfloor + 1}^{i_1 - 1} p(i_1, i_2, t) + \sum_{i_1 = \lfloor \frac{k \cdot \ln(t)}{\ln(m)} \rfloor + 1}^k \sum_{i_2 = i_1}^k p(i_2, i_1, t) \right) = \\ & \left(m^{\frac{1}{k}} \left(2 \left\lfloor \frac{k \cdot \ln(t)}{\ln(m)} \right\rfloor - 2k - 1 \right) - \frac{t}{m} + 2k - 2 \left\lfloor \frac{k \cdot \ln(t)}{\ln(m)} \right\rfloor - 1 \right. \\ & \left. + m^{-\frac{1}{k} \lfloor \frac{k \cdot \ln(t)}{\ln(m)} \rfloor} \left(m^{\frac{k+1}{k}} + 2t + m - tm^{1-\frac{1}{k} \lfloor \frac{k \cdot \ln(t)}{\ln(m)} \rfloor} \right) \right) / (2k^2 m (m^{1/k} - 1)^2). \end{aligned}$$

The estimated running time as stated by Theorem 4.2.7 thus follows. \square

A graphical representation for the estimated running time of A_k for the two examples $k = \log_2(m)$ and $k = \sqrt{m}$ for $m = 64$ is provided in Figure 4.6.

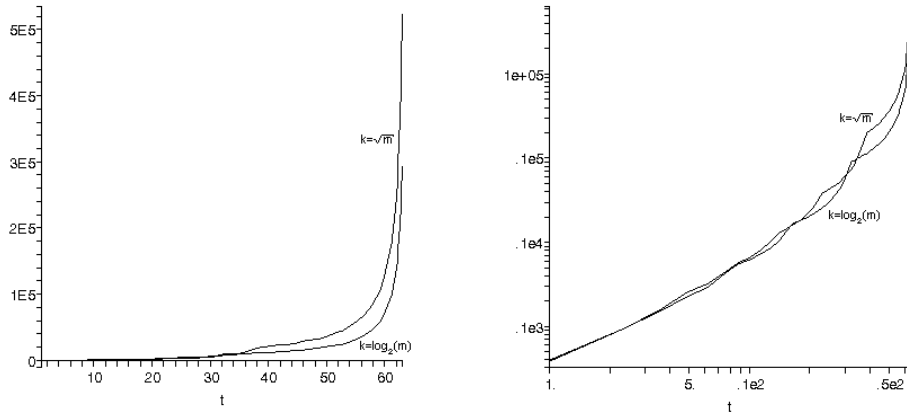


Figure 4.6: Plots of the expected running time of A_k ; $m=64$

To get a lower bound on the penalty Q_{A_k} of A_k let $t = m - 1$. To successfully meet on the only channel which is not jammed, both devices have to choose $i = k$ and decide for the last channel in the according channel set. Furthermore, one of them has to send while the other one is listening. Hence, we get $Q_{A_k} = \Omega\left(\frac{2k^2 m^2}{2m^2}\right) = \Omega(k^2)$.

4.3 Comparison

As A_3 guesses t to be either 1, $\sqrt{m}/2$ or $m/2$, it is clear that it is very fast if t is indeed one of these values. More precisely, as A_3 has only a constant number of guess classes, its expected running time matches the one from the optimal algorithm, i.e. $\mathbf{O}(1)$, $\mathbf{O}(\sqrt{m})$ and $\mathbf{O}(m)$ respectively. In contrast to this, the expected running time of $A_{\log_2(m)}$ has an additional factor of $\mathbf{O}(\log(m)^2)$ in these three cases. This comes from the fact that while A_3 makes an optimal guess very often, any device applying $A_{\log_2(m)}$ needs $\mathbf{O}(\log(m))$ more attempts. However, as soon as t does not lie in $\{1, \sqrt{m}/2, m/2\}$, A_3 gets worse compared to $A_{\log_2(m)}$ of course.

Even if in our example of $m = 64$ we have $Q_{A_3} \approx 21 < 36 = Q_{A_{\log_2(m)}}$, the penalty of $A_{\log_2(m)}$ is asymptotically better than the penalty of A_3 . While the ratio of the expected running time of A_3 over the expected running time of A_{opt} is maximized for $t = \sqrt{m}$ and yields $Q_{A_3} = \frac{18m^2}{(m-\sqrt{m}) \cdot 8\sqrt{m}} \in \mathbf{O}(\sqrt{m})$, the maximum of the according ratio for $A_{\log_2(m)}$ is $Q_{A_{\log_2(m)}} = \frac{2m^2 \log_2(m)^2}{2m^2} \in \mathbf{O}(\log(m)^2)$, which is reached for $t = m - 1$.

Far off is algorithm $A_{m/2}$ with $Q_{A_{m/2}} \in \mathbf{O}(m^2)$, as both devices running $A_{m/2}$ need to choose class $C_{m/2}$ (which happens with probability $4/m^2$) in order to have a chance to succeed if $t = m - 1$.

Note that the algorithms $A_{\log_2(m)}$ and $A_{k=\log_2(m)}$ are equivalent and thus have the same expected running time and penalty.

Table 4.2 gives an overview of the penalties of the algorithms A_3 , $A_{\log_2(m)}$, $A_{m/2}$ and A_k .

| | penalty | reached for |
|-----------------|----------------------------|---------------------------|
| A_3 | $\frac{9m}{4(\sqrt{m}-1)}$ | $t = \sqrt{m}$ |
| $A_{\log_2(m)}$ | $\log_2(m)^2$ | $t \in [m/2, \dots, m-1]$ |
| $A_{m/2}$ | $m^2/4$ | $t = m-1$ |
| A_k | $\Omega(k^2)$ | $t = m-1$ |

Table 4.2: Overview of the penalties

The expected running times of the algorithms A_{opt} , A_3 , $A_{\log_2(m)}$ and $A_{m/2}$ are graphically compared in Figure 4.7 and Figure 4.8 for $m = 64$. While Figure 4.7 compares the running times for $1 \leq t \leq m/2$, Figure 4.8 shows the upper part, i.e. $m/2 \leq t \leq m - 1$. Note that Figure 4.7(b) and Figure 4.8(b) have logarithmic scales.

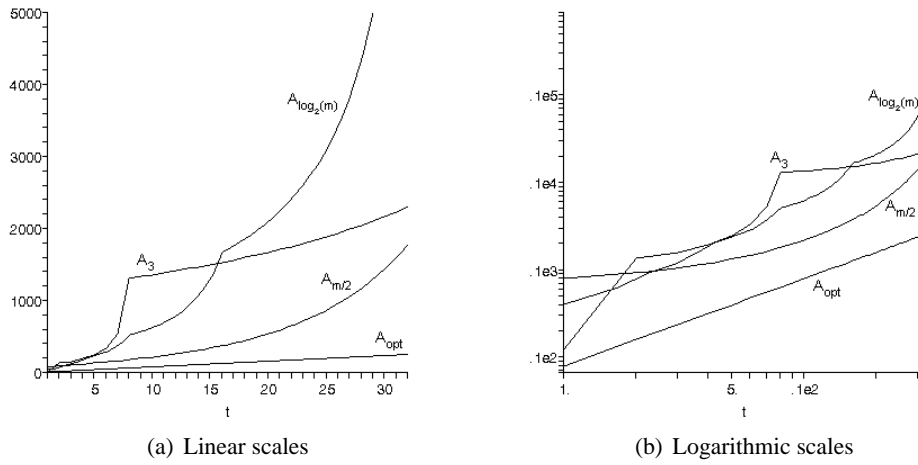


Figure 4.7: Comparison of the expected running times; $t = [1, \dots, m/2]$

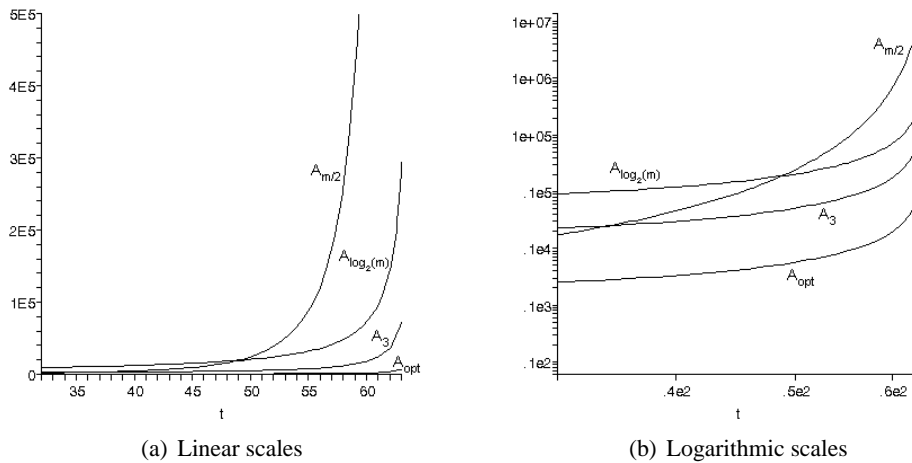


Figure 4.8: Comparison of the expected running times; $t = [m/2, \dots, m - 1]$

4.4 Classes vs. Probability Distributions

All algorithms we investigated before follow the *classes approach*, i.e. they first choose a class of channels and then they decide for one specific channel within this class to operate on. Hence, the probability distribution over the channels is a result of two decisions which are made uniformly at random. Another possibility for an algorithm is to follow the *direct probability approach*, i.e. to define a probability distribution over the channels to then decide for one of them according to this distribution. However, with a modification on how we chose the classes before, these two approaches are similar.

Lemma 4.4.1. *If the class of channels does not have to be chosen uniformly at random, then the classes approach and the direct probability approach are equivalent.*

Proof. It is obvious that every probability distribution which results from the classes approach can also be defined directly. It remains to show that this indeed holds for the other direction as well.

Let P be some probability distribution over the channels and let p_{c_i} be the according probability of channel c_i . Let the channels be ordered in a way such that $p_{c_1} \geq \dots \geq p_{c_m}$. We construct the classes and their weights, with which they will be chosen, as follows:

Class C_k for $1 \leq k \leq m$ contains all the channels in $[c_1, \dots, c_k]$ and has the weight $w_{C_k} = k \cdot (p_{c_k} - p_{c_{k+1}})$ where $p_{c_{m+1}} = 0$. Note that $\sum_{k=1}^m w_{C_k} = 1$.

To see that these classes together with their weights indeed yield P , we now analyze the probability that is assigned to a channel c_j in both approaches. As c_j occurs in all classes C_k for $j \leq k \leq m$, we only have to consider these classes. The probability that one of these classes is chosen is equal to its weight, and c_j will be chosen thereafter with probability $1/k$ for k being the size of elected class. Hence, in this classes approach we have

$$\Pr[c_j \text{ is chosen}] = \sum_{i=j}^m w_{C_i} \frac{1}{i} = \sum_{i=j}^m k \cdot (p_{c_k} - p_{c_{k+1}}) \frac{1}{k} = p_{c_j} - p_{c_{m+1}} = p_{c_j}$$

which concludes the proof. \square

Having these considerations in mind it seems natural to extend our algorithms to also support other probability distributions over the channel classes. This is especially useful if t is indeed unknown to the devices but they have knowledge of the distribution of t .

To investigate some examples of such distributions, we consider the algorithm $A_{\log_2(m)}$ from Section 4.2.2.

We restrict the successful meetings in the sense that we require both devices to also decide for the same class, i.e. the devices only succeed if in one round both of them consider the same set of channels and decide for the same channel therein. This gives an upper bound on the expected running time of the according algorithm.

4.4.1 Uniform Distribution

The distribution we looked at so far is the uniform distribution. That is, the probability for a class C_i to be chosen is equal to $1/\log_2(m)$. In Figure 4.9(a) the probability for the channel classes is presented, whereas Figure 4.9(b) shows, as a function of t , the expected running time divided by the expected running time of A_{opt} . Hence, the

highest point of the curve constitutes the penalty of the algorithm. Note that Figure 4.9(b) has a semi-logarithmic scaling.

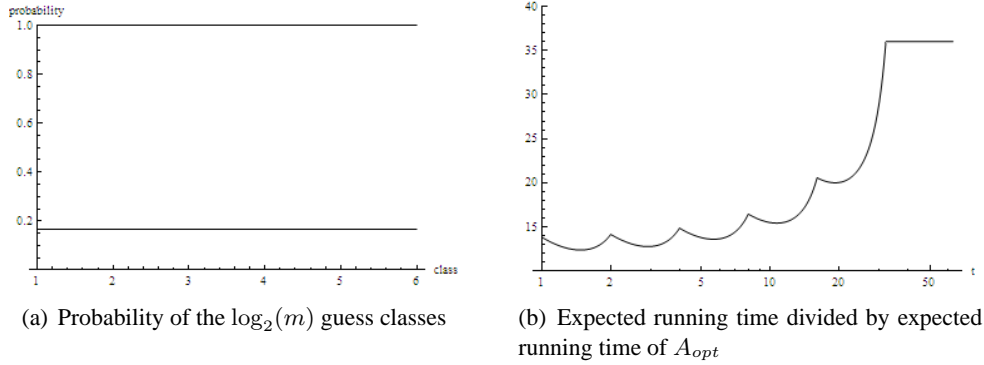


Figure 4.9: Uniform distribution over the channel classes; $m = 64$

The penalty of $A_{\log_2(m)}$ with a uniform distribution over the classes is $Q_{A_{\log_2(m)}^{uni}} = 36$, t being equal to $m - 1$.

4.4.2 Binomial Distribution

When a binomial distribution is applied, a class C_i has probability $\binom{\log_2(m)}{i} p^i (1 - p)^{\log_2(m) - i}$ for $0 < p < 1$ to be chosen. Figure 4.10 shows the probability of the classes as well as the expected running time compared to $A_{\log_2(m)}^{uni}$ for $p = 0.61$.

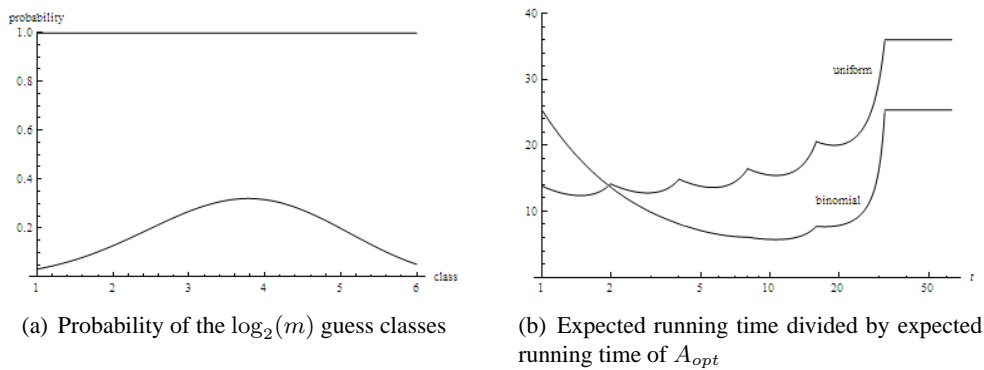


Figure 4.10: Binomial distribution over the channel classes; $m = 64$

The probability concentrates on lower classes for $p \rightarrow 0$ and, respectively, on higher classes for $p \rightarrow 1$. For small p the expected running time exceeds the one from the uniform distribution by far for large t .

$Q_{A_{\log_2(m)}^{bin}}$, the penalty of $A_{\log_2(m)}$ with a binomial distribution over the classes, is minimized for $p \approx 0.61$ to reach $Q_{A_{\log_2(m)}^{bin}} \approx 25.3 < Q_{A_{\log_2(m)}^{uni}}$.

4.4.3 Poisson Distribution

The probability for a class C_i is defined as $\frac{\lambda^i}{i!}e^{-\lambda}$ for $0 < \lambda$ when a poisson distribution is applied. Note that the probabilities have to be normalized by a division by $\sum_{i=1}^{\log(m)} \frac{\lambda^i}{i!}e^{-\lambda}$. The according plots are given by Figure 4.11 for $\lambda = 4.14$

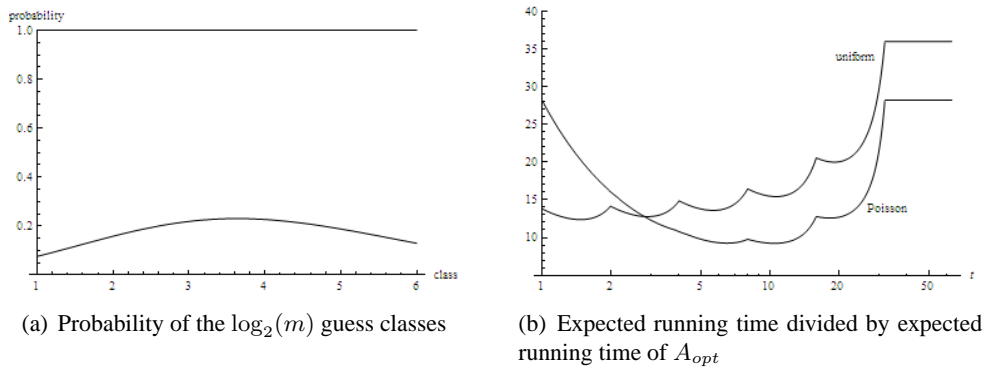


Figure 4.11: Poisson distribution over the channel classes; $m = 64$

For $\lambda \rightarrow 0$, the probability concentrates on lower classes whereas higher classes are emphasized by large λ . The expected running time behaves accordingly.

The penalty of the algorithm with a poisson distribution over the classes is minimized for $\lambda \approx 4.14$ and is $Q_{A_{\log_2(m)}^{poi}} \approx 28.2 < Q_{A_{\log_2(m)}^{uni}}$.

4.4.4 Geometric Distribution

In a geometric distribution the probability of a class is given by $(1-p)^{i-1}p$ for $0 < p < 1$. Again, the probabilities have to be normalized by the factor $1/\sum_{i=1}^{\log(m)} (1-p)^{i-1}p = 1/(1-(1-p)^{\log(m)})$. The plots for the geometric distribution with $p = 0.2$ are presented in Figure 4.12.

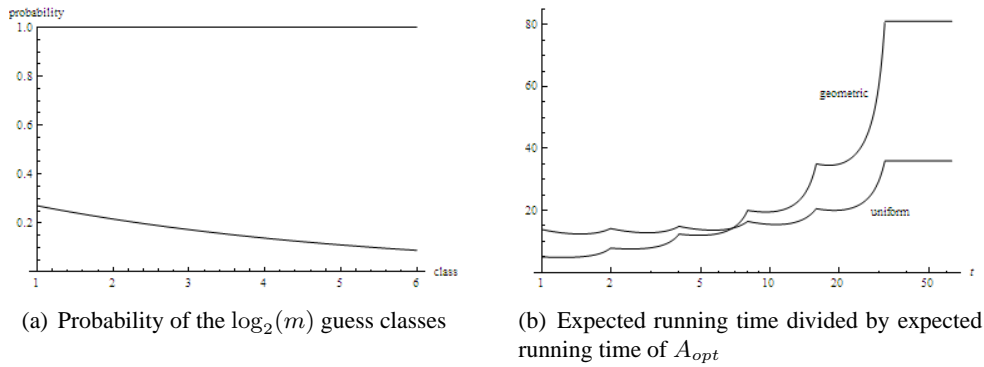


Figure 4.12: Geometric distribution over the channel classes; $m = 64$

The geometric distribution adapts to the uniform distribution for $p \rightarrow 0$ and concentrates on the low classes for $p \rightarrow 1$. Hence, it holds that $Q_{A_{\log_2(m)}^{geo}} = Q_{A_{\log_2(m)}^{uni}}$.

4.4.5 Exponential Distribution

The last distribution we consider is the exponential distribution, i.e. the probability for class C_i is equal to x^i for some $0 < x$, with normalization factor $1 / \sum_{i=1}^{\log(m)} x^i = \frac{x-1}{x \cdot \log(m+1) - x}$. The graphical representation for the exponential distribution with $x = 1.14$ as well as the comparison of its expected running time with the one resulting from the uniform distribution over the classes are given in Figure 4.13.

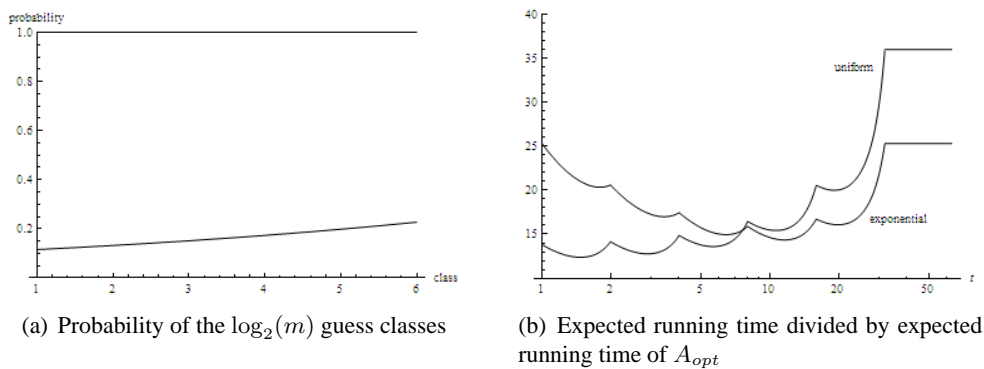


Figure 4.13: Exponential distribution over the channel classes; $m = 64$

The exponential distribution favors higher classes if x is chosen to be large and, respectively, lower classes if $x \rightarrow 0$.

If the decision for a class is based on an exponential distribution, then the penalty of the algorithm is minimized for $x \approx 1.14$ and reaches $Q_{A_{\log(m)}^{exp}} \approx 25.3 < Q_{A_{\log(m)}^{uni}}$.

4.4.6 Conclusion

Having the results from above, we see that the penalty of $A_{\log_2(m)}$ can be reduced by applying some other than the uniform distribution over the channel classes. Namely, for $m = 64$ we could improve the algorithm by applying either a binomial, a Poisson, or an exponential distribution. Evidentially, $A_{\log_2(m)}$ is improved if small classes have lower probability than medium or large classes.

4.5 Distribution Of The Number Of Blocked Channels

We now assume that the devices still do not know t but that they have knowledge of the probability distribution of t . That is, the jammer blocks exactly $t = t_i$ channels for some $1 \leq i \leq m/2$ with probability $p(t_i)$.

We want to construct an algorithm $A_{m/2}^{PDF(t)}$ which estimates t using the distribution $x_1, \dots, x_{m/2}$ over its $m/2$ guess classes such that the expected running time is minimized. Note that if a device chooses class C_i , this is equal to guessing $\hat{t} = t_i$ and results in considering $2t_i$ channels.

Let C_{d_1} be the elected guess class of device d_1 and, respectively, C_{d_2} be the elected guess class of the other device d_2 in some round. Assume that $t = t_i$. Since the nodes can only meet on unjammed channels, the success probability of the two devices in this round is equal to

$$\frac{\max\{\min\{d_1 - t_i, d_2 - t_i\}, 0\}}{d_1 \cdot d_2}.$$

Let p_i denote the overall success probability if $t = t_i$, i.e. if t_i channels are jammed. It holds that

$$\begin{aligned} p_i &= \sum_{d_1=\frac{t_i}{2}+1}^{m/2} \sum_{d_2=\frac{t_i}{2}+1}^{m/2} x_{d_1} x_{d_2} \frac{\min\{d_1 - t_i, d_2 - t_i\}}{d_1 \cdot d_2} \\ &= 2 \sum_{d_1=\frac{t_i}{2}+1}^{m/2} \sum_{d_2=d_1+1}^{m/2} x_{d_1} x_{d_2} \frac{d_1 - t_i}{d_1 \cdot d_2} + \sum_{d_1=\frac{t_i}{2}+1}^{m/2} x_{d_1}^2 \frac{d_1 - t_i}{d_1^2}. \end{aligned}$$

This leaves us with the following optimization problem:

$$\text{minimize } \sum_{i=1}^{m/2} p(t_i)/p_i$$

$$\text{subject to } \sum_{i=1}^{m/2} x_i = 1$$

Using the method of Lagrange, we obtain an equation system with $\frac{m}{2} + 1$ equations and $\frac{m}{2} + 1$ unknown variables:

$$\left[\frac{\delta}{\delta x_j} \left(\left(\sum_{i=1}^{\frac{m}{2}} p(t_i)/p_i \right) - \lambda \cdot \left(\sum_{i=1}^{\frac{m}{2}} x_i - 1 \right) \right) = 0 \right]_{j=1 \dots m/2}$$

$$\frac{\delta}{\delta \lambda} \left(\left(\sum_{i=1}^{\frac{m}{2}} p(t_i)/p_i \right) - \lambda \cdot \left(\sum_{i=1}^{\frac{m}{2}} x_i - 1 \right) \right) = 0$$

Note that this equation system is linear in all variables and can therefore be solved efficiently, e.g. using the Gaussian method.

5

Multiplayer

After having investigated the two player case, we now discuss the extension to multiple devices. That is, we are given devices d_1, \dots, d_n for $n > 2$ and all of them shall gain knowledge of all the other devices in the system. Note however that this knowledge can also be indirect, i.e. we do not request all pairs of devices to have communicated. Instead, if a device d_i already met some other devices, it may transmit their IDs during the next meeting with some device d_j .

In contrast to when $n = 2$, two devices which successfully meet on some channel have to find other devices thereafter. However, they can of course cooperate in order to search more efficiently. More precisely, they can partition the channels so that they do not search on the same channels at the same time. We state two examples of such a partitioning.

Assume the channels c_1, \dots, c_m to be ordered (and numbered) accordingly to their probability of being chosen by any device and assume the devices d_i and d_j to have successfully met on channel c_k .

Strategy S_1 . *Device d_i considers the even channels while d_j considers the odd channels only. On these respective channel sets, the algorithm of the devices is applied again.*

If more devices are found, then the channels can be partitioned into more subsets of course. Namely, if the devices d_i, \dots, d_{i+k} have knowledge of each others, then d_{i+j} will concentrate on all channels c_h where $(h \bmod (k + 1) = j)$. Note that the order of the devices can be given by their IDs.

It is clear though that the devices have to know how many devices have been found

so far as well as their position, with respect to the IDs, therein. Hence, the devices have to have some management rounds from time to time. However, these extra rounds give only low overhead. For example, the already found devices can be arranged in some balanced tree structure such that the information of the discovery of a new device can be propagated quickly to the root of this tree which then can reassign the channel partitions to all devices in only one time slot.

On the one hand, the number of devices which still can be found decreases with every device which is found. On the other hand, the devices can search more efficiently and less collisions occur. After all, the $\log(n)$ -factor which appears in the Coupon Collector problem (cf Appendix A) can be saved if the devices apply strategy \mathcal{S}_I .

Strategy \mathcal{S}_{II} . *Device d_1 continually stays on channel c_k while d_2 continually stays on channel c_{k+1} .*

As d_1 and d_2 met on channel c_k and as they are ordered according to the probability to be chosen by some device, it is clear that both, c_k and c_{k+1} are not jammed. Hence, if afterwards another device chooses one of these channels in some time slot, there is a successful meeting with probability 0.5, provided that no collision occurs. Note that also the channel c_{k-i} for $1 \leq i \leq k-1$ might not be blocked and hence would serve even better for a continual stay. However, the devices cannot be sure of this and the probability that c_k is one of the most popular unjammed channels is high.

Of course, if there is a device continually staying on every channel c_k, \dots, c_m , then the newly added device evades to channel c_{k-1} . Note that it can be learned quickly whether this is indeed the case.

While in strategy \mathcal{S}_I the order of the channels with respect to their probabilities does not change, this does not hold for strategy \mathcal{S}_{II} of course. Hence, if strategy \mathcal{S}_{II} is applied, it is not clear whether the assumption of a worst case jammer is still justified. Actually, as soon as the devices are able to change their probability distribution over the channels and as soon as they indeed have an intention to do so, the jammer we considered so far can no longer be looked at as a *worst case* jammer.

Furthermore, as two devices can also cooperate in the sense that they can arrange a meeting on a given channel during some given time slots, they can test whether this channel is jammed or not. Doing so, they can trick the worst case jammer and learn the exact value of t . As a result of this, we present further, not as easy to overcome jammer models in the next chapter.

6

Jammer Models

As stated in Chapter 3, we will now discuss further jammer models. To do so, we have to categorize them into pure senders on the one hand and jammers, which are able to send as well as to listen on the other hand. The second group can be further split into simpler ones which can only notice transmission which is being sent over a respective channel and more sophisticated ones which are even able to find out whether at least one device is also listening on that channel.

6.1 Pure Senders

Jammers in this category are, from a hardware perspective, very simple ones. They have no ability to listen on a channel and can therefore not react to what is happening.

6.1.1 Static Jammer

Static jammers are the most simple ones, not only from the hardware but also from the software point of view. Namely, they jam a fixed set of t channels which is never altered. However, this set can be chosen very effectively, as done in the example of the worst case jammer from Definition 3.2.1 which also belongs to this category.

Another approach is to choose the t channels at random. In fact, if the jammer has no information about the devices' probability distribution over the channels, then it cannot do any better than this.

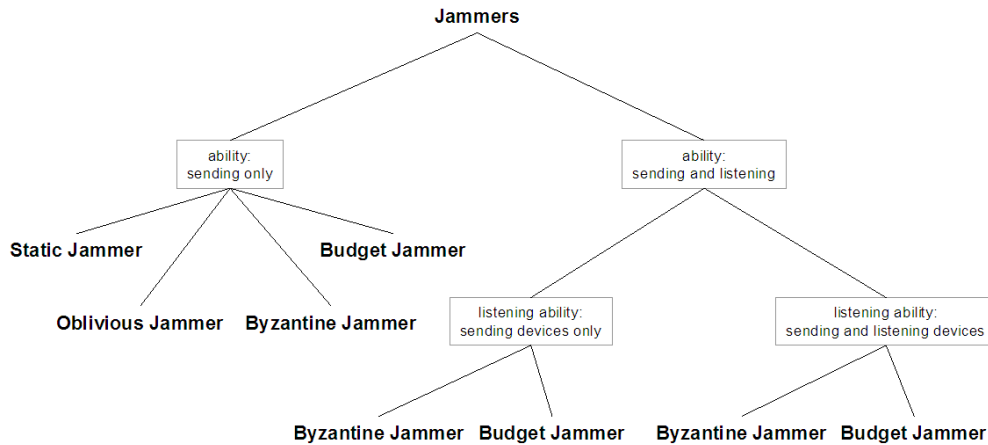


Figure 6.1: Categorization of jammers

6.1.2 Oblivious Jammer

This jammer chooses the t channels to jam in a totally random manner each round. Hence, every channel has the same probability of being jammed in a given time slot. Therefore, the devices can severely increase their probability of success if they try to find each other on only one channel they previously agreed on.

As it is so easy to cope with such a jammer, the question arises why a random jammer should be employed at all. In fact, if only the problem of connection establishment is concerned, an oblivious jammer is indeed deficient. However, once a connection between two devices is successfully established, an oblivious jammer can jam their communication a lot more effectively than a static jammer. This holds because the devices can quickly locate and thus avoid the latter whereas this is not possible with an oblivious jammer.

Furthermore, note that the strategy of looking for each other on one channel only is good if there are only two devices. But in the multiplayer case it requires the devices to either have a diminished probability of sending or other precaution to avoid collisions, such as e.g. a random backoff policy after having successfully received the message of another device.

6.1.3 Byzantine Jammer

The next jammer we consider may or may not change the currently jammed channels for the next round. However, it is not clear on which circumstances this jammer should base its decision. Strictly speaking, as we are considering jammers which are not able to listen, there is no point in time for the byzantine jammer to have an intention to alter the set of jammed channels.

However, as it has the possibility to do so randomly, it is clear that the devices can neither treat a byzantine jammer like a static jammer nor like a random jammer. In the multiplayer case, two devices which have found each other on some channel c_i should therefore not follow the strategy \mathcal{S}_{II} from Chapter 5 as the channels c_i and c_{i+1} might get blocked by the byzantine jammer once thereafter.

6.1.4 Budget Jammer

Finally, the even more powerful budget jammer can jam at most $t \cdot r$ channels during r rounds. That is, it can decide to jam less than t channels in one round and instead use the energy it thereby saved to jam accordingly more than t channels in subsequent rounds. Note however that it can of course never overstrain its energy account.

It is clear that the budget jammer can act like the worst case jammer from Section 3.2. But even if it has more power in the sense of freedom to vary its behaviour, it cannot delay the success of the devices more than the worst case jammer, assuming the algorithm of the devices to be static in the sense that it acts equivalently in every round. To see this, let us assume a budget jammer J_B is jamming the same set of channels C like a worst case jammer in some round i but decides to forego to jam $c_k \in C$ in round $i + 1$. Since the t channels with highest probability are in C , it is clear that J_B can never compensate the success probability it thereby missed to destroy by jamming a channel $c_l \notin C$ in some round $j > i$.

Hence, the algorithms we derived in Chapter 4 are at least as good for any budget jammer than they are for the worst case jammer. Note however that this only holds for the category of pure senders.

6.2 Listening Senders

In contrast to pure senders, listening senders have the additional ability to also listen on some channels and hence to check whether a device is currently transmitting there. This allows these jammers to react to the momentary situation. We assume such a jammer to have a number $t_l \leq t$ of channels to listen on and, respectively, $t_s = t - t_l$ channels to send on per round, where these two values can vary dynamically from round to round. However, we do not consider jammers which are able to listen on a channel at the beginning of a time slot to then, if they receive some signal, immediately start to jam it.

We distinguish between jammers which can only notice transmissions and jammers which can also determine whether another device is also listening on the respective channel.

In turn, we restrict us to the multiplayer case as in the case where $n = 2$ it is of no use for a jammer to listen on some channels. Either the devices did not succeed, then

the jammer should go on as before, or they did succeed, but then the jammer already failed.

6.2.1 Ability To Notice Senders

Byzantine Jammer

A byzantine jammer can now try to detect the point in time when two devices successfully meet. Namely, it can resign to jam some channels and instead listen on them. Doing so, it can become aware of changes in the channel patterns of two devices d_1 and d_2 , which indicate that they have been successful.

If the devices follow the strategy \mathcal{S}_{II} presented in Chapter 5, the jammer could exchange two of its so far jammed channels with channels c_i , through which the devices were successful, and c_{i+1} , and thus totally prevent a third device from being found by d_1 or d_2 respectively. However, the two devices could have agreed on a time slot in the future to meet on the same channel again and thus would quickly learn that the jammer indeed aims at destroying their strategy.

Even if the devices follow the other strategy from Chapter 5, i.e. strategy \mathcal{S}_I , the jammer has an advantage now and could for example concentrate on d_1 to make it more difficult for it to find further devices.

Budget Jammer

Compared to the byzantine jammer, a budget jammer has the advantage that it can not only totally prevent a third device from being found by d_1 or d_2 if these two devices follow the strategy \mathcal{S}_{II} from Chapter 5, but it can also save power by focussing only on them. This in turn can then be used later to jam more channels and hence disturbing other devices.

6.2.2 Ability To Notice Senders And Receivers

Byzantine Jammer

The advantage for the byzantine jammer when it can also notice listening devices is that it immediately detects successful meetings if they happen on a channel which it is observing. This means that it can react more quickly as it can save the rounds to realize changes in the channel patterns.

Budget Jammer

Of course, the consideration above also holds for budget jammers. Actually, as this saves even more power, such a jammer can also benefit more from this improved ability.

7

Simulations

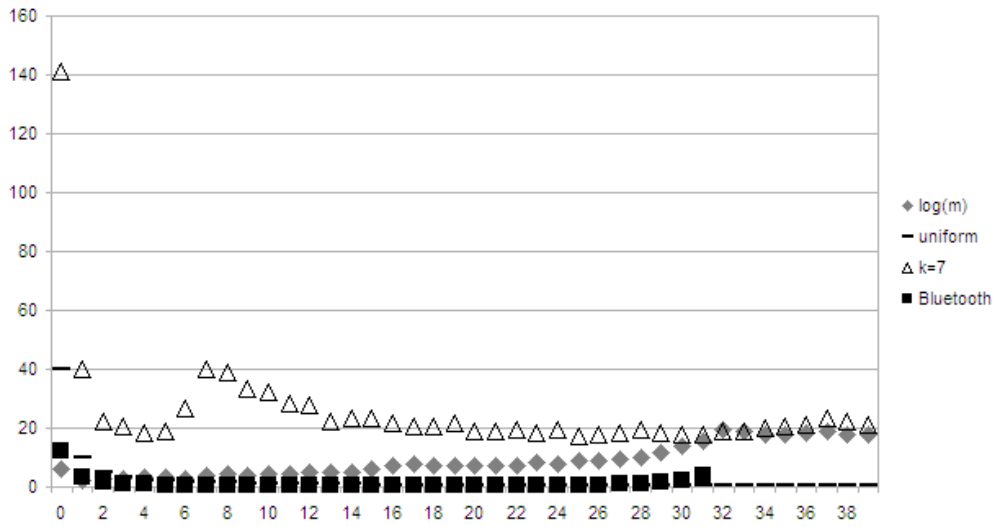
To conclude our considerations, we now give the results of some simulations of the algorithms we derived in Chapter 4.

In the first two simulations we compare the algorithms $A_{\log_2(m)}$, $A_{k=7}$, A_{uni} (which chooses any channel uniformly at random) as well as the inquiry algorithm from Bluetooth. The technical details of the inquiry algorithm from Bluetooth can be found in [14]. However, we slightly adapt it in order to be comparable to our algorithms. First, we assume that an inquiring device can only send on one channel per time slot and second, we do not repeat a train for 256 times as we assume the devices not to be sleeping during the inquiry phase.

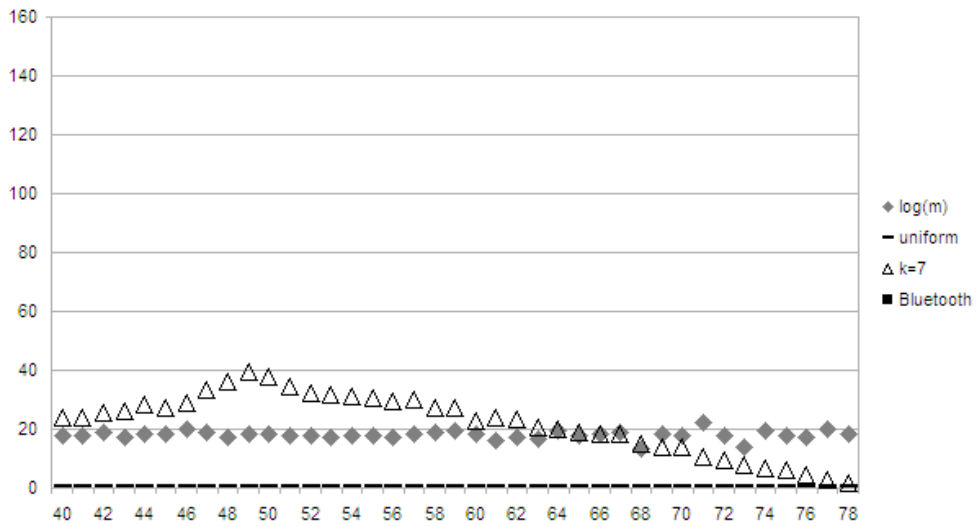
Figure 7.1 shows the expected running times of the four algorithms divided by the expected running time of A_{opt} as a function of t . As in the application of Bluetooth, we set $m = 79$ and we use a worst case jammer as adversary.

The Bluetooth inquiry algorithm behaves very good under these circumstances. However, as it uses only 32 of the available channels, a connection establishment is of course already impossible for $t \geq 32$. As expected, $A_{\log_2(m)}/A_{opt}$ is approximately constant for $t \geq 40$ whereas this ratio is even 1 in the same range for A_{uni} , since it then behaves exactly like A_{opt} .

Even if it is approximately constant, it is clear that the ratio of $A_{\log_2(m)}$ is relatively high. However, this algorithm has been designed for a considerably larger number of available channels. For m being rather small, it is a good idea though to pick any channel uniformly at random.



(a) $0 \leq t \leq 39$



(b) $40 \leq t \leq 78$

Figure 7.1: Expected running times of various algorithms divided by the expected running time of A_{opt} for the adversary being a worst case jammer; $m = 79$

Figure 7.2 presents the running times of the four algorithms when a random jammer, as described in Section 6.1.1, is applied. Note that it has semi-logarithmic scaling.

Again, the expected running time of the Bluetooth inquiry algorithm is equal to infinity for $t \geq 32$.

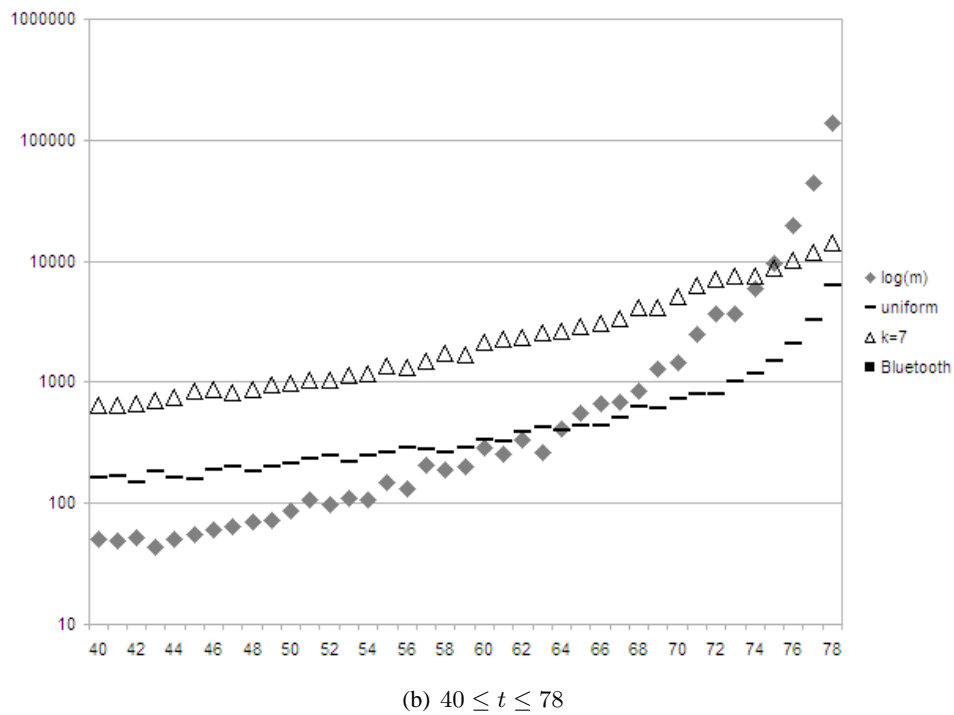
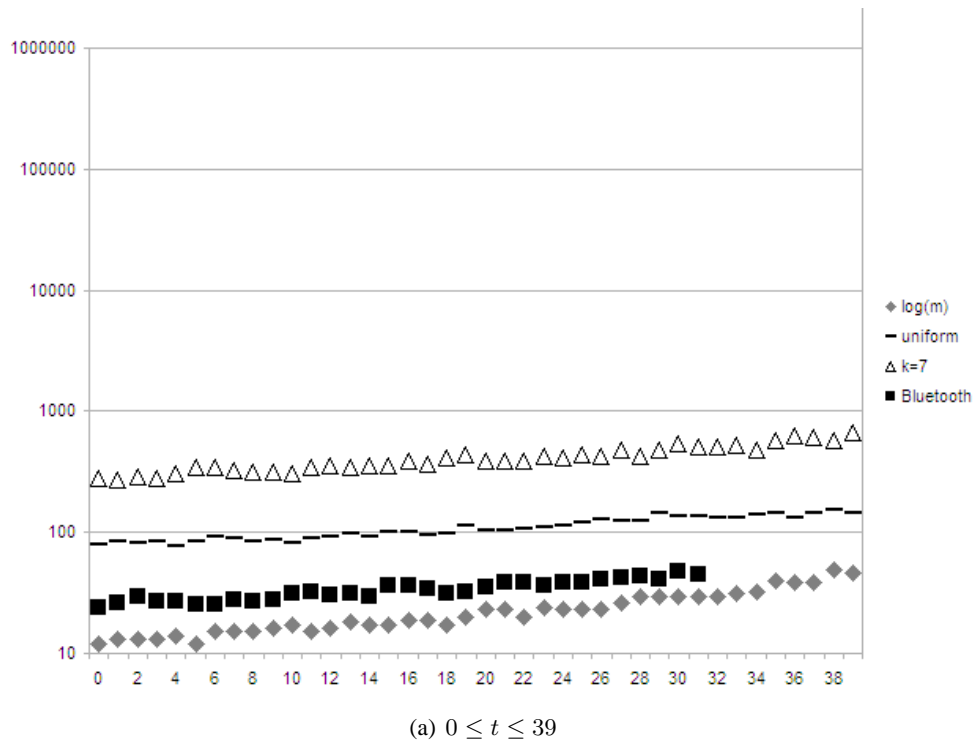


Figure 7.2: Expected running times of various algorithms for the adversary being a random jammer; $m = 79$

In a further case study we examine the influence of a microwave oven to the performance of $A_{\log_2(m)}$ on the one hand and to the performance of the Bluetooth inquiry algorithm on the other hand. A common microwave oven operates at 2.45GHz and thus interferes with the channels used by the Bluetooth inquiry algorithm.

Our setting consists of a microwave oven and four devices, two of them applying $A_{\log_2(m)}$ while the others two are Bluetooth nodes. Note that even if the microwave oven is off, the connection establishment of the two devices of one kind is disturbed by the interfering connection establishment of the two devices of the other kind.

The simulation has been run 10000 times and the average numbers of attempts needed for connection establishment are listed in Table 7.1 which shows that a microwave oven hardly influences the performance of $A_{\log_2(m)}$ but severely delays the connection establishment of Bluetooth devices.

| | microwave oven off | microwave oven on |
|-------------------|--------------------|-------------------|
| $A_{\log_2(m)}$ | 13.1 | 13.6 |
| Bluetooth inquiry | 32.6 | 49.0 |

Table 7.1: Average number of attempts needed for connection establishment

Finally, we examine the multiplayer case when the devices apply algorithm $A_{\log_2(m)}$ and strategy \mathcal{S}_1 from Chapter 5. Figure 7.3 presents the running times for $2 \leq n \leq 20$ when the adversary is a worst case jammer or, respectively, a random jammer.

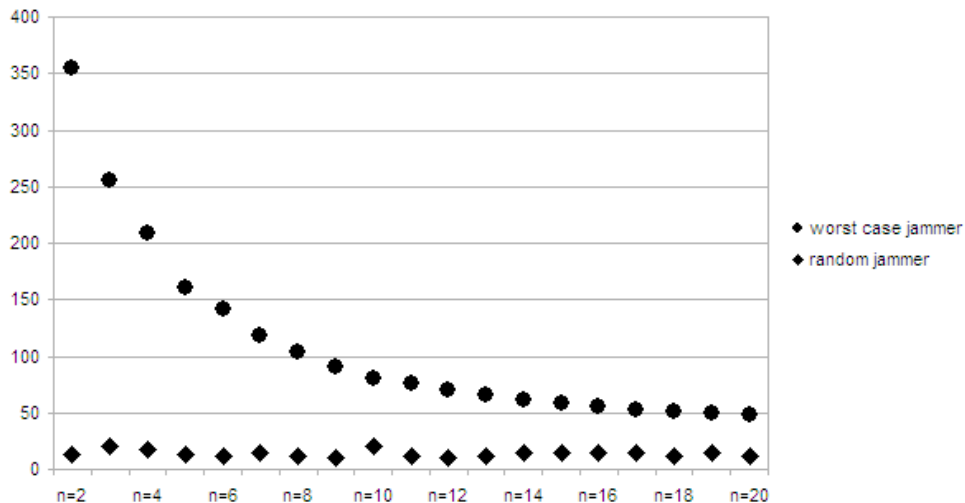


Figure 7.3: Expected running times of $A_{\log_2(m)}$ for $2 \leq n \leq 20$; $m = 79$ and $t = 10$

It is obvious that cooperation as described by strategy \mathcal{S}_1 really helps to signifi-

cantly decrease the number of time slots which are needed for the detection of multiple devices for the adversary being a worst case jammer. However, the expected running time changes only slightly if t channels are blocked by a random jammer.

8

Conclusion

It is a fundamental task for wireless devices to quickly discover potential communication partners. This task is further complicated by other devices, possibly of an entirely different kind, which operate in the same restricted frequency spectrum and thus cause collisions. Even worse, such collisions can be caused intentionally by malicious devices which jam some of the available frequencies.

In this thesis, starting by stating the optimal algorithm if t , the number of jammers, is known, we have established algorithms which allow the devices to quickly find each others despite the presence of an unknown number of jammers. We first designed algorithms with different numbers of guess classes, representing the optimal solution if the guess for t is correct. We then proved that the approach of describing the algorithm by guess classes is indeed as powerful as directly defining a probability distribution over the channels.

However, we did not prove lower bounds on the performance of algorithms for connection establishment. It is clear that it would be of great interest to further research in this direction, i.e. to find the optimal k of the algorithm A_k or even the optimal distribution over the channels in general.

We severely focused on the two player case, i.e. $n = 2$, and only shortly discussed the multiplayer case in this thesis. Obviously, this allows for extension in further examination, including for example multihop applications. Moreover, other techniques, such as e.g. orthogonal codes, which have not been in the scope of this thesis, can play an important role in a multiplayer study.

Although our simulations show that the algorithms we designed serve well for the problem of device discovery, more extensive simulations could be performed. An

especially interesting case study would be to extend the Bluetooth inquiry algorithm, as a state-of-the-art protocol for wireless connection establishment, to larger numbers of available frequency channels. In the simulations we have carried out for this thesis, the Bluetooth algorithm outperforms our algorithms if $t < 32$. However, it would be interesting to see whether this still holds for $m \gg 79$, as our algorithms are especially powerful for a large number of frequency channels.

Finally, we would also like to mention that the set of jammer models we described in this thesis can be further extended of course.

Acknowledgment

The author would like to thank his advisors Yvonne Anne Oswald and Stefan Schmid, who received his doctoral degree during the development of this thesis, as well as Prof. Dr. Roger Wattenhofer for numerous extensive and helpful discussions. Furthermore, special thanks go to Christoph Lenzen for very useful mathematical hints.



Coupon Collector Problem

The Coupon Collector problem is defined as follows: There is a pot with n numbers and we are allowed to pick one of these numbers per round. We register the number and put it back into the pot. The goal is to register every number at least once and we are interested in how many rounds we need in expectation to reach this goal.

Let X_i denote the event of picking a new number, given that we already registered i numbers. The expected running time until we reach the goal is

$$\begin{aligned} \mathbf{E}[\text{number of rounds to reach the goal}] &= \mathbf{E}[X_0] + \mathbf{E}[X_1] + \cdots + \mathbf{E}[X_{n-1}] \\ &= \frac{1}{\mathbf{Pr}[X_0]} + \frac{1}{\mathbf{Pr}[X_1]} + \cdots + \frac{1}{\mathbf{Pr}[X_{n-1}]} \\ &= \frac{n}{n-0} + \frac{n}{n-1} + \cdots + \frac{n}{n-(n-1)} \\ &= n \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1} \right) \\ &= n \cdot H_n \end{aligned}$$

where H_n is the n -th harmonic number for which it holds that $H_n \in \mathbf{O}(\ln(n))$.

Bibliography

- [1] Ghada Alnifie and Robert Simon. A Multi-channel Defense Against Jamming Attacks in Wireless Sensor Networks. In *Proc. 3rd ACM Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, pages 95–104, 2007.
- [2] Baruch Awerbuch, Andrea Richa, and Christian Scheideler. A Jamming-Resistant MAC Protocol for Single-Hop Wireless Networks. In *Proc. 27th Annual Symposium on Principles of Distributed Computing (PODC)*, 2008.
- [3] Jerry T. Chiang and Yih-Chun Hu. Cross-layer Jamming Detection and Mitigation in Wireless Broadcast Networks. In *Proc. 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 346–349, 2007.
- [4] Clayton W. Commander, Panos M. Pardalos, Valeriy Ryabchenko, Stan Uryasev, and Grigoriy Zrazhevsky. The Wireless Network Jamming Problem. In *Air Force Research Laboratory, Tech Report 07-11-06-332*, 2007.
- [5] Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport. Gossiping in a Multi-Channel Radio Network (An Oblivious Approach to Coping With Malicious Interference). In *Proc. 21st Ann. Conference on Distributed Computing (DISC)*, 2007.
- [6] Seth Gilbert, Rashid Guerraoui, and Calvin Newport. Of Malicious Motes and Suspicious Sensors. In *Proc. International Conference on Principles of Distributed Systems (OPODIS)*, 2006.
- [7] Chiu-Yuen Koo, Vartika Bhandari, Jonathan Katz, and Nitin H. Vaidya. Reliable broadcast in radio networks: the bounded collision case. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 258–264, New York, NY, USA, 2006. ACM.
- [8] Srinivasan Krishnamurthy, Mansi Thoppian, Srikant Kuppa, R. Chandrasekaran, Neeraj Mittal, S. Venkatesan, and Ravi Prakash. Time-efficient distributed layer-2 auto-configuration for cognitive radio networks. *Comput. Netw.*, 52(4):831–849, 2008.
- [9] Yee Wei Law, Lodewijk van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient Link-layer Jamming Attacks Against Wireless Sensor Network MAC Protocols. In *Proc. 3rd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN)*, pages 76–88, 2005.

- [10] Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. In *INFOCOM*, 2007.
- [11] Xin Liu, G. Noubir, R. Sundaram, and San Tan. Spread: Foiling smart jammers using multi-layer agility. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2536–2540, May 2007.
- [12] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein. Using channel hopping to increase 802.11 resilience to jamming attacks. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2526–2530, May 2007.
- [13] Andrzej Pelc and David Peleg. Feasibility and complexity of broadcasting with random transmission failures. *Theor. Comput. Sci.*, 370(1-3):279–292, 2007.
- [14] Brian S. Peterson, Rusty O. Baldwin, and Jeffrey P. Kharoufeh. A specification-compatible bluetooth inquiry simplification. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*, page 90307.1, Washington, DC, USA, 2004. IEEE Computer Society.
- [15] Marvin K. Simon, Jim K. Omura, Robert A. Scholtz, and Barry K. Levitt. *Spread spectrum communications handbook (revised ed.)*. McGraw-Hill, Inc., New York, NY, USA, 1994.
- [16] Mario Strasser, Christina Pöpper, Srdjan Capkun, and Mario Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *IEEE Symposium on Security and Privacy*, pages 64–78, 2008.
- [17] Y. C. Tay, K. Jamieson, and H. Balakrishnan. Collision-Minimizing CSMA and Its Applications to Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 22(6), 2004.
- [18] Anthony D. Wood, John A. Stankovic, and Gang Zhou. DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks. In *Proc. 4th Annual IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2007.
- [19] Wenyuan Xu, Ke Ma, Wade Trappe, and Yanyong Zhang. Jamming Sensor Networks: Attack and Defense Strategies. *IEEE Network*, 2006.
- [20] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *Proc. 6th ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc)*, pages 46–57, 2005.
- [21] Wenyuan Xu, Timothy Wood, Wade Trappe, and Yanyong Zhang. Channel Surfing and Spatial Retreats: Defenses against Wireless Denial of Service. In *Proc. 3rd ACM Workshop on Wireless Security (WiSe)*, pages 80–89, 2004.