Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
SCIENZE DELL'INGEGNERIA

CICLO XXII

COORDINATORE Prof. Stefano Trillo

SISTEMI PER LA MOBILITÀ DEGLI UTENTI E DEGLI APPLICATIVI IN RETI WIRED E WIRELESS

Settore Scientifico Disciplinare ING-INF/03

Dottorando
Dott. Bellettini Carlo

Tutore
Prof. Mazzini Gianluca

Anni 2007/2009

Contents

List of Figures

List of Tables

On the Concept of Mobility

The words *mobility* and *network* are found together in many contexts. The issue alone of modeling geographical user mobility in wireless networks has countless applications. Just to name a few:

- resource allocation, typically bandwidth;
- pervasive computing, in a world where every object is a computing device that can be of help for persons or other devices;
- service continuity: in the Internet realm, this is the case of the various mobile IP solutions, but we can also focus on the lower-level issue of physical hand-off, possibly between heterogeneous networks;
- location-aware services, from navigation, to advertising;
- disruption-tolerant networks, for example in vehicular, military, social, or cooperative applications.

Depending on one's background, the concept is investigated with very different tools and aims. Given also the huge number of works, therefore, it would simply not be practical to cite the "most relevant" contributions to a "field" so difficult to define. Nevertheless, for highlighting once more the broadness of the subject, let us cite [1], where Nokia points out the scientific and commercial need of considering even cultural factors when answering the "who is doing what and where" question.

It may seem that now there is not much to add to the concept of *mobility*, but it is not so. Actually, the last decade saw also a growing interest in *code* mobility, i.e. the possibility for software applications (or parts thereof) to migrate and keeps working in different devices and environments.

The usual idea is that of stateful *agents*, which can carry with them all the code (i.e. the program itself) and the data they need. A notable real-life and successful application is distributed computing, which under certain hypothesis can void the need of expensive supercomputers. The general rationale is splitting a very demanding computing task (as protein folding or integer factorization) into a large number of independent sub-problems, each addressable by limited-power machines, weakly connected (typically through the Internet, the quintessence of a wired network).

One of the most extreme visions, however, is probably *ActiveNet* [2], a replacement for Internet. Not just a fancy name, but rather a thorough reworking of passive packet-switched networks that would evolve packets into "capsules", a sort of micro-agents. Eventually, *any* participating node (from routers to user devices) would perform some computation, according to the instructions embedded in the capsule received or relayed.

However appealing this proposal might be, we decided to concern ourselves with present-day issues. In particular, we organized this thesis in two distinct and independent parts. We of course refer the reader to the respective introductory sections for the details, but let us anticipate that:

- **Part I**

  deals with *audio fingerprinting*, and a special emphasis is put on the application of broadcast monitoring and on the implementation aspects[1]. Although the problem is tackled from many sides, one of the most prominent difficulties is the high computing power required for the task. We thus devised and operated a distributed-computing solution, which is described in detail. Tests were conducted on the computing cluster available at the Department of Engineering of the University of Ferrara.

[1]Part of this work is being used commercially.

- **Part II**

  focuses instead on wireless networks. Even if the approach is quite general, the stress is on WiFi networks. More specifically, we tried to evaluate how mobile-users' experience can be improved. Two tools are considered. In the first place, we wrote a simulator and used it to estimate the impact of pricing strategies in allocating the bandwidth resource, finding out the need for such solutions. Secondly, we developed a high-level simulator that strongly advises to deepen the topic of user cooperation for the selection of the "best" point of access, when many are available. We also propose one such policy.

Part I

Mobility of Applications: a Framework for Audio Fingerprinting

Chapter 1

What is Audio Fingerprinting?

Besides the unquestionable scientific interest, the availability of vast, digital archives of music and the related commercial interests are pushing many efforts into the field of automatic audio recognition. The opportunity of exploiting more computing power, and at steadily decreasing costs, certainly plays a key role as well. Indeed, chances are that what a decade ago would have been practical only by means of specialized hardware could now be accomplished in software (sometimes even on portable devices).

Though related to speech recognition, this truly multiform and demanding topic is actually addressed in distinct ways. Audio recognition may aim to discern whether or not two pieces are in fact the same, regardless of their outer appearance (i.e. coding, distortions). More generally, it comes into play when we are interested in reliably identifying an unknown excerpt of e.g. music, given a large set of references. A first glimpse of its rationale and possible applications is given in Fig. 1.1.

The task of automated audio recognition used to be accomplished by using invasive watermarking techniques. However, this requires either permanent human intervention, or that a single watermarked source accounts for every possible instance of a given pattern. Neither method is feasible: the former needs an active listener who correctly marks pieces, whereas the latter is clearly not realistic. A different approach is therefore needed.

A working, and actually very good, solution is the so-called audio fingerprinting [3, 4], known also as robust audio hashing. Its purpose is to allow electronic devices to

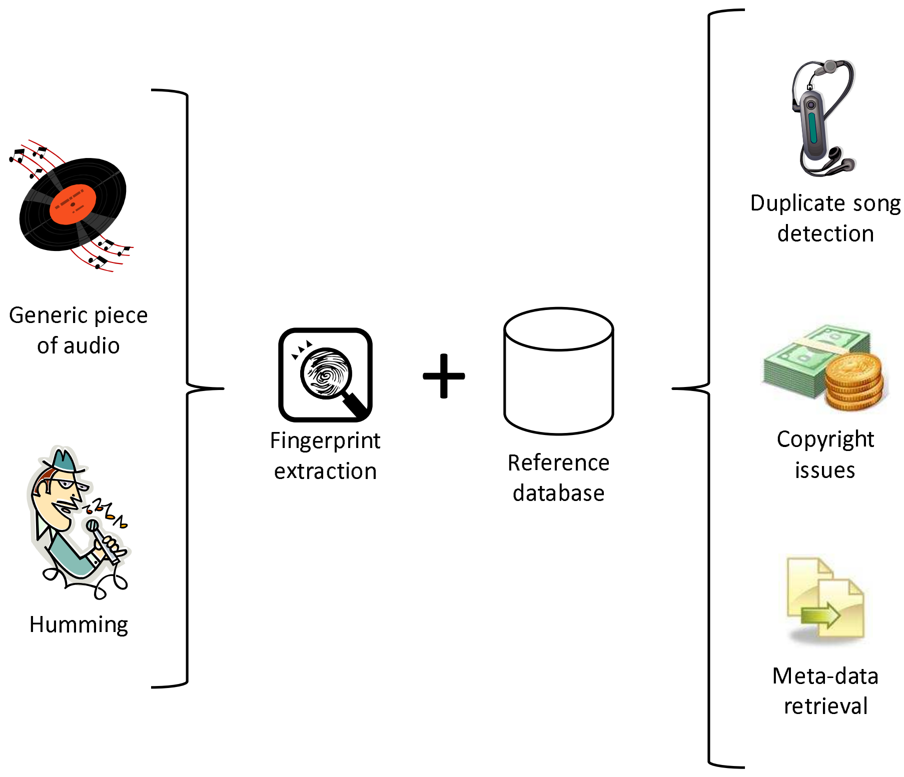


Figure 1.1: A simplified overview of what audio fingerprinting is.

identify perceptual similarities between audio contents. The term “fingerprint” recalls the fact that every piece of audio bears unique features, as detectable by listeners.

The usual system in which a fingerprinting algorithm operates, sees the building of a large fingerprint database, used as a reference source for identifying unknown fingerprints (compare also Fig. 2.1). Note that the search task is heavily demanding, since we are supposed to find the most similar (not simply exact) match in a huge amount of data (e.g. some 100 000 song fingerprints).

Such tight search time requirements (a gain factor of roughly 100, at least, should generally be attained over real time) and its high complexity also led to putting considerable effort into the matching problem. In [5], for example, gene-sequence matching algorithms are borrowed from biology, allowing for a lightning-fast identification of an event-based fingerprint, computable over the output of any feature extraction algorithm.

On our side, we focused our attention on a simple yet very robust algorithm, whose basic operation was first described by Haitsma et al. in [6]. The distinctive feature it extracts is intimately related to the audio signal energy. More specifically, it takes into account how the energy difference among a set of sub-bands varies in time. In other words, it evaluates the second-order derivative of the energy spectrogram. The reasons that drove our choice and the details of the algorithm are provided later in 2.1.

We analyze in this work many essential aspects of algorithm [7]. We chose it as the core of a more general framework for audio fingerprinting, which we will describe in detail. Focusing especially on the application of broadcast monitoring, we also provide effective strategies for improving the overall system and discuss the results of plentiful experiments, based on a database of approx. 100 000 songs. By broadcast monitoring we mean the continuous tracking of an audio source such as a radio channel. Musical TV and satellite channels may also be processed in a similar fashion, even if a combined approach including their video component would probably be a better choice. But let's take a look at how audio fingerprinting has been performed so far.

As we pointed out, the interest in the field has been steadily growing in these years, thus producing quite a large amount of good contributions. Significant reviews can be found in the cited [3, 4] and also in [8], but they date back to 2005 and cannot account for the latest developments.

First of all, we stress that applications are really manifold. For example, copyright issues could be easily addressed in p2p file-sharing networks or video sharing services, while meta-data retrieval and broadcast monitoring [5, 9] can be achieved. A music consumer could fast check whether his or her large collection already contains a given song, too [10]. Another nice application is the so-called "query by humming" [11], an early work in multimedia searching. A similar algorithm is now freely available on-line [12].

According to the researchers background and goals, the topic can be addressed in many different ways. The first step invariably involves isolating a sequence of these "features" in the piece of audio, and the longer the piece of audio, the more the features. This set of features is said to be the "fingerprint" of the piece.

Most of all, the fingerprint must be sufficiently distinguishable, so that two fingerprints can be reliably told apart or regarded as similar [13]. Along with the retrieving technique, it must prove also robust, i.e. exhibit high reliability even when various kinds of distortions occur, such as equalizing, white noise, pitching and so on. Finally, it must be fast to compute from e.g. a PCM (Pulse Code Modulation) signal, and also achieve a significant reduction factor in size, with respect to the original PCM samples.

On the basis of these considerations, we can also speak of "robust hashing" since, similarly to standard hashing techniques, audio fingerprinting aims to produce a compact and fast-to-check-and-retrieve representation of a complex audio signal. Usually, as in our case, there is the need to work only on the waveform representation of the audio signal, with no further semantic aid (as a score would be). Needless to say, regular hash functions cannot be employed, since they rely on the precise bit sequence of the particular digitalization of the audio signal.

A large selection of different audio features can be extracted, usually chosen on the basis of heuristic considerations, with a few exceptions [14]. It is also possible to combine more of them into one single identification model [15], aiming to mutually compensate eventual weaknesses. However, this can be computationally expensive, if not prohibitive beyond small databases. Other less common approaches include Principal Component Analysis (PCA), as in [16] and [17], or statistical modeling combined with Hidden Markov Models (HMMs), for example [18]. Although effective, it is often difficult to render them efficient enough for real-time employment.

The usual feature extracting technique involves a combined time-frequency analysis, mainly performed through Fourier transforms. A few works propose instead wavelet transforms, which may be a more rewarding choice [19–21]. A combination of both a FFT and a DCT is then employed in [22]. This paper is particularly interesting since it builds on [6], as our approach does, but introduces a further processing stage (which involves the DCT), in order to improve the indexed search reliability. The technique is especially effective when the audio query is rather distorted. However, it is not clear to what extent this extra stage elongates processing time. Moreover, since we focus on broadcast monitoring, the added complexity would not pay off, given the relatively mild distortions broadcast audio usually faces, for which the solutions we suggest in 3 prove very effective.

On the retrieval problems, we point out the contributions [23–25], which mathematically argument effective strategies. In particular, the first two borrow respectively from classical full-text indexing and coding theory. As far as the latter is concerned, we stress that its applicability in some contexts is limited by the use of short clip fingerprints, instead of whole songs, and in this respect it is not trivial to foresee its possible efficiency, when extended. A second drawback is that, at its present state, it is not able to track an audio stream (e.g. to evaluate the duration of a broadcast song), since the scheme is inherently incapable of handling cropping. For this reason, there are even cases in which two different songs could be regarded as identical, for example when their fingerprinted excerpts are a same musical section, while the sung parts differ (i.e. different performers).

This first part of the thesis is organized as follows. In the next chapter, we discuss the main points behind audio fingerprinting and a broad selection of existing works. The system is detailed in chapter 2, while in 3 we introduce strategies for improving its performance, both in speed and reliability (the latter with particular regards to the broadcast monitoring application). In chapters 4 to 6, we present our scalable testbed, a viable solution to the pitch-shift distortion and the robustness of the algorithm to thermal noise. Some further scalability issues are addressed in 7 and we draw our conclusions in 7.3.

Chapter 2

System Description

As previously hinted, our system follows the usual architecture of three blocks (see Fig. 2.1): the feature extractor, the fingerprint database and the seeker. The extractor must be fed with some audio input, while the seeker produces a match. Such match can be found by comparing against the reference database.

Though sufficient for evaluation purpose, these blocks fall short in real-world applications. For example, we may be interested in processing the output of a microphone or of a radio receiver. Therefore, a transcoder must be used to properly format the input. Our implementation takes audio files as input, handling a variety of compressed and non-compressed formats, and then works on raw waveforms. In particular, MP3 files (or parts thereof) were used as input.

Post-processing is also needed, first of all to correctly interpret the outcome of the seeker, say match a given id to the correct metadata. With the aid of Fig. 2.1, let's now examine in detail how things work.

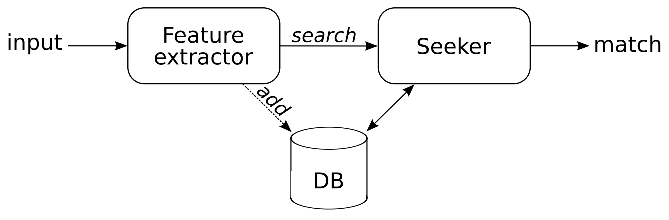


Figure 2.1: Overview of the fingerprinting system.

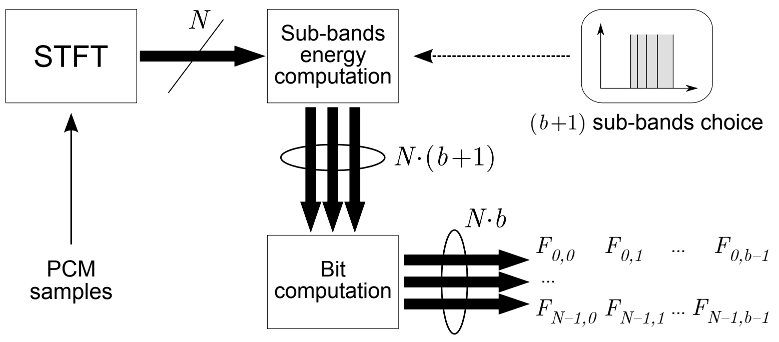


Figure 2.2: The Streaming Audio Fingerprinting (SAF) algorithm.

2.1 Feature Extractor

The feature extractor is the most relevant block, since it greatly accounts for the overall effectiveness of the system. On the other hand, a highly efficient seeker must be implemented (see 2.4).

We recall that the feature extractor is an algorithm that must produce the fingerprint of a piece of audio, by selecting some of its distinguishing characteristics. Among the many methods investigated in the last years, we selected that proposed by Philips in 2001, in its revised and extended version [7]. Beyond its experimented effectiveness, we mainly based our choice on its great flexibility. According to its original proposers [26], we will denote the feature extractor algorithm as Streaming Audio Fingerprinting (SAF), but it is also known in the literature as Philips Robust Hash (PRH).

The Philips approach provides a convenient and easy-to-handle output (a bit matrix), and ample opportunity of customization. Moreover, its simple steps allow some kind of statistical modeling, though only approximated [27, 28].

Let's follow Fig. 2.2. The input is constituted by raw PCM samples, whose frequency must be 44 100 Hz, with a quantization level of 16 bit/sample[1]. A stereophonic input is immediately converted to monophonic by averaging the two channels.

Since the subsequent processing discards much of the original bandwidth, the input

[1]We observe that these limitations can be easily circumvented by the pre-processing stage, before feeding the feature extractor.

is downsampled to 5 kHz by a low-pass filter and a decimator. Its integer period varies so to have, on average, the correct downsampling ratio. In the present case of PCM samples extracted from MP3 files, we verified that a rough equiripple 16-tap FIR filter (designed with Parks-McClellan algorithm) gives the same results as a more refined 41-tap filter, which should be the minimum order for the required specifications. Therefore, we used the faster 16-tap filter.

As is often the case in the audio analysis field [29], the algorithm works on the input spectrogram. It can be obtained by taking first a Short-Time Fourier Transform (STFT) of the audio samples, and then the squared modulus of the transform (energy). This gives N frames in the frequency domain, where N is linearly proportional to the length of the input.

The three steps for the STFT are summarized in Fig. 2.3:

1. input segmentation into N *overlapping* frames;
2. windowing of each frame according to a convenient function;
3. discrete Fourier transform by means of a FFT (Fast Fourier Transform) algorithm.

Here, the length of each frame is 16 384 samples (or approximately 0.37 s), while the overlap leaves out only 512 samples, or $\frac{1}{32}$ of the frame length. These parameters lead to a good trade-off between frequency and time resolution. Such a large overlap is crucial in the further step of recognizing unknown excerpts. Not only does it allow a comparison between misaligned framing structures, but it also makes the system more robust to distortions: a good algorithm should always produce fingerprints that look very much alike, regardless of the particular framing offset.

The window function considerably smooths the frame to be transformed and alleviates the problem of the spectral bias in the spectrogram. As in [7], the choice fell on the Hann window, which has been proved to be nearly optimal as to its bias characteristics in this context.

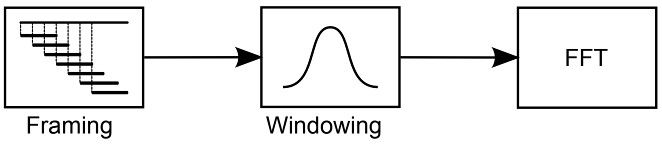


Figure 2.3: Rationale of the Short-Time Fourier Transform.

As a further step, each transformed frame is sliced into $b+1$ bands. Differently from the original paper, we used as reference the usual 12-TET tuning system, keeping frequencies up to about 1976 Hz and setting the lower bound l as a function of b, given the constraints imposed by the musical scale. In particular, its value in Hz is given by

$$l = 440 \cdot 2^{\frac{26-b}{12}} \quad .$$

This division is beneficial since it allows to successfully combat the pitch distortion, an intrinsic weakness of the original system. The bandwidth kept depends on the particular value of b. The details are presented in chapter 5.

The distinguishing feature considered by the algorithm is the energy difference of sub-bands, both among them and in time. By summing every relevant sample, the energy of each sub-band is computed as a contribution $E_{n,m}$ where $0 \leq m \leq b$ is the sub-band index and $0 \leq n \leq N-1$ is the frame index. A value $\mathrm{FP}(n,m)$ is finally obtained according to the rule

$$\mathrm{FP}_{n,m} = \begin{cases} 1 & \text{if} \quad \alpha_{n,m} - \alpha_{n-1,m} > 0 \\ 0 & \qquad \text{otherwise} \end{cases} \quad ,$$

where $\mathrm{FP}_{n,m}$ (for which $0 \leq m \leq b-1$) is the m-th bit relative to frame n and α is the energy difference among contiguous sub-bands, namely

$$\alpha_{n,m} = E_{n,m} - E_{n,m+1} \quad .$$

We highlight that there has been at least one proposal to improve the binarization [30], but at some more computational expense. Note that, for a given n, $\mathrm{FP}(n,\cdot)$ is a b-bit vector spanning the considered bandwidth and representative of a very tiny fraction of the whole song. Conversely, when m is fixed, $\mathrm{FP}(\cdot, m)$ is a N-bit vector which roughly gives the energy variation trend in a sub-band. The sequence of the N b-bit vectors, obtained from the N temporal frames, is the fingerprint of the input piece of audio. An example is depicted in Fig. 3.5a, where the patterns are stacked vertically and the frequency dimension is horizontal.

It is worth noting that, given the length L in seconds of an excerpt (sampled at a sample frequency f_s), the number N of patterns (lines) extracted as fingerprint may be easily computed as

$$N = \lfloor \frac{L \cdot f_s - \mathrm{frame\_length}}{\mathrm{skip}} \rfloor + 1 \quad .$$

2.2 Fingerprint Database

The database is basically an archive of song fingerprints. As we have seen in the introduction, many different approaches can be exploited to manage and search the database, also depending on the particular representation of the extracted features. Here, the database is a collection of binary files, possibly organized in distinct folders, each representing a single and whole song. It currently holds 100 000 entries, highly differentiated in genre and mostly of western origin. Its actual organization is detailed in 4.1.

Normally, no extra-processing is performed over the extracted fingerprints, though a further layer could be easily inserted here to provide additional functionalities. For example, it would make the database either more compact, to save storage space, or more convenient, to save search time and to be more reliable or robust in seeking. An example is introduced in 3.4.

2.3 Distance Metric

Regardless of the particular method employed for searching, a distance metric must be defined between the fingerprint X of an unknown excerpt and a fingerprint R referenced in the database. In practice, being R typically much longer than X, we define

$$d(X, R) = \underset{\tilde{R}}{\operatorname{argmin}}\, d(X, \tilde{R}) \quad ,$$

where $\tilde{R}$ is a fraction of R as long as X. How do we define $d(X, \tilde{R})$? The binary representation induces to use as metric the normalized Hamming distance, i.e. the number of different bits between X and $\tilde{R}$, divided by the total number of bits of X (or of $\tilde{R}$, since they are the same). That is:

$$d(X, \tilde{R}) = \frac{1}{N \cdot b} \sum_{n=0}^{N-1} \sum_{m=0}^{b-1} X_{n,m} \oplus \tilde{R}_{n,m} \quad ,$$

where $\oplus$ denotes the bitwise exclusive OR. One or more thresholds on the value of $d(X, \tilde{R})$ can be chosen, so to provide a hard indication of the similarity between the two excerpts. On the basis of our experiments, good-quality excerpts give always a correct match when $d(\cdot)$ is below 0.30.

However, distortions (such as recordings in harsh environments) tend to significantly increase the acceptable threshold value. Prior knowledge on the quality of the input may be of help in setting the threshold. Clear examples of this phenomenon are shown later in 5.3 and 6. Our proposal there to tune the threshold according to the devised task, tries actually to heuristically address the "metric learning" problem, very recently investigated in [31].

In general, d will always be greater than zero, even if X and $\tilde{R}$ come from the same non-distorted song, due to the non-synchronized framing. If we were to exhaustively search the database, we would adopt a minimum distance approach, taking as a match

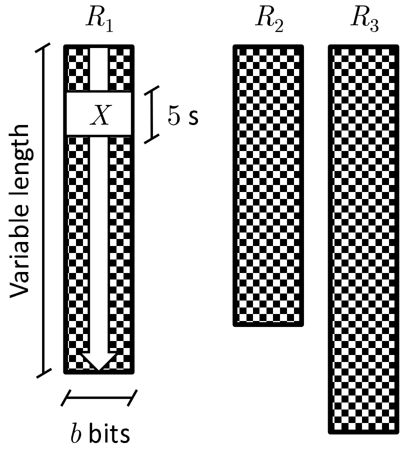


Figure 2.4: The exhaustive search, where X is the short, unknown audio chunk, while R_i are the fingerprints of the reference songs stored in the database.

for X a region $\hat{R}$ such that

$$\hat{R} = \operatorname*{argmin}_{\tilde{R}} d(X, \tilde{R}) \quad ,$$

where the search is extended to the whole database. If this value is above the reference threshold, we can safely conclude that no match was found.

2.4 Seeker

Given the fingerprint of an unknown excerpt, the seeker has the role of either finding the best match in the database, or declaring that the excerpt has no match. As a corollary application for the seeker, we observe that it can be a precious help also in updating the database. Consider the following scenario: a fairly large database as ours, of roughly 100 000 entries, is currently available. Every week or so, there's the need to enlarge it, in order to comprise recent acquisitions: the availability of a reliable seeker can successfully avoid adding duplicates [10], which could lead to troubles.

The naïve searching method requires an exhaustive comparison (i.e. linear scan) against the whole database, but this becomes rapidly unfeasible as the database grows,

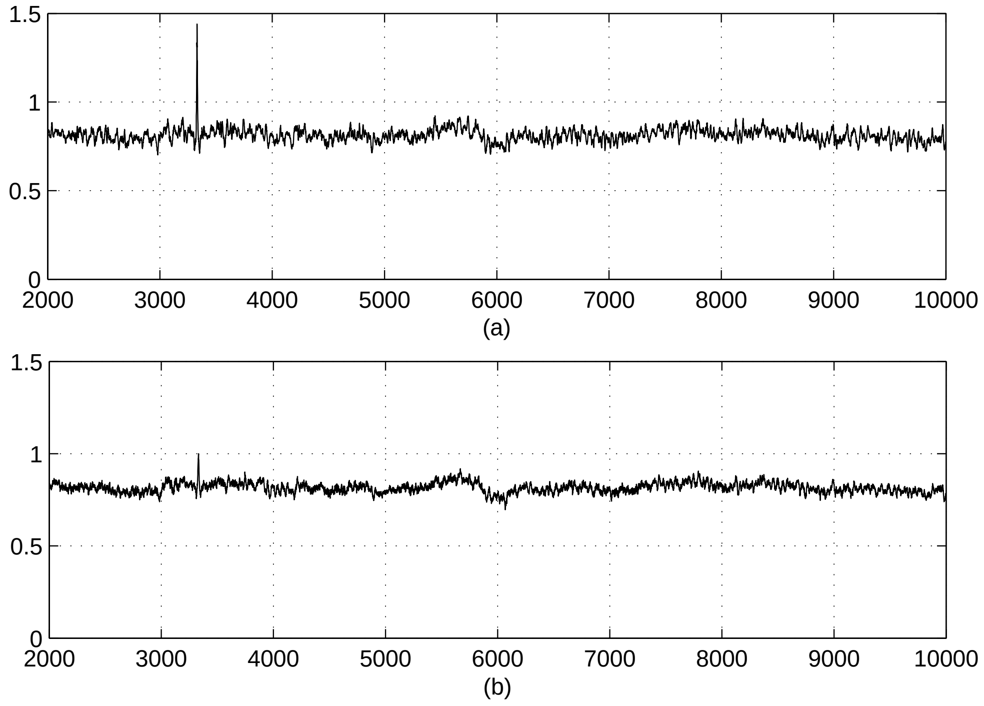


Figure 2.5: Cross-correlations between an unknown excerpt and the matching reference fingerprint. In case (a) the excerpt is undistorted and the synchronization point is easily detectable in correspondence of the high peak. On the other hand, in (b) the test excerpt is corrupted by white noise (SNR of 5 dB), which impairs the peak detection.

even with a careful implementation. Fig. 2.4 depicts this basic search approach. On the basis of our experience and for good-quality songs, some computation may be saved reducing by an integer factor $2 \div 10$ the alignments tried, relying on the large overlap in the STFT block. However, brute-force is highly inefficient.

A possible work-around, which scans anyway every database entry, is described in [32]. There, we compute the distance only on promising regions, whose locations are determined by the peaks of the cross-correlation between X and each possible R. The cross-correlation is performed along the time dimension and averaging along the frequency axis. An example is reported in Fig. 2.5. Nevertheless, also this approach requires too much time for being used in practice, even if it can be taken into consideration when careful investigations must be carried out. Indexing (3.1) is probably the best choice.

Chapter 3

Performance Improving

This chapter proposes a number of strategies for improving the reliability or the speed performance of the system, with special emphasis on issues connected to the broadcast monitoring application. Some of them are later discussed and evaluated in more detail.

3.1 Indexing

Following the rationale suggested in [7], we implemented our seeker on the basis of a simple, yet effective, method. A lookup-table (in fact, a *chain* hash table) acts as an index to quickly retrieve all the locations of a given pattern in the database. By "pattern" we denote any of the possible b-bit binary vectors, as those computed by the feature extractor, and by "location" we mean the song and the relevant offset, or an equivalent information.

When a match does exist, if we suppose that the unknown fingerprint and its matching reference both exhibit at least one feature vector exactly alike, then the index may be indeed effective. Querying it for the patterns computed from the unknown excerpt will give a number of promising alignments, thus dramatically reducing the number of comparisons needed, in the same fashion of the cross-correlation method. Moreover, look-ups are thousands of times faster (the exact ratio depending on the parameters used, compare chapter 7) than linear scan.

Ideally, the index would list all possible patterns, but this is not always feasible or advisable. Memory constraints may render impossible handling a 2^b-entry look-up

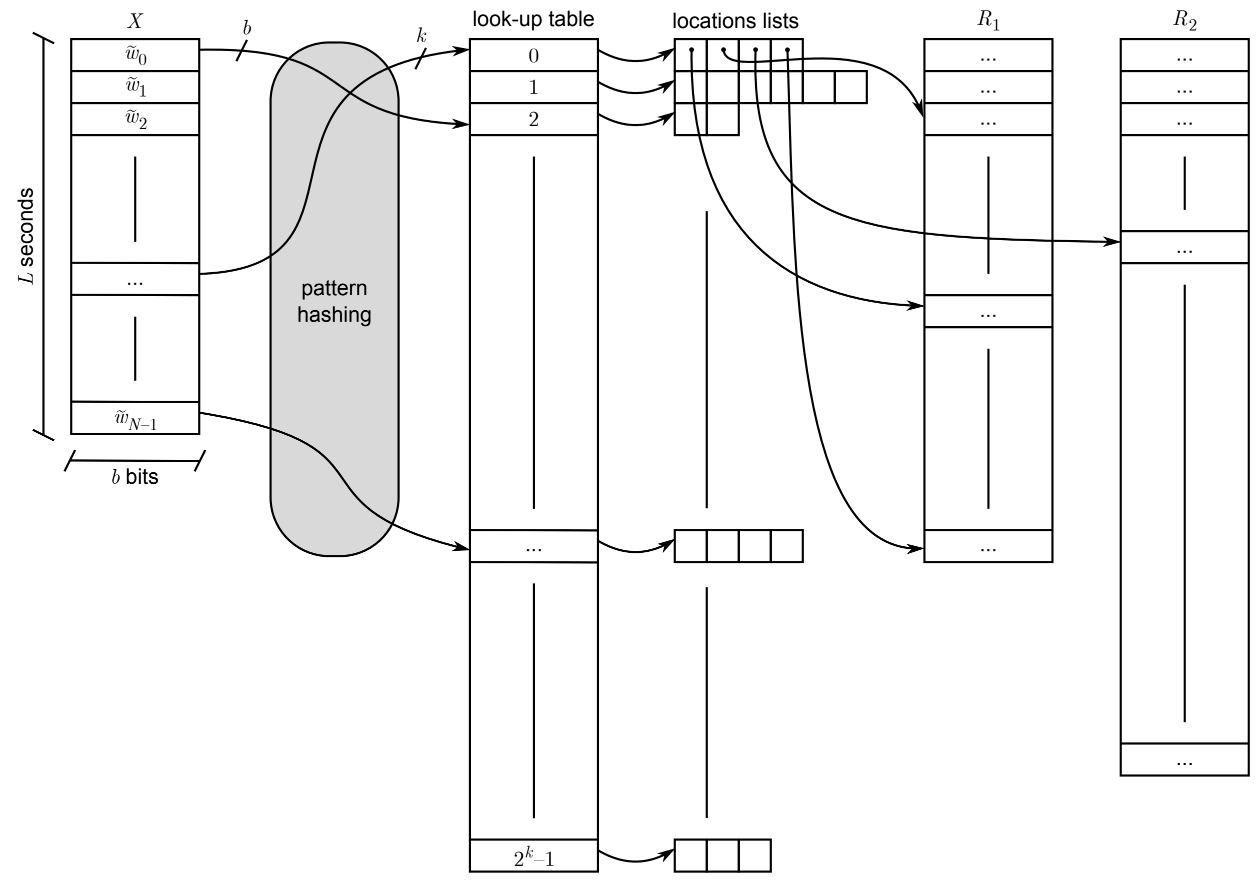


Figure 3.1: The indexing algorithm for fast retrieval of promising alignments between the unknown excerpt and the reference database. The gray block corresponds to the hashing algorithm of Fig. 3.2.

table (and permanent storage would be too slow), and excessively distributing the patterns in too many bins may lead to missing the correct synchronization points, as we verified. We thus propose to:

1. hash each b-bit pattern w in the database according to some convenient function $h(\cdot)$, giving a k-bit word $\tilde{w} = h(w)$, where $k \leq b$;

2. update entry $\tilde{w}$-th of the look-up table with the location (song and offset) of w, possibly chaining it to the existing ones (*separate chaining* collision resolution method, by means of a linked list).

This rationale can be readily understood by looking at Fig. 3.1.

The hash function carries out two roles. Firstly, it is needed to reduce the pattern space from 2^b to 2^k, so that standard amounts of memory (e.g. 4 GiB) become acceptable. Secondly, it greatly helps in equalizing performance. In this respect, it should be noted that the pattern distribution in the database is highly uneven, and in real-world application it is important to precisely foresee, and take into account, the time required for a search task.

Conversely, we point out here, and numerically justify in 7, the need of a large enough k: too small a value, actually, would lead to a small number of large buckets in which patterns are collected, so that the search time could still be unacceptably long. Let's now dive into the details of our implementation.

3.2 Pattern Hashing

We understand that the unstated hypothesis of having at least one exact match in the look-ups is quite strong. However, we propose the following pattern hashing method, which greatly relaxes this constraint and achieves very good results, as will be immediately cleared.

The overall hashing process is summarized in Fig. 3.2, where the dashed frame encloses the rationale just devised. In implementing the look-up table, the choice of a good hash function $h(\cdot)$ must not be underestimated. Apart from distributing the patterns as evenly as possible, it must also be very fast to compute. Two good algorithms with these properties may be found at [33] and [34]. Both were carefully evaluated and compared, giving similar results, and we chose the former.

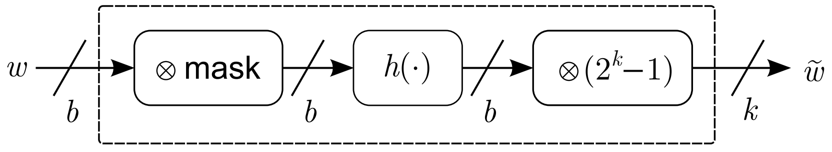


Figure 3.2: Scheme of the algorithm that transforms each single feature vectors from b into $k \leq b$ bits. The transformed vectors then populate the hash table used for efficient pattern look-ups.

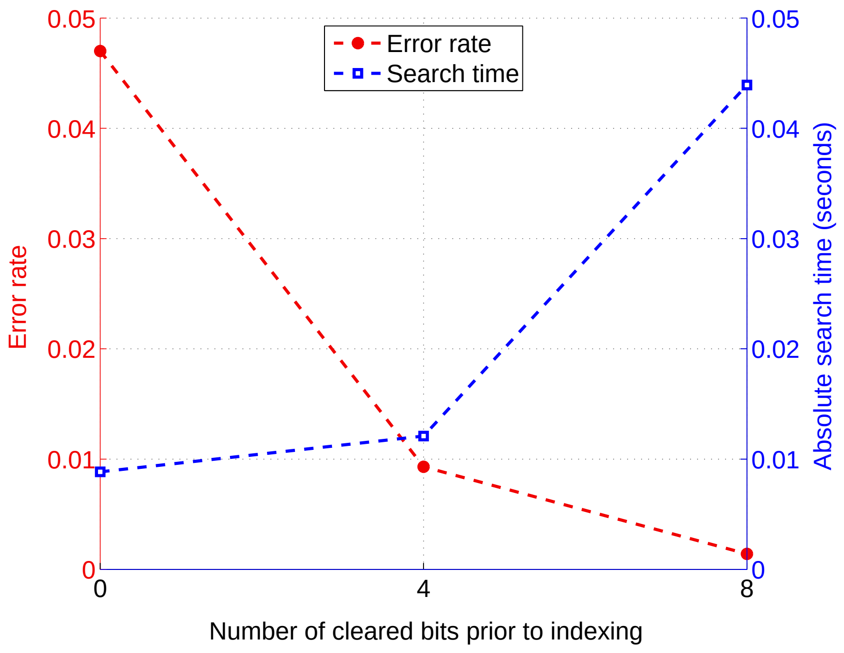


Figure 3.3: Impact of clearing bits in the computed patterns.

Since it realizes a 32-to-32-bit transformation, we kept a number k of the least significant bits of the hashed pattern as the search key. In other words, the key used for building and then querying the index is actually computed as $h(w) \otimes (2^k - 1)$, where h is the hash function, x the input pattern and $\otimes$ denotes the bitwise AND operator. Values from $k = 16$ through $k = 26$ at steps of 2 have been tested.

When the indexing technique is used, we also observed a significant improvement in the recognition error rate if some bits of the patterns are cleared prior to the hashing. This means replacing $w \to w \otimes \text{mask}$, where mask is a convenient b-bit value. The dimensionality reduction of the pattern space spreads the patterns in a lower number of bins in the hash function target space, so the increased search time must be properly taken into account.

Fig. 3.3 clearly highlights this dependence, for some special cases and $b = 32$. In particular, we represent mask $=$ 0x0FFFFFFF as an example of clearing 4 bits and mask $=$ 0x00FFFFFF for 8 bits, when the duration of the excerpt is 5 seconds.

On the basis of many trials, when $b = 32$, we experimentally found the best results by setting mask = 0x00FFFFFF, which leads to acceptable search times. We believe that this choice is rewarding because of the high variability and inherent weakness of the particular bits involved. In our implementation, they are representative of the lower frequencies. Since it is not possible to exhaustively try the whole range of possibilities, further investigation may be required at this stage. We also point out that this approach is related to the bit-toggling solution illustrated in [7].

3.3 Caching

The computer volatile memory is typically orders of magnitude faster than ordinary storage (i.e. the hard disk). To better drive implementation decisions, a variable-size cache was thus implemented in RAM, with respect to the fingerprint archive. Although there may exist more refined replacement policies, we used a FIFO (First In, First Out) ring buffer. The rationale is represented in Fig. 3.4.

Whenever part of an entry (reference fingerprint) is needed, the database entry is loaded in its entirety in the first available bin, for future reference. In many cases, as when tracking an audio broadcast, it is reasonable to assume that many excerpts from the same song will be present. When there are no free bins, then the oldest one is replaced. The size of the cache has been varied from 0% to 100% of the database size, at steps of 10%.

3.4 Runs Filtering

In Fig. 3.5a we report a sample, short fingerprint. Because of the high temporal correlation between the overlapping segments of the input, the bit matrix is mainly constituted by runs of 0s and 1s. We note anyway some “undecided” regions, where 0s and 1s tend to alternate. The negative impact on the overall system, in the long run, is twofold: not only does it lead to an increased distance between excerpts and possibly matching references, but also limits the robustness in case, for example, of pitch-shifted audio (5).

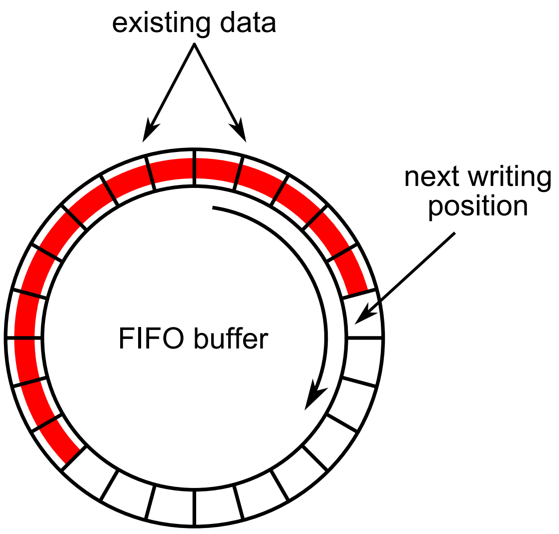
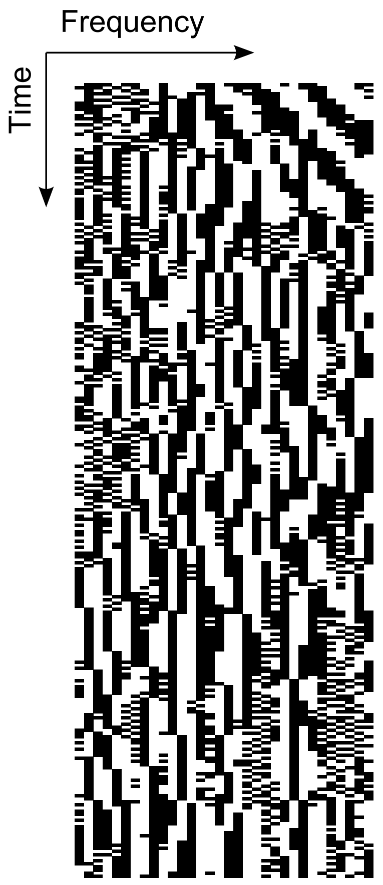
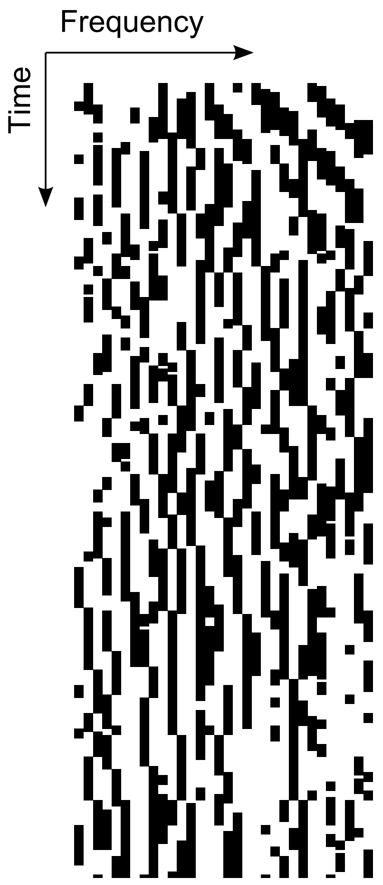


Figure 3.4: The ring buffer used for evaluating the impact of caching the fingerprint database in the computer memory.

These thoughts suggested us to add little processing to both the unknown and the reference fingerprints, in order to keep only runs of a minimum configurable length δ, as depicted in Fig. 3.5b for $\delta = 3$. After this extra stage, the operation follows as usual with the eventual look-up of promising locations and then distance metric computation. We will show that this precaution (i.e. the runs filtering) allows, to a certain degree, to obtain an average smaller distance in case of match. At the same time, it does not reduce too much the distance with respect to non-matching entries. Finally, we point out that it does not cause the database to be rebuilt and can be easily done on the fly at the time of loading.



(a)



(b)

Figure 3.5: Sample fingerprints (a) before and (b) after filtering with $\delta = 3$. 256 32-bit patterns, each representative of a time frame, are vertically stacked, thus spanning the frequency dimension horizontally. The duration of the chunk is about 3 s. δ denotes the magnitude of the runs filtering described in 3.4.

3.5 Database Partitioning

It is intuitively clear that the search time is inherently related to the size of the database, as we will see in detail in section 7.3. Therefore, finding a way of partitioning the database could be a practical way to reduce search times and complexity, beyond increasing scalability.

A first approach could be that of determining the genre of a song, e.g. following [35], which is also a good review of some relevant works, or [36]. Unfortunately, there is no expressed agreement on the definition of "genre" and if it were, it would change over time, along with the common perception of music. Moreover, its evaluation is usually computationally expensive. We thus propose to classify songs on the basis of their *tempo* (or beats per minute, bpm), which in many cases can be objectively measured with a good accuracy. An additional bin may be considered for all those songs whose tempo is unclear, irregular, or anyway needs human supervision for being estimated.

Tempo can be described as the rhythm pattern of a piece of music and characterize especially western music. If the beats per second are constant along a song, then we can compute them from each song and partition the database into sections, each of them including a specific beat range, possibly non-overlapping.

On the other hand, song beat may differ in different parts of a song, e.g. between verse and chorus. In that case, the database sections may overlap and the same song may be assigned to more than one group. This leads to a loss of efficiency, but does not impair the correctness, as long as the beat estimation is correct.

Many approaches for beat estimation have been proposed, and even a contest was held for finding good algorithms [37]. We also propose the following, which is straightforward and gave good results on many songs taken from a commercial radio broadcast. Assuming that the song exhibits energy peaks on beats, we compute the temporal cross-correlation between its PCM samples and a number of impulse trains of convenient period. We then take as beat reference the train with the highest correlation with the song. In order to reduce the complexity, the song may be also downsampled, and possibly segmented in order to take variations into account.

The periods of the impulse trains may be chosen among a wide range or it may be limited by prior information. The impulse is bell-shaped, with a duration of 100 ms. For efficiency, the impulse trains should be generated just once and then retrieved when needed.

When the song presents a distinct rhythm pattern given by a drum or a similar instrument – as is the case of many rock and pop songs –, our approach yields typically little or no errors, at most within a few bpm of what a human experimenter would feel right. However, it is unable to produce meaningful results when no rhythm patterns is clearly perceptible (e.g. melodic and classical music).

Given the size of the database, the number of segments and the computational burden required by the beat estimation, it will be possible to evaluate the trade-off between the effort needed to estimate the tempo and the advantage derived from the database partitioning. Although the database can be partitioned off-line, we stress the need for a fast beat estimation algorithm, in order to actually have an overall processing gain, since the beats per minute must be estimated on the input chunks as well. This constraint may be lifted only if we can rely on prior tempo information. We currently do not have an implementation efficient enough to prove useful, but according to [37], the beat estimation could take as low as 0.02 times the excerpt length.

It is worth noting that the beat estimation may impose some constraints on the minimum excerpt length. However, encouraged by our tests, we believe that a value of 5 seconds, adopted and justified in the following sections, should be enough for most cases.

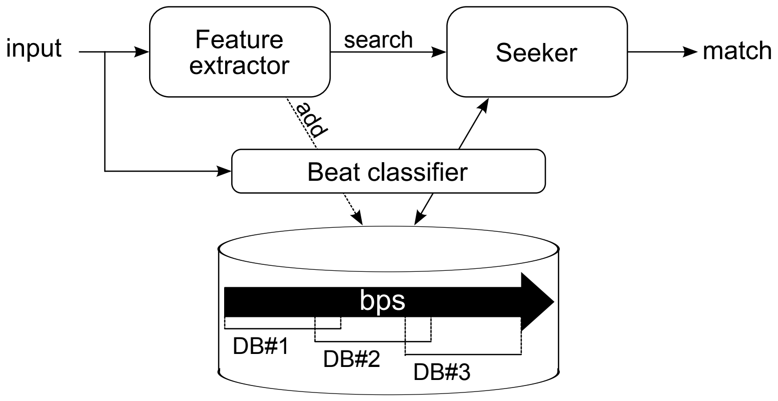


Figure 3.6: Database partitioning through beat classification, with possibly overlapping sections.

3.6 Speech and Silence Detection

In the view of broadcast monitoring and tracking, a mechanism of speech detection may prove really profitable. On the other hand, we are aware there are notable applications which would not benefit from it, like the addressing of copyright infringement issues in file-sharing networks. Nevertheless, we believe it is worth discussing.

First of all, speech detection can be implemented by pre-processing the audio prior to the fingerprinting, thus allowing to discard non-musical sections and achieve a better efficiency. Secondly, it could also occur at the output of system, trying to analyze what has not been recognized, along with other blocks, such as a recognizer that operates on the whole database instead of on just the group determined by the estimated beat.

As in other relevant works [38], our approach to speech/music separation exploits a well-known feature, the zero-crossing rate (ZCR). It can be computed directly on temporal frames and gives a rough indication of the frequency range in which most of the energy lies. Since the human speech is bounded up to a few hundreds hertz and music usually involves also much higher frequencies, the ZCR allows to discriminate the two cases.

Of course, since this method has no real understanding of the content of the eventual speech detected, it is not able to tell apart an actual speech from a rap excerpt or a sung part with little or no music. It can also be tricked by loud background music.

By considering the frequencies of human speech, we enhanced this method by computing the fractional residual energy outside this band. Combining the ZCR and the residual energy indicator, we have very good estimation results, tested on many real radio recordings. The exact parameters can be tuned according to the particular context, so to allow for a flexible tolerance.

Similarly to the beat recognition case, the speech detection algorithm is implemented off-line and is not part of the audio recognizing system. Moreover, it constrains the excerpt length, though a few seconds are enough for a reliable speech detection.

We conclude this section by noting that it is also important to discriminate silent inputs, as they easily match randomly against the database and lead to false positives. We do take into account this case by making sure that the total energy of the processed excerpt is above a threshold, which was heuristically tuned. In particular, we evaluate whether more than 40% of the waveform samples (normalized in the range $[-1, 1]$) have an absolute value greater than 0.02.

3.7 Post-processing of Results

When analysis is made to carefully track the content of an audio stream, as in the broadcast monitoring application, the reliability of the audio recognizer can be greatly increased by taking into account the analysis history. That is, those excerpts difficult

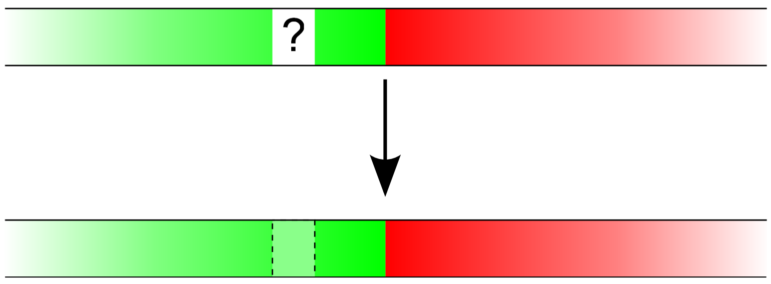


Figure 3.7: A possible inference thanks to the recognizer history. We depict here a sequence of two songs, colored respectively in green and red. A fraction of the green song was not recognized, but can be safely inferred.

to recognize can be handled by looking at the matches previously found. Even better, in most situations it is possible to take advantage of future matches as well.

We point out that the duration of the excerpts in the stream to track must be carefully chosen. Since each excerpt will provide one and only one match, it cannot be too long, otherwise we will easily mix different songs. On the other hand, too short an excerpt won't provide enough information for a reliable matching. On the basis of our experiments and previous works, we suggest excerpts of 3.3 to 5 seconds each. Fig. 4.2), commented in section 4.2, reports

Once the stream, or a sufficiently large part of it, has been processed, our algorithm verifies whether some excerpts could not be recognized. Unknown excerpts surrounded by long enough segments of a same song, are assigned to it as well (an example is depicted in Fig 3.7). The particular values to use highly depend on the context and cannot be determined once for all. By this strategy, however, we were able to correctly and precisely track ambient recordings made in a very noisy discotheque, under the hypotheses of a minimum duration for each song.

As a corollary of this application, in the case of broadcast monitoring it would also be natural – instead of blindly looking up promising alignments all over the database – to first check whether the current chunk is simply a further chunk of a same song. The saving in computation would be particularly significant for very large databases. However, also for easier comparison with other fingerprinting systems, we do not consider here this enhancement.

Chapter 4

Implementation and Preliminary Investigations

We first present our testbed and then, through the remaining chapters, we discuss the outcomes of ample experiments on SAF, the fingerprint algorithm detailed in 2.1. For reasons of efficiency, our implementation is written entirely in (portable) C++. Tests were run on 32-bit middle-end desktop PCs, with 3.00 GHz CPUs, and 3.5 GiB of available RAM. If not stated otherwise, tests were carried out on excerpts whose duration is 5 s, corresponding to $N = 399$ b-bit vectors with the parameters given in 2.1, where $b = 32$ and the band division is 12-TET.

4.1 Testbed

With the proposed parameters, we extract about 5168 4-byte feature vectors per playing minute. Since the average song length in our 100 000-song database[1] in MM:SS is 4:18, we have a total storage of 8460 MiB, or 87 KiB per track, with $2.22 \cdot 10^9$ feature vectors stored.

We highlighted in 3 the strong need for both indexing and operating in the RAM memory. Even leaving out the size of the index, however, the size of the database alone rendered this task unfeasible to the machines available to us. We thus devised and operated the following solution:

[1]We are very grateful to Knowmark s.r.l. for their support at this stage.

1. we evenly split the database into 10 sub-databases, each assigned to an independent instance of the recognizer. To be precise, we used 10 sub-databases of 10 000 entries each, but on average, this is the same as evenly dividing the total database size, which linearly depends on the duration of the songs;

2. each of the 10 independent recognizers is then fed with the same audio excerpt, randomly extracted from those existing in the database[2];

3. we finally merge the results, taking into account the smallest distance among the 10 distances independently computed across the 10 sub-databases. Decisions on the acceptability of such a distance follows as previously described.

On the positive side, we highlight that the instances of the recognizer may safely be separate processes, since each will deal with just a tenth (or a fraction, in any case) of the whole database, independently. The same rationale can also be easily customized and tailored to one's needs and resources (e.g. some machines are faster than others).

If we stick to an on-line recognition task, the main drawback is the need for either 10 rather modest machines (which is our case and it is cheap), or for one or more powerful ones (e.g., a 16 GB RAM machine would be able to handle approx. 60 000 to 70, 000 fingerprints, if we include the indexing structure). Moreover, Loading a sub-database from disk induces some temporal overhead, but this is acceptable if the audio stream is long enough (see also table 7.1). The good gain of processing time over real time must be also considered (compare section 7). Post-processing, on the other hand, is not an issue, since it can be done on the fly (given suitable inter-process mechanisms) and requires practically no time.

Nevertheless, for the sake of easiness of implementation, we performed off-line the merging step, which is indeed realistic for broadcast monitoring. We can better follow the steps of the recognition process with the aid of Fig. 4.1, which is simplified to just 3 sub-databases.

[2]The issue of false positives has been addressed in [7], and in our experience we honestly never found none.

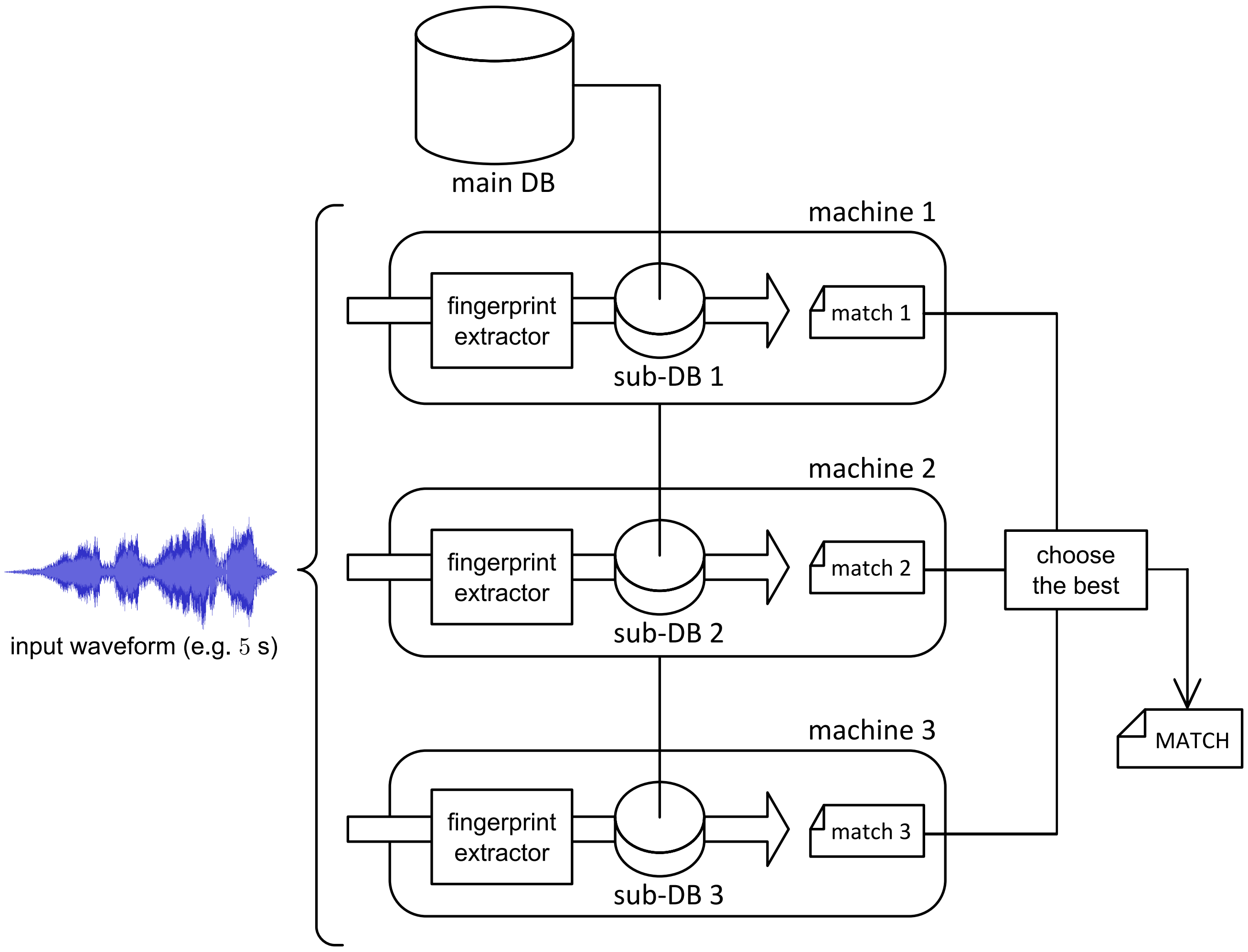


Figure 4.1: Overview of the off-line recognition process, as per the strategy described in section 4.1, in order to overcome memory limitations. For clarity, only 3 sub-databases are depicted here.

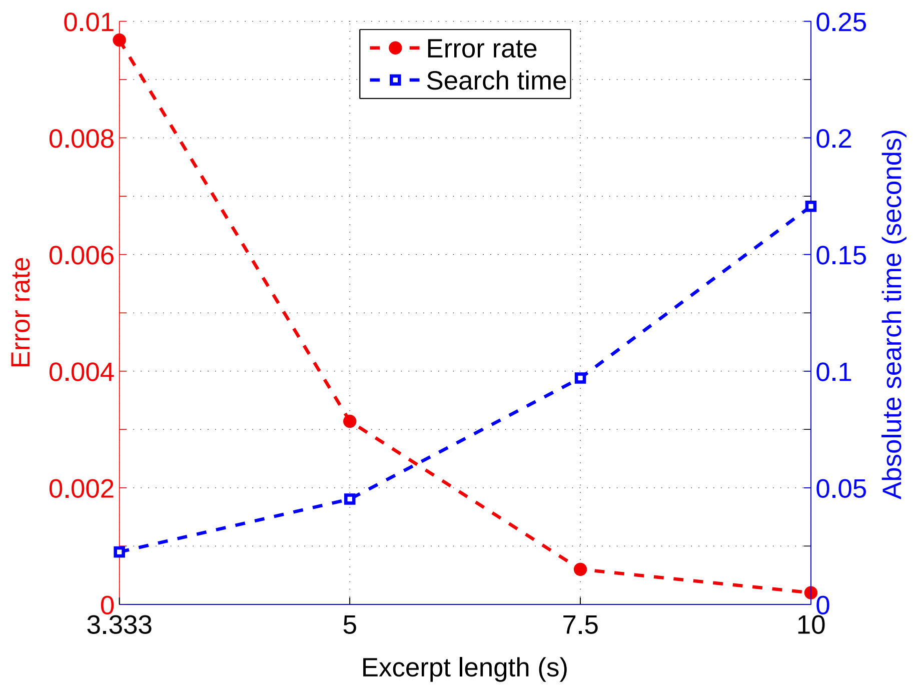


Figure 4.2: Impact of the excerpt length on the error rate and on the search time ($k = 24$).

We also stress that in the context of broadcast monitoring and all others which do not require an immediate answer to an audio query, part of the audio stream can be processed multiple times, sequentially, by a same machine, which iterates through the sub-databases.

We finally observe that the blind database partitioning just discussed is instrumental to allow a certain degree of scalability, and is logically orthogonal to the tempo-based partition suggested in 3.5. The two approaches can thus be profitably combined, or even introduced at different times into an existing architecture.

4.2 Effectiveness Investigations

We speak of "success" when the system gives as match the correct reference. The "error rate" is thus computed as the one's complement of the success rate, obtained by averaging over a very large number of independent trials, in a Monte Carlo fashion.

While the error rate figure is given by comparing against the whole 100 000-entry database, we point out that all the timing results presented in later sections are related to the single sub-databases of size 10 000: the sub-databases are in fact approximately homogeneous and browsed concurrently, and we can safely neglect the post-processing time.

First of all, we verified through brute-force search that the fingerprint algorithm under analysis is thoroughly effective on undistorted excerpts. In this case, the error rate is exactly 0, even for very short excerpts of just 3.3 seconds.

According to 3.1 and 3.2, indexing and pattern masking are then introduced. We report in Fig. 4.2 the trade-off between accuracy of the match and search time required per each excerpt, with respect to its duration. While we observe a significant drop-off in the error rate switching from 3.3- to 5-second excerpts, the improvement is less noticeable if we further increase its duration. Thus, we chose 5 seconds as the reference duration for excerpts.

Chapter 5

Addressing Pitch Distortions

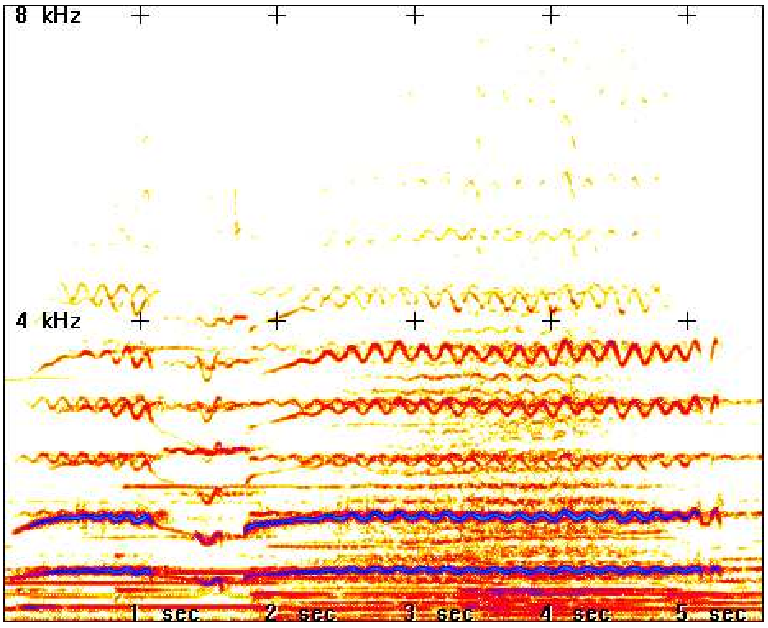
The SAF algorithm proves highly reliable with respect to a large number of signal degradations [7], as we also experimented. However, due to its inherent characteristics, this holds as long as the frequency and time texture of the song are not significantly altered. As reported in [26] and [39], even a moderate time- or frequency-scaling voids the effectiveness of the system.

By "pitch distortion" we denote a lowering or a raising in pitch of an audio signal, without affecting the tempo (time scale), i.e. the time length of the excerpt is unchanged. Note that this is first of all a frequency shifting, but it also involves a dilation in the frequency domain, in order to preserve the musical relationship of the harmonics [40]. This phenomenon can be clearly seen from the spectrograms[1] of Fig. 5.1.

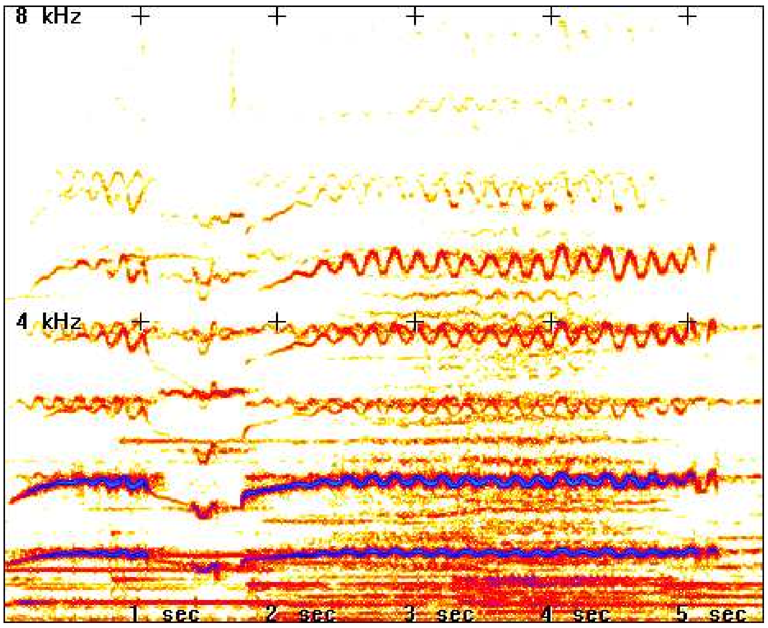
As with any other distortion, a trivial approach is trying to restore (e.g. "de-pitch") an excerpt before taking its fingerprint. If correctly carried on, this is always successful, but most probably unfeasible, as seen later on.

As hinted in 2.2, we could also combat these weaknesses by re-encoding each fingerprint, e.g. as a sequence of symbols, untying this new fingerprint from the original song structure and allowing for a more tolerant search. An example was cited in the introduction [5]. Both time- and frequency-scale distortions could be successfully ad-

[1]Picture adapted from the output of the freeware program *Spectrogram 4.1.2*, by R.S. Horne. The STFT used has a frequency resolution of about 21.5 Hz and a time step of 12 ms. Only the left channel is considered.



(a)



(b)

Figure 5.1: These spectrograms represent a few seconds of Anne Browne's *Summertime*, which we have available in MP3 format. The time dimension is horizontal, while the vertical dimension is the frequency. While in (a) the song is undistorted, in (b) we have the same excerpt raised in pitch by 5 semitones (same duration). Both frequency translation (towards higher frequencies) and dilation can be clearly seen.

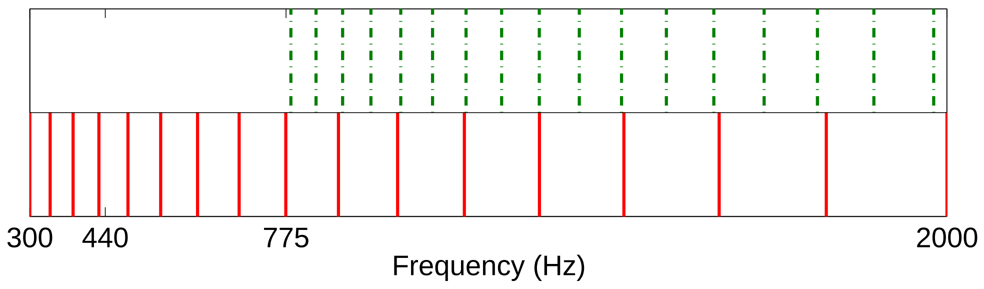
dressed, with the great benefit of building over the existing database, and with no need to switch to a different fingerprinting algorithm.

However, this approach can be overkill for most of the applications. Then, it would be interesting to exploit the existing data without considerable modifications. We thus propose a simpler method, which can successfully address very high pitch distortions, giving at the same time very good results.

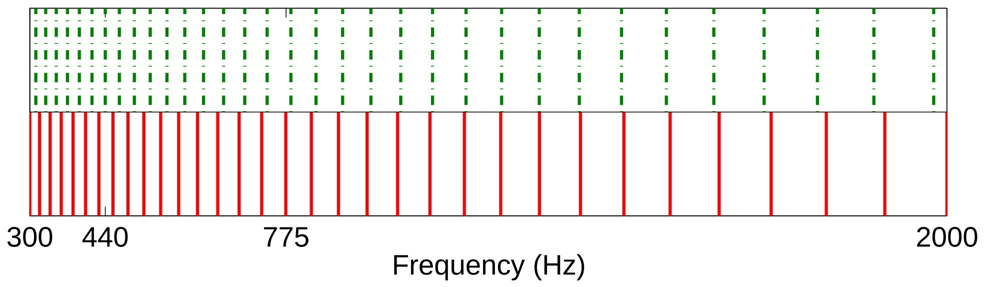
The band division introduced in 2.1 exploits quite an often used tuning system, at least as far as the western world is concerned: equal temperament, or 12-TET. Leaving out the historical reasons which drove his adoption, we only point out that it is built from a basic tone in the neighborhood of 440 Hz (called A4 or La4). In our case, this exact value was used. Each of the 12 available notes is then obtained by multiplying (or dividing) the frequency of the adjacent tone by a factor of $2^{\frac{1}{12}}$ (as seen in 2.1). This interval is called a “semitone”.

The iterated procedure leads to the musical scale (C, C$\sharp$, D, ...B) or (Do, Do$\sharp$, Re, ...Ti), probably known to the reader, where C$\sharp$ and D$\flat$ (and so on for analogue couples) have actually the same frequency. The interval between a tone and its doubled-frequency counterpart is called “octave”, and is such that we perceive two sounds an octave apart as having the same pitch. In conclusion, 12-TET is a tuning system where we can perceive as much as 12 unique pitches. Note that 12-TET is not related to the Bark scale.

For the sake of comparison, we built an additional database, so to evaluate the response of the system with respect to two different band quantization choices. The most straightforward approach is to log-equally divide a meaningful bandwidth, justified by the fact that the human ear has an approximate logarithmic response. The base used for the logarithm is 10 and the performed subdivision will be denoted by EQL. We report in Fig. 5.2 a comparison between the two schemes, 12-TET and EQL, for the cases $b = 16$ and $b = 32$. The former case leads to a lower bound of $l = 784$ Hz, whereas the second to $l = 311$ Hz. This last is very similar to the $300 \div 2000$ Hz bandwidth suggested in [7].



(a)



(b)

Figure 5.2: Comparison between 12-TET (dashed) and EQL sub-bands, in linear scale. (a) $b = 16$ (b) $b = 32$

5.1 An Alternative Search Strategy

Thanks to the 12-TET band division, it is possible to conceive an alternative search strategy. Taking up the picture of the unknown fingerprint slid along the time dimension of the database, we propose to extend the search to the frequency dimension, thus shifting by one or more bits the fingerprint of the excerpt prior to the comparison. The idea is depicted in Fig. 5.3.

Depending on the expected magnitude and direction of the distortion, the search should be appropriately limited, in order to save time. Since a frequency shifting is involved in pitching, we will show that this method allows to greatly improve the robustness of the recognizer, at least for not excessively strong distortions.

The major drawback is the impossibility to use the indexing technique described in 3.1 with no modifications. Therefore, the time required by the tests presented here

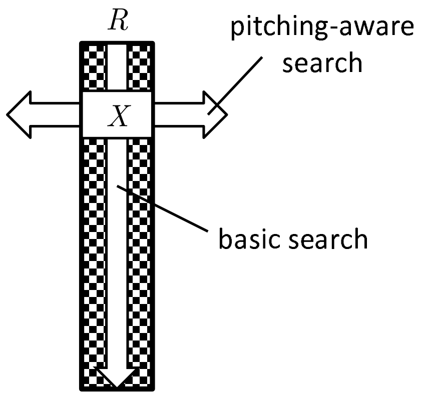


Figure 5.3: The proposed search for combating pitching, where X is the short, unknown audio chunk, while R is the fingerprint of a reference song.

is of the same order of magnitude of a brute-force approach (see also 7.3), with the further multiplying factor of a frequency-wise shifting. At the moment, with realistic database sizes, this can be acceptable only when a deep automatic analysis is required or there are no strict time constraints. This can well be for certain off-line monitoring tasks, as the processing of a relatively-short daily stream, or the analysis of a weekly event.

Although the proposed 12-TET band division is based on a western convention, we point out that the described search strategy holds regardless of the specific kind of music being processed, save the limitations discussed in the following section (i.e. the tailoring of the band division for combating a conceivable distortion granularity).

Recalling the “de-pitch” approach, it would naturally fall in the pre-processing stage, or – more efficiently – just after the STFT computation, whose large time-overlap parameter will greatly benefit the accuracy of the pitching operation [40]. On the positive side, this approach always compares full b-bit words, but in turn it requires a complex, non-negligible computational effort, and almost always some trial-and-error (since it looks improbable to know in advance the intensity and direction of the distortion, even if just approximate). Conversely, performing a bit-shift operation is simple and almost instantaneous, although the number of actually compared bits is linearly reduced by the magnitude of the bit shift.

Another more subtle issue is due to the choice of the frequency domain quantization in the fingerprint algorithm. At the fingerprint level, it actually maps a bit shift to a fixed frequency shift, so it accounts for at least two major drawbacks in the fingerprint-shift approach. Firstly, it allows to correctly handle pitch distortions only if their magnitude falls about evenly on quantization boundaries and secondly, it poses a limit on the maximum tolerable distortion. In other words, the bit shift cannot be too large, in order to retain a significant fraction of the available information.

We conclude this section by noting that, if both the time and frequency scales are linearly stretched up or down by a few percents, an effective workaround has been presented in [26]. Its main advantage – in contrast to our approach – is a very good handling of these distortions with a fine granularity, at least within its range of use.

Unfortunately, it has also many drawbacks. In the first place, it requires substantial modification of the algorithm, rendering any existing databases completely useless, which is unacceptable in many cases, e.g. because of the unavailability of the source songs. Secondly, the running time of the new algorithm is appreciably longer, since it requires – in addition – as much as two Fourier transforms (implementing auto-correlation), a low-pass filter and a downsampling (this could not be an issue if the constraints on the search time are not particularly tight).

The authors locate the need of their method especially in the recognizing of radio broadcasts, where time constraints may indeed lead to speeding up songs. In these cases, however, the speed up is limited and involves only the time scale, a situation which is well handled by the large-overlap framing in the existing algorithm. We conducted many experiments in radio broadcasts with results entirely comparable to the non-distorted case (i.e. error-free). Nevertheless, the approach could prove useful in certain contexts.

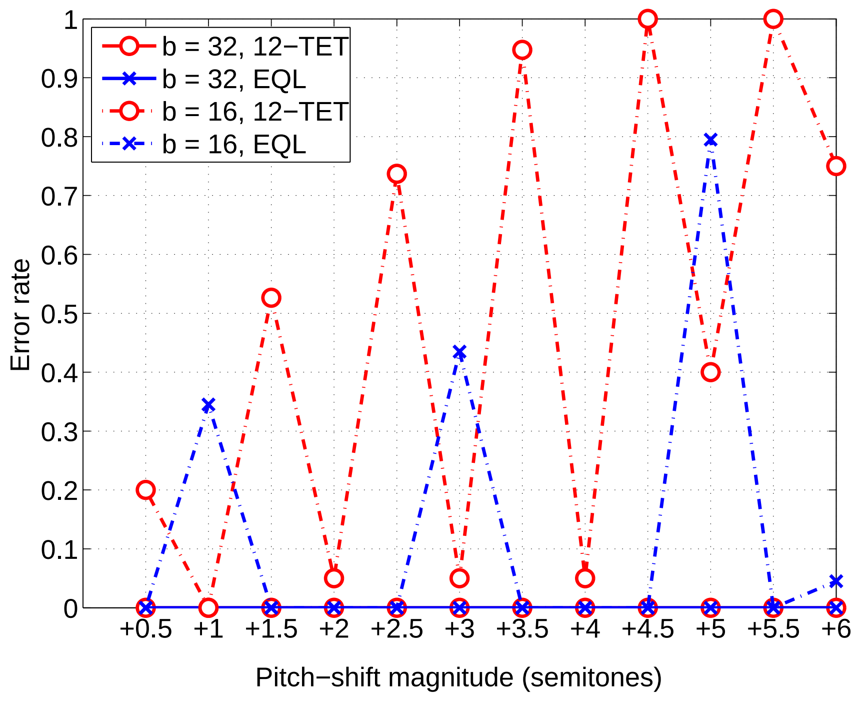


Figure 5.4: Error rate as a function of the pitch-shift and for different parameters.

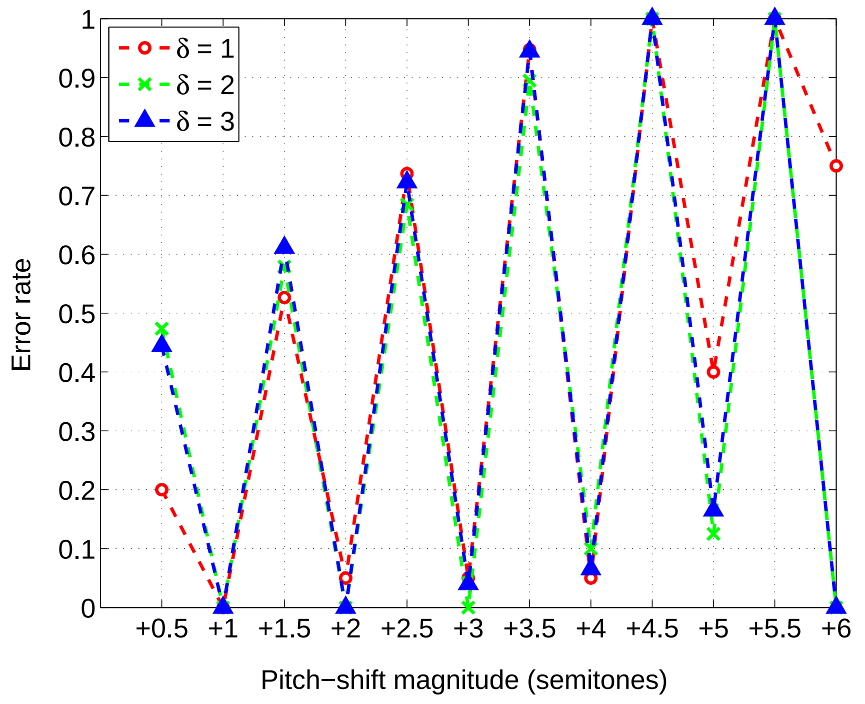


Figure 5.5: Error rate as a function of the pitch-shift and δ ($b = 16$).

5.2 Error Rate

A number of random songs was selected among those existing in the database and manually pitched using the open source tool *Audacity* [41]. The magnitude of the distortion varies from +0.5 to +6 semitones, with a step of 0.5 semitones, involving an approximate percent frequency displacement summarized by table 5.1. On the basis of our previous discussion, a 12-semitone pitch shift, i.e. an octave, would correspond to a 100% frequency shift. For reasons of symmetry emerged in a previous work [39], we do not consider here negative pitch shifts.

Let's first focus on the fingerprint-shift search strategy. As we can see in Fig. 5.4, a careful choice of the sub-bands may be essential, also depending on the particular value of b used. For $b = 16$ and EQL, the behavior alternates between an excellent success rate (around distortions of an even number of semitones) and a very poor one. Similarly, its 12-TET counterpart is heavily troubled by half-semitone distortions. When $b = 32$, the proposed search strategy is instead thoroughly effective, regardless of the particular band division. The same can be observed in the following Fig. 5.7

On the contrary, if we had a viable "de-pitch" approach, it would lead to perfect results in terms of success rate. This would be comparable to the undistorted case, with the only drawback of a slightly increased distance. According to our tests, this increase would be at most in the neighborhood of an additional $5 \div 6\%$.

If the exact pitch shift is not known or guessed, indeed, we are exactly in the same case of having a pitch-shifted excerpt and employing the basic search strategy. That is, without a very good estimation of the pitch shift involved, or without enough (and computationally expensive) trials and errors, the "de-pitch" solution is quite useless. A concerted strategy could be thought of, but at a great expense in terms of search time. From now on, only the bit-shift approach will be brought into discussion.

When $b = 16$, it is possible to further reduce the error rate by employing the strategy described in 3.4 and filtering out the shortest runs in the fingerprints, prior to the distance computation. For clarity, we report here the results up to a value of

+0.5	+1	+1.5	+2
+2.93%	+5.95%	+9.05%	+12.24%
+2.5	+3	+3.5	+4
+15.54%	+18.92%	+22.41%	+25.99%
+4.5	+5	+4.5	+6
+29.68%	+33.48%	+37.40%	+41.42%

Table 5.1: Pitching/frequency-displacement approximate mapping.

$\delta = 3$. As we can see from Fig. 5.5, $\delta = 2$ appears a well-balanced choice, with respect to the error rate metric.

5.3 Computed Distance

Since we use a threshold mechanism for telling correct matches apart, a most important metric to investigate is the average distance computed in case of success or error. Indeed, both can help in fixing (a) reliability threshold(s) and are a soft indication of the overall robustness of the recognizer. In Figs. 5.6 for $b = 16$ and 5.7 for $b = 32$, we observe that the distance in case of success tends to increase with the magnitude of the distortion, with more or less a constant penalty when half-semitone distortions occur. The lower values for the high pitch shift of +6 semitones may be due to the particular distribution of most of the erroneous bits when pitch shifting is involved.

Introducing $\delta > 1$ does help in lowering the evaluated distance with respect to the correct match. This is already clear in the presented figures, so we did not report the dependence of the metric from δ itself, which is constituted by slowly-decreasing, quasi-parallel straight lines, as δ grows.

We point out that $\delta = 2$ can be regarded as very appropriate, since it reduces both the distance with respect to the match and the overall error rate. The only, minor drawback is a slightly reduced distance in case of error (should they occur): this could potentially lead the computed distance close to the threshold used for discriminating whether or not we have a match.

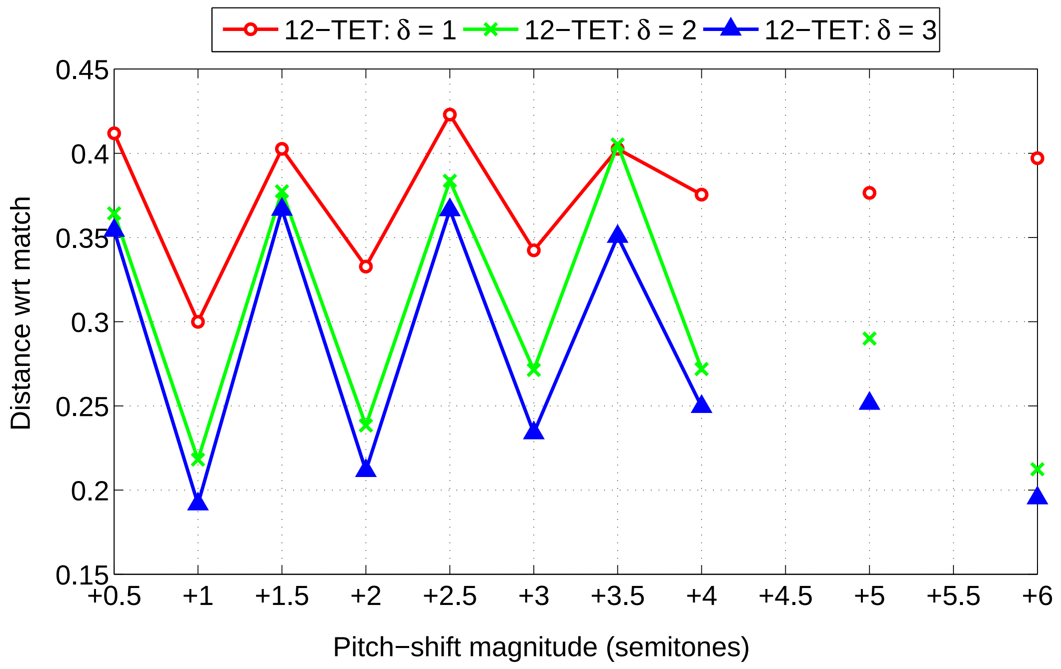


Figure 5.6: Given $b = 16$, normalized Hamming distance against the correct match as a function of both the pitch-shift magnitude and δ. The distances for $+4.5$ and $+5.5$ are missing because no success was recorded.

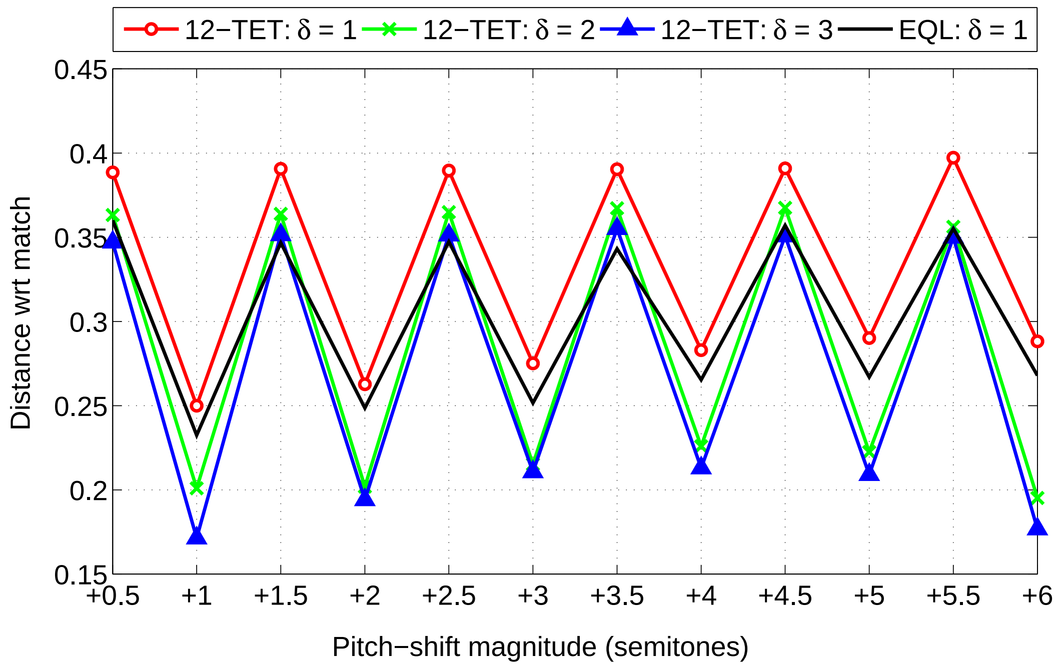


Figure 5.7: Given $b = 32$, normalized Hamming distance against the correct match as a function of both the pitch-shift magnitude and δ.

Chapter 6

Performance Under AWGN

Additive white gaussian noise (AWGN), or thermal noise, is very common and worth investigating as far as the performance of the system is concerned. In a previous work [32], we found that $b = 16$ could in this case lead to better results. However, the database used was orders of magnitude smaller than the actual one, and the cross-correlation search method described in 2.4 was employed.

Carrying on similar experiments on a much wider database and cautiously relying on the brute-force search, we obtained the curves in Fig. 6.1, where we represent the average error rate with respect to the Signal-to-Noise Ratio (SNR) of the excerpt. The excerpts were randomly chosen from the whole database and then conveniently corrupted on the fly.

The choice $b = 16$ may actually pay off in certain circumstances, but it emerges that $b = 32$ is actually better in a wider range of cases. Moreover, as we will soon see in the next section, $b = 32$ is almost a mandatory choice if we aim at reasonable search times.

Nevertheless, it is interesting to note that $b = 16$ would lead to a noticeably smaller distance between the unknown excerpt and its match, with respect to the case $b = 32$. In addition, should matching errors occur, they will show up with a higher distance than the case $b = 32$, as can be seen in Fig. 6.2. On the basis of these considerations, we believe the choice $b = 16$ could be acceptable and rewarding, if search times are not the primary concern.

This last figure also suggests the possibility of increasing the tolerable threshold for declaring a match, in order to adopt a more aggressive strategy and trying to avoid false negatives. We believe that a more rewarding choice is instead adopting a two-threshold mechanism, associating to them different reliability indicators. For example, a distance metric below a threshold $T_1 = 0.30$ can be taken as a safe match, while a value between T_1 and $T_2 = 0.35$ should be considered less reliable. From this point of view, the post-processing strategy described in 3.7 can be really of help, as well as the tunable threshold hinted in 2.3.

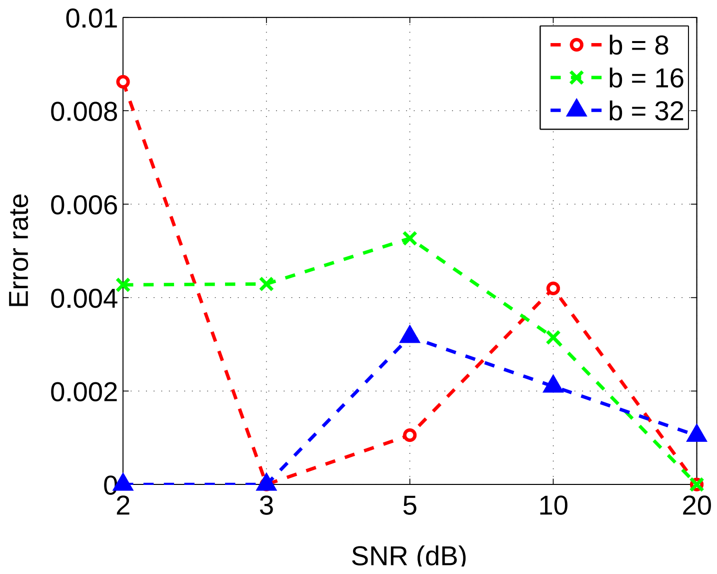


Figure 6.1: Error rate for three different choices of b.

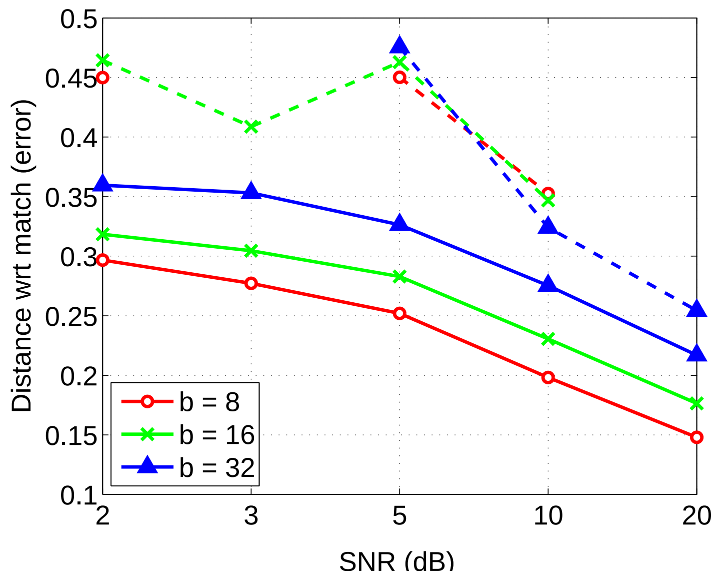


Figure 6.2: In the continuous lines, we depict the distance with respect to match for three different choices of b. Their respective counterpart in case of error, when errors occur, is represented by the dashed lines.

Chapter 7

Timing Performance

The results presented in the following sections, valid in accordance to the specifications in 4, can be of great help when it comes to design the system. In particular, our considerations investigate its scalability and definitely show the need of operating as much as possible in RAM and having $b = 32$.

7.1 Indexing

First of all, let's focus on Fig. 7.1, where we report both the absolute search time and the number of comparisons effectively made (i.e. the number of alignment locations tried in the database). Both non-cached and fully-cached indexing are reported, while we discuss intermediate values later in 7.2. The most striking outcome is that a sub-database fully loaded on memory (i.e. a 100% cache) leads to a search time which is always well beyond an order of magnitude faster than the approach without cache, ranging from 370 s to 2.1 s for $k = 16$ (gain of about 176), and from 2.92 s to less than 0.04 s for $k = 26$ (gain of about 73).

As expected, a smaller number of comparisons (from 1 000 000 for $k = 16$ to 20 000 for $k = 26$) leads to a significantly smaller computation time (roughly linear with it), but it is interesting to note that only the full-cache approach can follow the trend of the made comparisons. In particular, the decreasing trend of the no-cache curve soon slows down, as enhanced by the logarithmic scale. This could be due to the constant and unavoidable hard disk seeking time, whose impact is the more noticeable, the

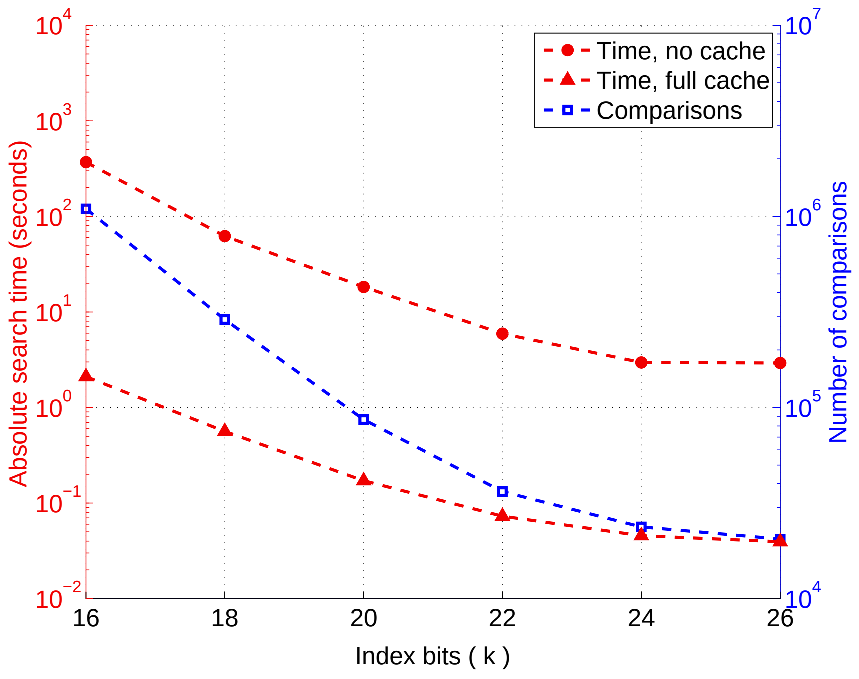


Figure 7.1: Absolute search time as a function of the index parameter k and highlighting the number of actual comparisons made.

fewer readings are performed. The average number of comparisons is about the same in both cases (i.e. with or without cache), with only tiny fluctuations.

From the presented results, it is clear that $k = 16$ does not lead to a very good performance, in comparison with higher values, since the associated index still exhibits too many candidate positions. Indeed, if we had a uniform pattern distribution in our 10 000-entry sub-databases, there would be about $2.22 \cdot 10^8/2^k$ candidate positions for each feature vector of the unknown fingerprint, thus suggesting to take k as large as possible. If we would have chosen $b = 16$, no value of k beyond 16 would have been acceptable. Taking two feature vectors at a time did not prove reliable at all, while an intermediate value of b between 16 and 32 would have lost the very practical 32-bit alignment. We pay the doubling of b in terms of doubling not only the database size, and this is generally not troublesome, but also the time required for a single comparison. Nevertheless, the dramatic reduction in the number of comparisons is enough to provide very good search times.

k	Storage (MiB)	Loading time (s)
16	802	14.84
18	810	15.33
20	840	17.14
22	960	22.87
24	1440	39.86
26	3360	86.64

Table 7.1: Index requirements in terms of storage and loading time, as a function of k. Note that the source database size is 846 MiB.

7.2 Caching

An important merit figure is the ratio between the actual playing time of the excerpt and the search time, an essential parameter for an effective, real-time operation. We report the results in Fig. 7.2, which is but 5 divided by the absolute times of Fig. 7.1. In the case $k = 24$, we have a value of about 110 in the full-cache case, compared to a bare 1.7 if we have to read every time from the storage disk. To be precise, the location list is ordered, then we could benefit from a one-entry cache, if a pattern shows more than once in a single entry, but the gain is really negligible.

For a more insightful discussion, table 7.1 shows the actual index sizes and their associated loading time, for the tested values of k. Both measures linearly scales with the sub-database size, but the index size may easily grow to 1.7 ($k = 24$) or even 4 ($k = 26$) times the associated sub-database. Given the actual cheap cost of data storage, this should not be an issue. The loading time, also, is not worrisome, since it must typically be performed only once, when the recognizing software is started. Note also that, speaking of single sub-databases and without index, the size gain factor of their 846 MiB over the corresponding PCM material (44 100 Hz, 16 bit, stereo) is 512, which is not very high. If we add the size of the index built for $k = 24$, the gain decreases to a mere 190 (we suppose that no compaction algorithm is applied). When interested in mobile applications, we should be aware of this issue.

In Fig. 7.3 we report the impact of the cache size on the absolute and relative search time. On the basis of the previous discussion, k is set to 24, so to allow also for

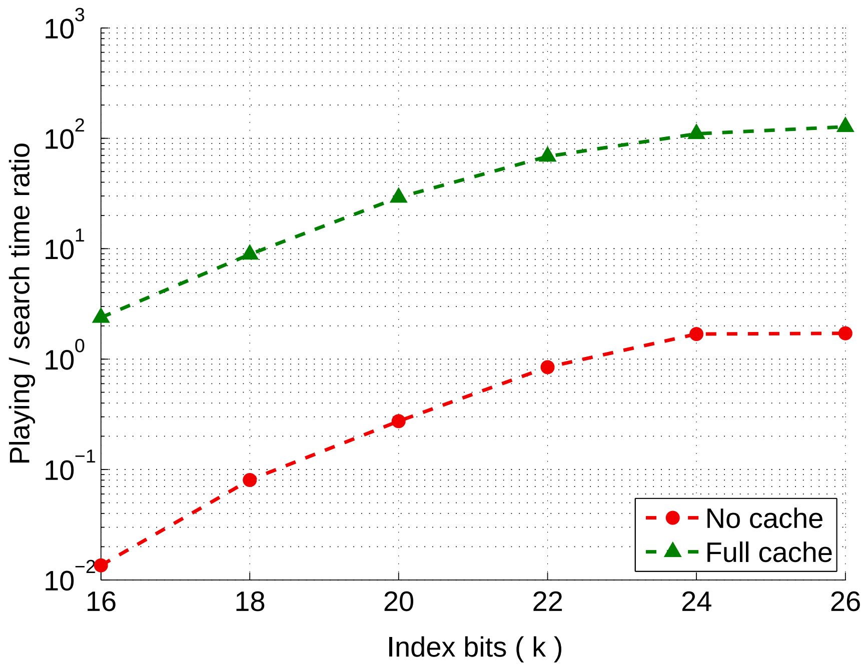


Figure 7.2: Ratio between search time and playing time of the excerpt, as a function of the index parameter k.

a fully-cached indexing. We see that the implemented cache must not be too small, in order not to worsen the performance. On the contrary, it is beneficial if it can hold at least half of the stored references. If the slow disk access is completely avoided, then – as seen above – we achieve a good gain of more than 100 over real time. It is probably possible to improve the implementation efficiency and/or the replacement policy, but we would expect similar trends.

7.3 Comparison with brute-force approach

We highlight here the benefits, in terms of search time, of both indexing and then caching over the trivial brute-force approach, as a function of the sub-database size, thus providing scalability hints. For the stated reasons, we again set $k = 24$. The exhaustive search is performed after the sub-database is loaded in memory in its entirety, and very long times (unreported here) are required if this condition is not met.

By looking at Fig. 7.4, we already notice a good improvement if we just index (thus

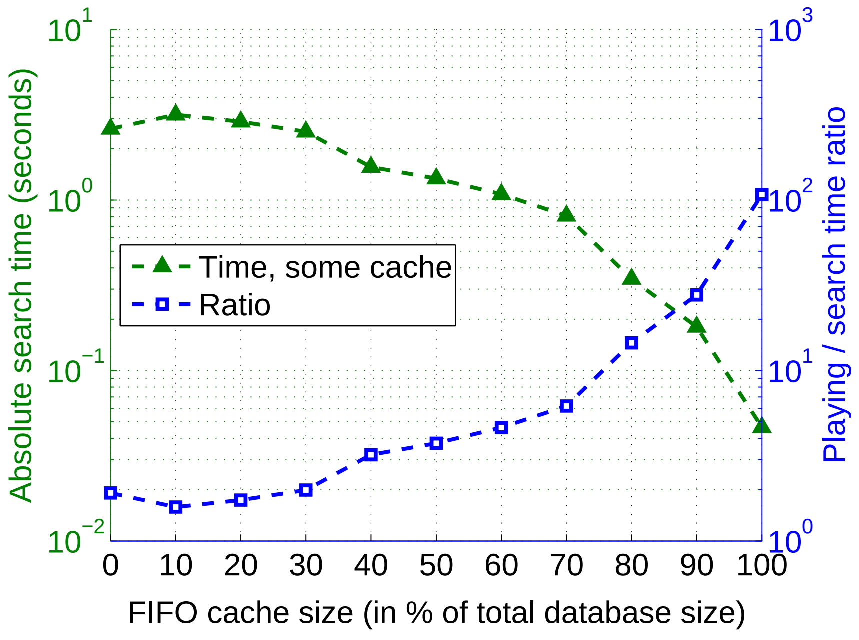


Figure 7.3: Efficiency of the cached indexing approach, in terms of absolute and relative search time, as a function of the cache size.

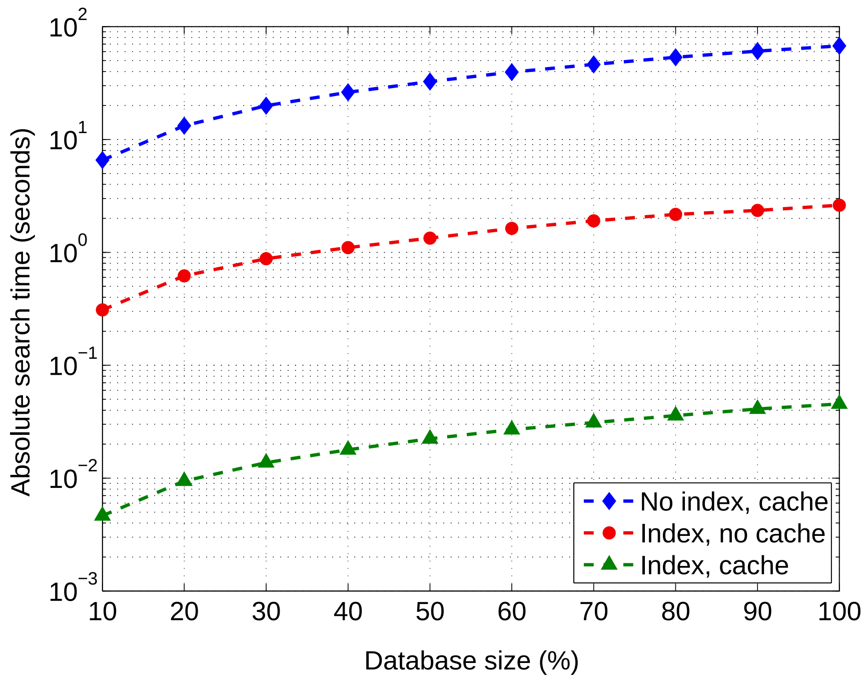


Figure 7.4: Absolute search time as a function of the database size and comparing exhaustive search (which has database on memory), indexing and fully-cached indexing. The index parameter k is 24.

accessing the disk at each comparison), with a time gain factor that ranges from about 21 for 1000 database entries, growing to over 25 when there are 10 000 of them. First of all, as seen, indexing leads to a reduced number of comparisons, which is in the order of $2 \cdot 10^8$ for brute force (this value is lower than the number of stored feature vectors because some of alignments at the end of each entry are actually not tried, since they would not allow a full distance evaluation with the unknown excerpt). In the second place, we believe that the growing trend of the gain may be well justified by the fact that fewer comparisons imply fewer disk accesses, which constitutes the real bottleneck, so their impact is progressively reduced.

A further, and significant, performance boost can come from the use of caching. If we load on memory all the entries that will be involved in comparisons, then the gain factor over brute-force is constant for every database size and evaluated in over $1,500$. The steadiness of the value well reflects the overcoming of the disk bottleneck and constitutes a valid indication for evaluating the benefit of the presented indexing scheme. On the other hand, the gain factor with the respect to the number of comparisons is around $8,500$ (see again Fig. 7.1 when $k = 24$).

Finally, Fig. 7.5 focuses instead on the ratio between playing and search time. The order of the curves is of course the opposite of that found in Fig. 7.4, and the full- and no-cache approaches may be compared with Fig. 7.2 when $k = 24$. Considering the whole sub-database, we observe a gain of almost 110 for the cached-index approach, in comparison to a value of about 2 if no cache is present and less than 0.1 when no index is present, i.e. the search time is a hundred times longer than the playing time.

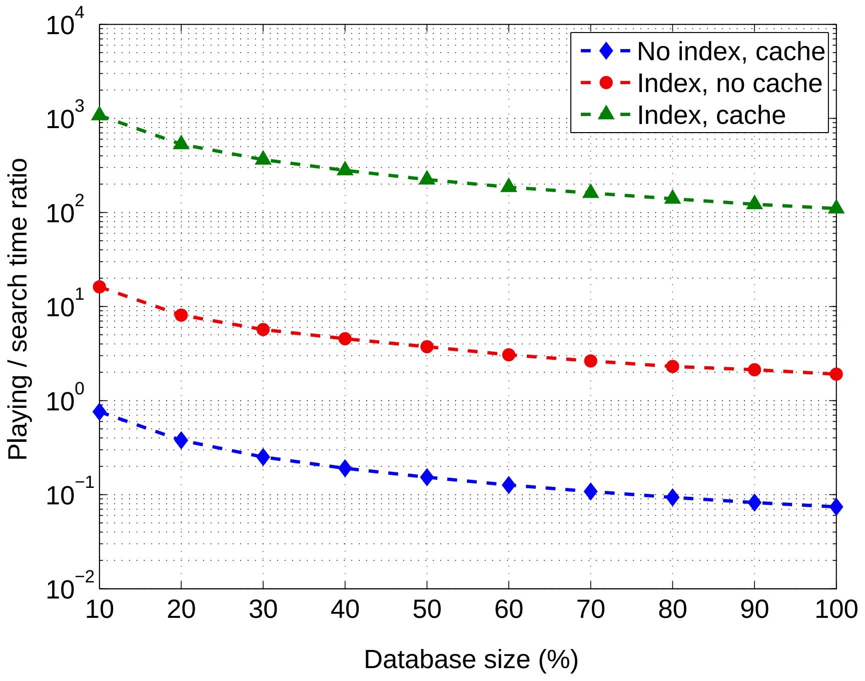


Figure 7.5: Ratio between search time and playing time of the excerpt, as a function of the database size and comparing exhaustive search (which has database on memory), indexing and fully-cached indexing. The index parameter k is 24.

Conclusions

Building on a well-known fingerprint algorithm [7], we developed and discussed a more comprehensive framework for automatic audio recognition. We mainly supported our considerations and suggestions on the basis of careful trials and experiments conducted on real and ample data, i.e. a 100 000-entry fingerprint database.

We would like to stress that databases of this size are rarely seen in the scientific literature, although there are notable exceptions in the commercial field, like the popular *Shazam* (whose available resources are unknown to the author). A second important aspect is the *efficiency* of our implementation: although still quite demanding in terms in computing power, it achieves an excellent performance, as discussed in the past sections.

A number of effective, and possibly efficient, strategies for selecting good algorithm parameters and for improving the overall performance of the system, has also been proposed. Special emphasis is put on the broadcast monitoring application, which also represents a growing market.

We believe the discussion we provided can help to a good extent not only the understanding of the topic, but also the design and use of such systems. Further improvements and refinements are yet needed for a less resource-demanding solution.

Part II

Mobility of Users: User Allocation in Wireless Networks

Chapter 8

User Mobility and Pricing Policy

The economic aspects of resource allocation in wireless networks [42] – and especially of radio resource management (RRM) – have a twofold, key effect on their real-life implementation. On the first hand, the provider aims to achieve a high revenue. At the same time, users try to afford the "best" possible service, from both an economical and technical point of view.

The service price setup can then be effective in managing the constrained bandwidth, allowing for a better coordination and a more efficient utilization. In other words, price tuning can be seen as an implicit admission control or congestion control mechanism [43–45]. Here, we concern ourselves with only the "rate" resource (i.e. the maximum throughput allocated to a user), but we could proceed in a similar way for other kinds of "resources".

In this chapter, we first present a possible model for RRM in Wireless Local Area Networks (WLANs), and then a detailed simulation study based on it. As seen in many studies, operating conditions strongly affect the performance of the network. Intuitively, according to the number of users existing in the network, the rate allocated to a specific user can range from an upper bound given by its entire signaling rate, to a rate almost equal to zero. The former is guaranteed to be assigned when there is only one user in the network, whereas if more and more users are present, then congestion will cause the traffic to be stuck by a huge amount of collisions.

The most difficult part in analyzing the multiplexing of the users on the shared radio resource is that the CSMA/CA (Carrier-Sense Multiple Access with Collision Avoidance Medium Access Control) mechanism of IEEE 802.11x protocols intrinsically does not allow easy and simple evaluations, and not even a closed expression.

As we will see in a later section, the impact of collisions and consequent intervals spent by the terminal for backoffs and retransmissions may result in a considerable difference for each terminal between the requested transmission rate and achieved performance.

We then propose a simulation study which traces the outcome of packets by considering collisions and backoff intervals of a general CSMA/CA MAC. For the sake of simplicity, we focus on the capacity constraints of a simple IEEE 802.11b hot-spot [46]. This model allows to study how the rates are affected by the CSMA/CA constraint and which is the relationship between the requested rate and the actually allocated rate. In accordance to the model, we also investigate the role of pricing in determining resource usage.

We also point out that – in the same lines of thinking – the last years saw several valid contributions. For example, an insightful study of resource allocation and pricing strategies in multi-hop networks can be found in [47], while [48] focuses on mesh networks, proposing pricing as an incentive mechanism for boosting collaboration. On the other hand, the study [49] aims to model RRM and Quality of Service (QoS) from a game-theoretic point of view.

8.1 IEEE 802.11 Distributed Coordination Function

WLANs have been deployed for many years and they are a widespread and consolidated reality. They are often found in the form of hot-spots, where a central access point is used as a gateway by a set of mobile terminals. This implementation is especially suitable for the IEEE 802.11 protocol Distributed Coordination Function (DCF), a MAC which implements a CSMA/CA policy and that we are going to briefly summarize (with no reference to propagation issues like hidden and exposed terminal).

While most of the public and private hot-spots are nowadays running the IEEE 802.11g protocol, the DCF protocol is basically the same with respect to the older, but still used, IEEE 802.11b protocol, in order to allow for compatibility with legacy devices. As previously stated, we will thus focus on IEEE 802.11b only.

In the OSI (Open System Interconnection) framework, the IEEE 802.11 protocol specify both the PHY (physical) and MAC layer, i.e. layer 1 and lower layer 2. As far as the PHY layer is concerned, we just point out the availability of different technologies, the most used being Direct Sequence Spread Spectrum (DSSS) and Orthogonal Frequency Division Multiplexing (OFDM), with a variety of signaling rates.

The intrinsically broadcast nature of the radio channel is addressed by the MAC layer through either a Distributed Coordination Function (DCF) or through a Point Coordination Function (PCF). The latter is hardly ever supported by devices, which justifies our focus on DCF.

The DCF may be implemented in wireless networks, regardless of the presence of an Access Point (AP) and builds on the CSMA/CA protocol. Whenever a station, or the AP itself, has a packet to send, it comes into play a two- or four-way handshake.

The former, said basic access, sees the simple sending of the packet, followed by the recipient's acknowledgment (ACK). Error detection is performed by means of a 32-bit Cyclic Redundancy Check (CRC).

Some details are now essential to understand the following sections. The channel must be sensed idle (i.e. transmission-free, the CSMA part of the protocol) for at least DIFS (DCF Interframe Spacing) before sending the data packet, while just for a SIFS (Short IFS) interval in the case of the ACK frame (SIFS $<$ DIFS always). In the 802.11b protocol, we have DIFS $= 50\mu$ and SIFS $= 10\mu$s. For comparison, a typical propagation delay is in the order, at most, of 1μs.

To avoid the concurrent access of competing stations (CA part of the protocol), each station waits an additional backoff time before any transmission (at least if the station sensed the channel busy at least once, i.e. it is aware that the channel is really shared). Such interval is computed as:

$$\text{backoff\_time} = \lfloor 2^{4+i} \cdot U \rfloor \cdot \text{slot\_time} \quad ,$$

where $1 \leq i \leq i_{max}$ is the number of the retransmission and U is a random variable uniformly distributed in $[0, 1]$; slot_time is dependent upon the particular PHY layer, approximately equal to 20 μs in the case of IEEE 802.11b.

An alternative formulation defines the backoff time as a uniform random variable in $[0, w - 1]$, where initially $w = CW_{min}$ (Contention Window) and w is doubled at each retransmission up to a value CW_{max}. In the case of IEEE 802.11b with DSSS, $CW_{min} = 32$ and $CW_{max} = 1024$ (the unit of measure is the slot time). An appropriate tuning of these parameters allows to dynamically improve the overall throughput, and to implement priority classes.

In any case, the backoff time may be seen as a countdown which always follows a DIFS period: it starts and keeps counting after the channel has been sensed free for DIFS, stops when the channel is busy, resumes after a further transmission-free DIFS. Each unsuccessful delivery attempt increases the backoff timer, and the packet is discarded after too many attempts. As an extra safety measure, when the channel is estimated as too heavily degraded or crowded (e.g. the station receives a corrupted packet, as detected by the CRC) the backoff time is increased by EIFS (Extended IFS).

A second packet exchange method is four-way and includes an additional couple of control messages, called Request To Send (RTS) and Clear To Send (CTS). Instead of directly sending the data message after DIFS and backoff, a station sends first a RTS packet. The recipient, after SIFS, replies with a CTS. The actual packet exchange (which is acknowledged) then takes place as in the previous case.

The RTS and CTS signal all listening stations to defer their own transmissions, so that collisions may happen only while RTS is being sent. Since RTS and CTS are very short frames, the duration of the collision is greatly reduced, leading to significant benefits for the overall system. There is in fact a better throughput and a notable saving in power consumption (especially important for portable devices). In some cases, of course, the RTS/CTS overhead may be non-negligible. A trade-off analysis may be found in [50].

8.2 The MEDUSA model

The MEDUSA (Micro-economic Elastic Decentralized Users's Service Acceptance) model was introduced several years ago [51] in order to jointly take into account not only RRM and QoS, but also the price paid by the users for achieving a given service level.

The model comprises microeconomics and game-theoretic elements: in fact, we can see telecommunications networks as non-cooperative games, where each user is a player who makes decisions on the basis of some *utility function*. Such utility functions are the tool for modeling their preferences and perceived QoS. In our case, their *preferences* are the desirable transmission rates to be granted by the AP.

In this scenario, each user can independently decide whether to join, stay in, or leave the network. Moreover, the provider itself can be considered a player, whose goal is maximizing its revenue. The model is therefore completely decentralized.

As previously stated, we consider here a simple system, with one AP and a number of stations. It may be studied under different points of view, such as its total QoS (estimated on the basis of each user's figure), the number of admitted users, the total allocated resource, or also the total revenue of the provider.

The MEDUSA model introduces a user utility function $u(r)$, where r is a scalar, continuous variable which represent the resource allocated to the user. The multi-variate case can be seen as an easy extension, in which all components are treated one at a time. We consider here the resource *rate*, i.e. the bandwidth (usually measured in bits per second, bps) allocated to a user. This constrains of course the values for r.

For example, in the IEEE 802.11b case, we have, at least approximately:

$$\sum_{i=1}^{N} r_i = 11 \text{ Mbps} \quad .$$

Utility functions are representative of user's satisfaction and they customarily satisfy to at least the following simple constraints:

$$\frac{du(r)}{dr} \geq 0$$

$$\lim_{r \to +\infty} \frac{du(r)}{dr} = 0 \quad ,$$

which reflect the increasing of user's perceived QoS (and then utility) with an increased resource allocation, but also the progressive flatting in the increased satisfaction. In other words, there is a threshold beyond which no relevant improvements in user satisfaction are attained, even when significantly allocating more resource to the user.

If we assume a normalized upper limit for $u(r)$, a possible choice for the utility function is

$$u(r) = \frac{(r/\kappa)^{\zeta}}{1 + (r/\kappa)^{\zeta}} \quad ,$$

where $\zeta \geq 2$ and $\kappa > 0$ can be freely chosen in order to represent the different choices of different users. This function is sigmoid-shaped, which is often used in these cases.

If r is the "rate" resource, we speak of elastic traffic when $u'(r)$ is continuous for every r. This assumption makes sense especially in soft-capacity systems, e.g. 802.11x and 3G+ networks.

In a similar fashion, a reasonable pricing function $p(r)$ must be such that, at least:

$$\frac{dp(r)}{dr} \geq 0 \quad ,$$

while other constraints may be freely chosen by the service provider. We recall that price strategies must be decided in advance and have a decisive impact on the use of the service. Our choice is for a linear pricing function:

$$p(r) = \alpha r \quad ,$$

with $\alpha > 0$, the same of every user.

Given $u(r)$ and $p(r)$, how do we actually put them into play? They must be linked through a *service acceptance probability* function, which will be used to simulate user's decisions. Intuitively, this function must depend upon the utility and the price in an increasing and decreasing fashion respectively:

$$\frac{\partial A}{\partial u} \geq 0$$

$$\frac{\partial A}{\partial p} \leq 0 \quad .$$

A function that represents a probability (i.e. is valued in $(0, 1)$) and satisfy to these constraints is:

$$A(u, p) = 1 - e^{-ku^{\mu}(p/\phi)^{-\epsilon}} \quad ,$$

where k, μ, ϵ are positive constants representing respectively a scaling factor, the sensitivity to the utility, the sensitivity to the price. Also in this case, the parameters may be conveniently adjusted to take into account different kinds of users. Further details on the acceptance probability function may be found in [51].

The parameter ϕ is a convenient normalization factor, representing the price which is considered fair by a fraction, say A_0, of users when they get maximum utility, and thus $A(1, \phi) = A_0$. This leads to having $k = -\log(1 - A_0)$.

The service provider revenue may be computed easily from the above definitions. If N users are in the system, and user i is allocated a resource r_i, it is possible to define the system allocation vector $\mathbf{r} = \{r_1, r_2, \ldots r_N\}$ and the total revenue:

$$R(r) = \sum_{i=1}^{N} R_i = \sum_{i=1}^{N} p_i A(u_i, p_i) \quad .$$

Let us point out that the resource allocation is centralized in the AP, while it is up to the user whether to accept or decline the service.

The allocation policy is described in the next section and, in general, may be decided on the solution of some (constrained) optimization problem. The optimal strategy should be tuned on the $u_i(r)$, and changed whenever some user leaves or joins. This is usually unfeasible, first of all because users' preferences are unknown.

8.2.1 A Preliminary Investigation

An immediate way to apply the MEDUSA framework is as follows. Firstly, we let each user determine its most preferable transmission rate r_i as:

$$r_i = \operatorname*{argmax}_r A(u_i(r), p(r)) \quad .$$

Then, we perform a feasibility check, i.e., we control whether the total requested rates exceed the maximum capacity, say 11 Mbps. If this is the case, we assume that the packets get lost and the rate of the owning user is decreased. In doing this, we adopt a sequential approach which tries to capture the fact that in a real WLAN users are allocated one at a time.

From an idealized point of view, the CSMA/CA capacity can be represented by the fact that users can always be allocated until a saturation point is reached. If the bandwidth allocation is perfectly elastic, when the new user's request exceeds the available bandwidth, the allocation vector is rescaled in order to satisfy the bandwidth constraint (we can speak of a proportional reallocation).

This can lead to a rough evaluation of the capacity, quite correct from the qualitative point of view. The conclusions drawn from here are unfortunately way too optimistic, since the throughput and all related metrics are highly overestimated. The interested reader may found simulative results in [52].

The main problem in this kind of analysis, in fact, is that the overhead and the consequent throughput decrease of the IEEE 802.11b MAC can not be taken into account. To this end, we developed a more detailed simulator, which is explained in the following section.

8.3 Simulator

We recall that the scenario is a hot-spot with a single AP. We are not concerned here with the physical characteristics of the system, such as the geographical position of the users, the size of the cell, or the propagation issues. We assume just that stations are within mutual reach of the AP and put our focus on how the sharing of the CSMA/CA DCF channel affect the QoS of each user, their decision whether to leave or stay, and eventually the revenue of the provider.

Our simulator is Monte Carlo-like and takes into account a wide variety of the parameters introduced in the previous section. We consider a discrete timeline of 10 seconds of transmission, where temporal intervals can be allocated to the users. In this way, we try to capture the scheduling of time slots, where the exchange of signaling and data packets is performed. On this basis, we evaluate carrier sensing, collisions and consequent rescheduling of packet transmissions due to exponential backoff.

The simulator, written in MATLAB [1], works at packet level and is approximately event-based. This allows us to actually count the collisions when they occur and to track the actual rate experienced by users with a good approximation.

In particular, we lay down on the timeline, one user at a time, all the packets she will send in the given 10 s. When a user is added to the system, we evaluate her impact, in terms of QoS, on all the other users. The allocation algorithm mimics the CSMA/CA rationale previously described. Since all the signaling packets (ACK, RTS/CTS) are typically very small compared to data packets, they are neglected.

In detail, with reference to the Kendall notation, we assume an M/G/1 system, that is:

- a Poisson generation process for packets to send. This means they have an

[1]Source code available upon request.

exponential (memory-less) interarrival time with average $1/\lambda$, where λ is the packet generation rate in packets per second;

- a general service policy, which depends upon the simulator implementation (detailed in the following paragraphs);
- a single server, which is constituted by the only channel taken into account.

In order to simulate different data rates, we chose to have a fixed data packet size of 8 KiB, hence the packet arrival rates λ are directly derived from the rates r_i allocated according to the MEDUSA model, as:

$$\lambda_i = \frac{r_i}{\text{packet\_size}}$$

In other words, the higher rate assigned to a user is simulated by sending more packets from that user. The generation rate λ is then used to randomly draw the start of each packet on the timeline (as a difference with respect to the previous arrival instant).

Each slot in the timeline represents both a certain amount of bits, and a corresponding time fraction. To be precise, assuming the 802.11b rate of 11 Mbps and slots of 1024 bits, we have exactly 112640 slots in our 10 s-timeline, each equal to about 89 μs. Therefore, a packet of 8 KiB uses the channel for exactly 64 slots. A higher accuracy may be attained by reducing the size of the slots, or increasing the duration of the simulated period. According to the positions above, the starting slot of the next packet is generated by first sampling an exponential distribution of appropriate parameter *lambda*, and then by converting this continuous value into our discrete timeline.

We allocate users sequentially (and their packets as well), to save computational complexity, and only the currently considered user is simulated as “active”, i.e. able to reschedule packets in case of collisions, as will be soon cleared. The data rate achieved by user i is approximately equal to r_i if its packets do not collide with others allocated previously.

If the allocation of a new series of time intervals, corresponding to a new user entering the network, overlaps with already allocated users, our simulator checks for every overlap if it happens within an exposure time corresponding to one 64-th of the packet (i.e. one slot, or 128 bytes, or 89 μs). Note that different values of the packet size and the exposure time have also been tried, showing only small quantitative variations. However, the approach can be tailored to the specific characteristics of a given IEEE 802.11x implementation.

Considering the discrete timeline, in allocating the time slots to the generated packets, the following cases may arise (recall that all packets have the same size):

1. no overlap: in this case, the slots are simply marked as allocated to the current (new) user;

2. partial overlap, and the *new* user's packet starts *after* the start of an existing packet;

3. partial overlap, and the *new* user's packet starts *before* the start of an existing packet;

4. complete overlap.

Let's see how we worked out each of the non-trivial cases.

8.3.1 Partial Overlap

For every overlap of packets, if the transmission starting points are further than the exposure time, the packet with the earlier transmission time is kept, whereas the other one is rescheduled with an exponential backoff, if it belongs to the new user. This aims at simulating that the second user has sensed the transmission of the first one and its packet has been rescheduled.

In the discrete timeline, this translates in having two packets partially (but *not completely*) overlapped. Fig. 8.1 depicts the case when the currently considered user reschedules its transmission due to the busy channel. On the other hand, as in Fig. 8.2,

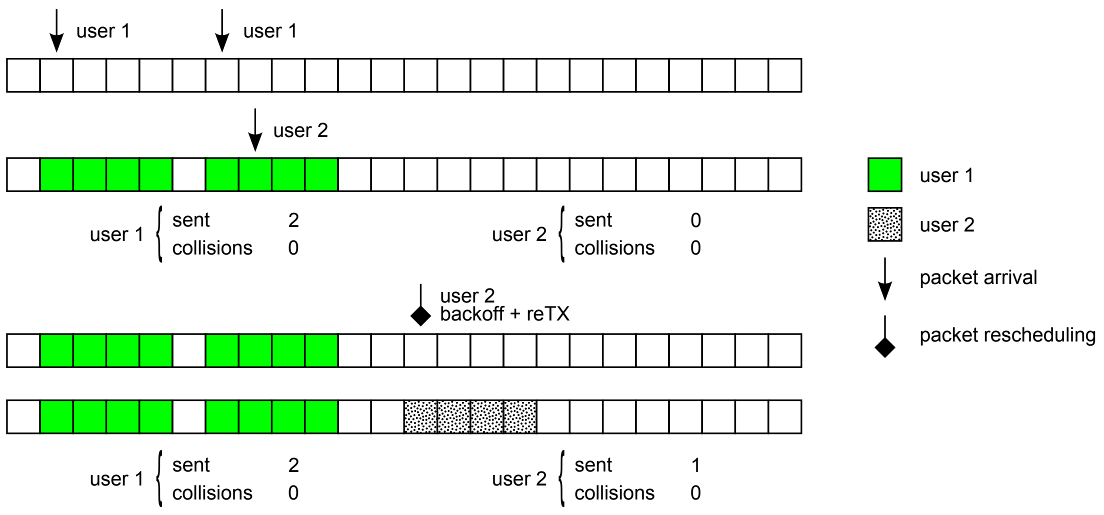

user 1
user 1
user 2
user 1 sent 2 collisions 0
user 2 sent 0 collisions 0
user 2 backoff + reTX
user 1 sent 2 collisions 0
user 2 sent 1 collisions 0
user 1
user 2
packet arrival
packet rescheduling

Figure 8.1: An example of backoff rescheduling when the current user senses a busy channel. For clarity, a packet is here only 4 slots long, instead of 64.

if the current user's packet temporally precedes an existing one, the former is kept, while no backoff is considered for the latter, which is just lost and counted as a collision (thus reducing its owner's experienced rate).

As a third case of overlapping out of the exposure time, if both overlapping packets correspond to a same user (i.e. the interarrival time is too short to fall after its own previous packet), the second packet is simply postponed, as depicted in Fig. 8.3.

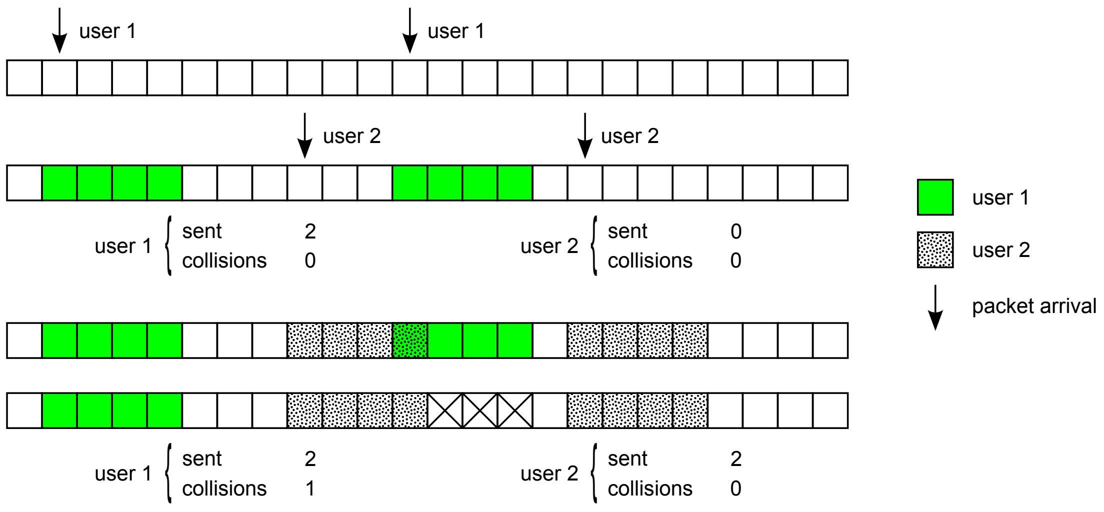

user 1
user 1
user 2
user 2
user 1 sent 2 collisions 0
user 2 sent 0 collisions 0
user 1 sent 2 collisions 1
user 2 sent 2 collisions 0
user 1
user 2
packet arrival

Figure 8.2: As an approximation, no backoff is considered if the converse of Fig. 8.1 happens: here, the packet of the current user *precedes* that of an existing user. The "X"s signal the removal of the packet (see section 8.3.2) from the simulated timeline.

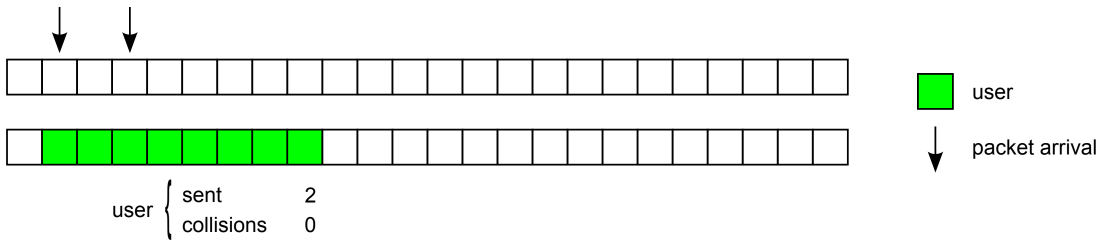


Figure 8.3: Postponing of next packet in case of short interarrival times.

8.3.2 Complete Overlap

There is also the chance that overlap occurs within the exposure time: a collision then arises, and both packets are lost. In an approximate way, and consistently with a previous case, the backoff process is initiated for the second user only. This leads to a slight underestimation of the actual rate experienced by the first user. In the discrete timeline, this corresponds to having identical arrival slots for different users (see Fig. 8.4).

We point out that every time a collision arises, in any way, one of the two users does not have the opportunity to reschedule its packet. However, we also proceed to *remove* the offending packet from the timeline, thus allowing future users, in our accurate Monte Carlo approach, to compensate for that. All users have in fact the same weight and the same parameters distribution.

8.3.3 Evaluating Users' Reactions

After having added a new user and the evaluation time of the virtual 10 seconds, it is possible to evaluate the rate actually achieved by each user, and to analyze *a posteriori* the acceptance (stay/leave) of this value. In particular, the actual rate experienced by user i is computed as:

$$\rho_i = \frac{\text{packets\_successfully\_sent}}{\lambda_i \cdot \text{simulation\_period}} \cdot r_i \quad ,$$

where r_i is the initial allocation. It should be noted that even when no collisions occur, $r_i \neq \rho_i$ in general, since the interarrival time is randomly computed.

Actually, dynamic allocation changes imply that users might be happy with the initial allocation, but refuse the degradation in the service due to lower r_i. For this reason, after each allocation of a new user, we re-evaluate the acceptance probability of all users with a conditional approach, according to the Bayes theorem. We assume that if the users request is r_i, a lower allocation $\rho_i < r_i$ is accepted, conditioned on

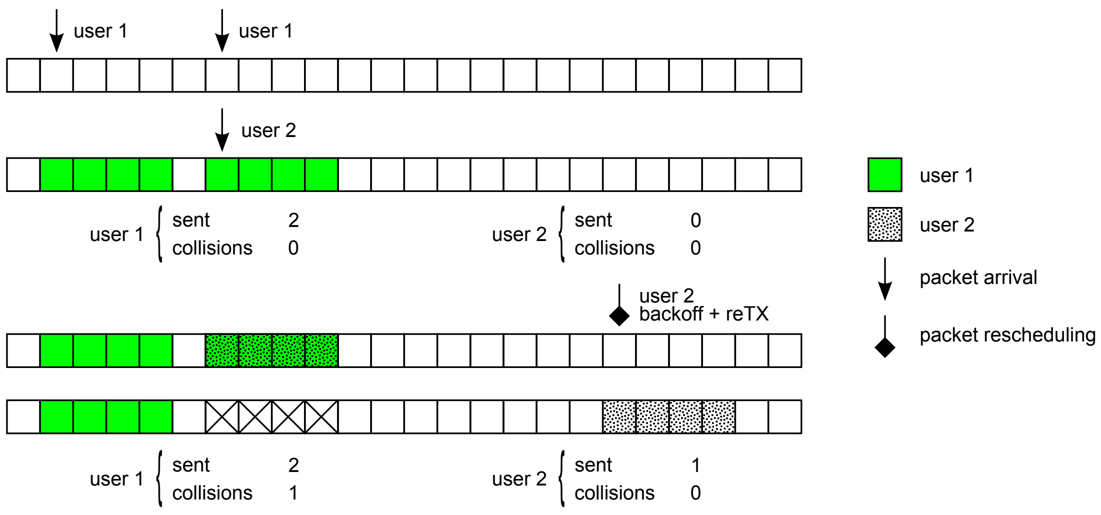

user 1
user 1
user 2
user 1 sent 2 collisions 0
user 2 sent 0 collisions 0
user 2 backoff + reTX
user 1 sent 2 collisions 1
user 2 sent 1 collisions 0
user 1
user 2
packet arrival
packet rescheduling

Figure 8.4: Collisions occur if two packets are scheduled the same starting slot (mimicking the real-life exposure time). The backoff process is initiated by the second user only.

the fact that r_i is acceptable, with the following probability:

$$A(\rho_i|r_i) = \begin{cases} \frac{A(\rho_i)}{A(r_i)} & \text{if } A(\rho_i) \leq A(r_i) \\ 1 & \text{if } A(\rho_i) > A(r_i) \end{cases},$$

where the first case accounts simply for the basic concept of conditional probability, where the second considers that users will never refuse a service improvement.

Unsatisfied users are thus removed and another timeline of 10 seconds is considered, where only the remaining users are allocated. This approach might be heavy from the computational point of view, nevertheless it is also capable to obtain realistic values for the rate allocation according to the IEEE 802.11 MAC.

The re-evaluation on the $A(\rho_i|r_i)$'s of the users' satisfaction might imply that already allocated users can leave the service, so that a certain amount of resource is freed and could be iteratively re-allocated. However, for the sake of an easy computation, the amount of freed resource is taken into account only in future allocations. This does not introduce inconsistencies in the results, even though tiny oscillations of the curves can be found. Actually, we highlight that the accurate Monte Carlo simulations allow to greatly smooth and overcome this approximation.

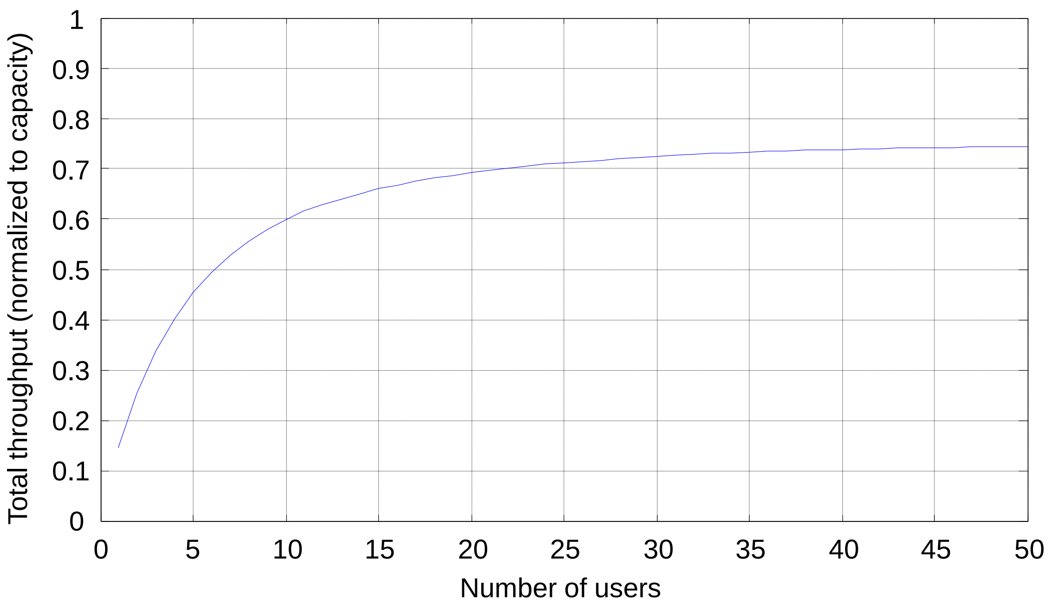


Figure 8.5: Cell throughput in condition of saturation and in the hypothesis of no users leaving the system.

8.3.4 On the Validity of the Simulator

Besides the meaningful results presented in the next section, a very good indication of the well-behaving of our allocation algorithm design may be found if we consider the following:

- all N users accept the service and join the hot-spot;
- no user leaves;
- each user is allocated a fixed rate $\overline{r}$, such that the users can oversaturate the system, i.e. $N \cdot \overline{r}$ is greater than the hot-spot capacity.

In particular, for $\overline{r} = 1.6$ and $N = 50$, the measured throughput in a 11 Mbps-cell is depicted in Fig. 8.5. We can see that the system saturates at about 75%, which is in good agreement with the theoretical studies [50, 53].

Parameter	Value
WLAN capacity	11 Mbps
ζ	uniform in $[2 \div 14]$
κ	uniform in $[0.01 \div 0.125]$
ϕ	1.0
μ	2.0
ϵ	4.0

Table 8.1: Parameters for the MEDUSA model.

8.4 Results

The simulation scenario consists of N potential users connected to an IEEE 802.11b AP to simulate the WLAN hot-spot having a maximum capacity of 11 Mbps. We assume that all users are covered by the same access point and multi-hop capability is not present. Table 8.1 shows the parameters of the MEDUSA model to fully specify the formulae presented in the previous sections. In these results, we always trace N on the x-axis, i.e. we investigate how the performance is influenced by variations in the number of potential users. Also, we consider different values of the price coefficient α to study the effect of pricing.

Figure 8.6 shows the number of admitted users, i.e., the number of users who accept the service conditions vs. the total number of users in the system. The acceptance decreases as the price increases, but still we can observe an increasing behavior for all curves, as more users are present. This implies that the IEEE 802.11b MAC protocol is able to allocate users in an elastic manner.

Approximately, we observe a linear increase common to all price curves, then when the capacity is saturated the slope decreases but the increase remains more or less linear. The larger the price, the further the number of users at which the slope change happens, and also the lower the final slope. The acceptance rate evaluation is better emphasized in Figure 8.7, where the values of Figure 8.6 are represented as a fraction of the potential users.

Figure 8.8 represents the total amount of allocated resource as a function of the load. This can be seen as a throughput estimate, since the CSMA/CA capacity is

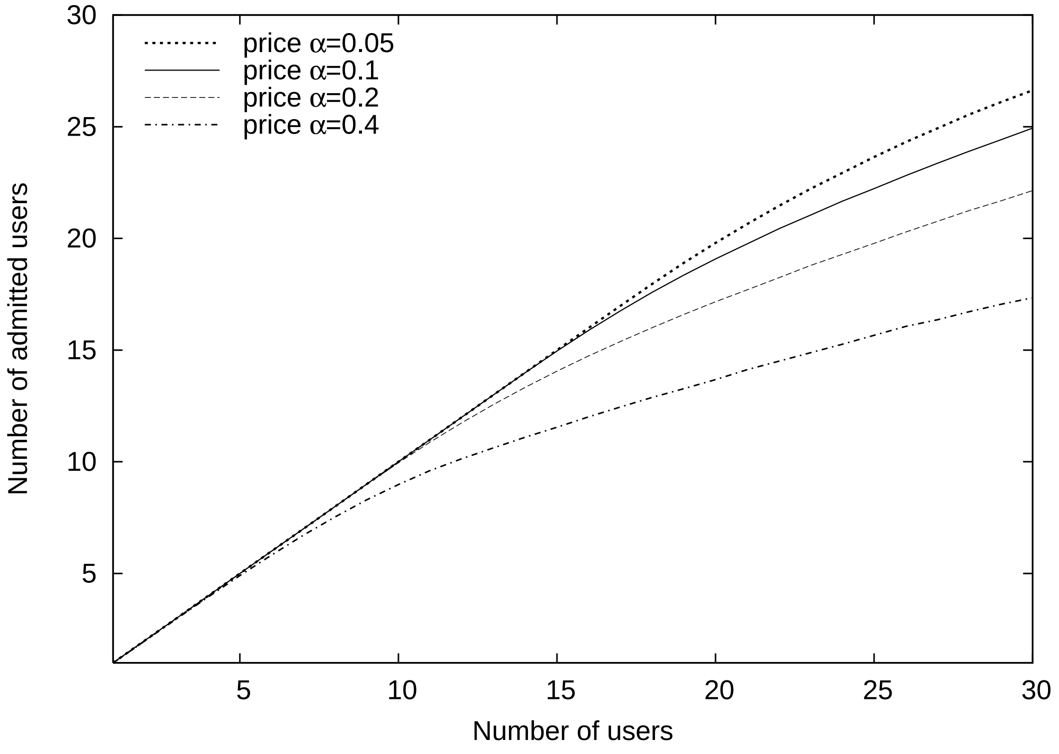


Figure 8.6: Number of admitted users as a function of the load

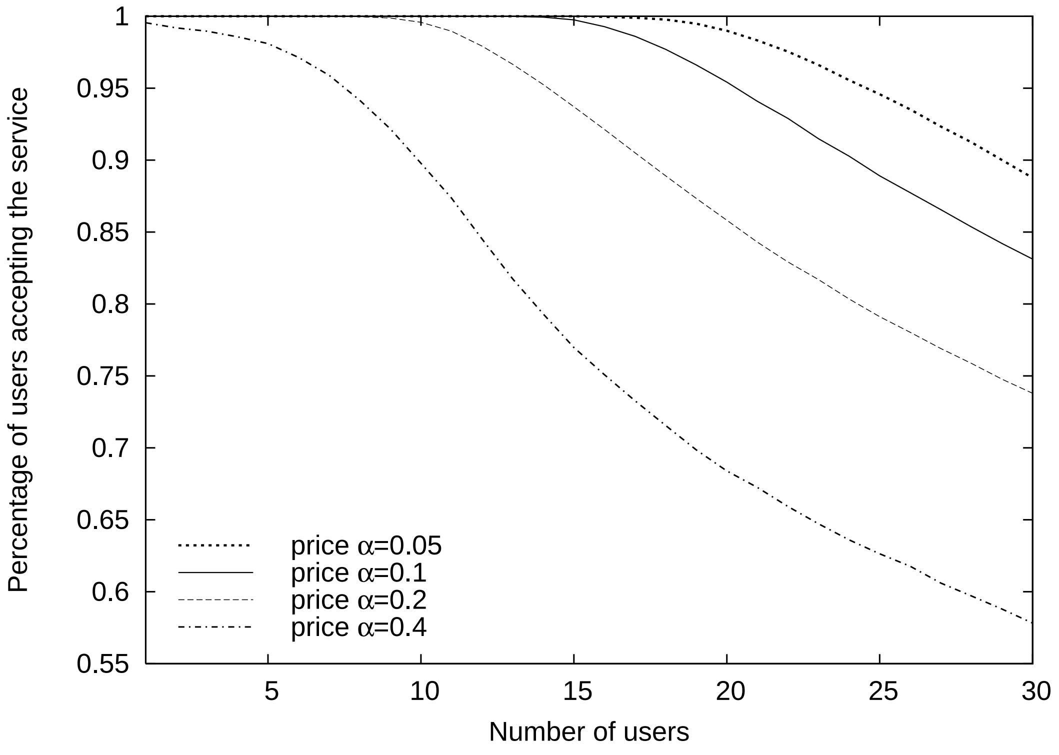


Figure 8.7: Admission rate as a function of the load

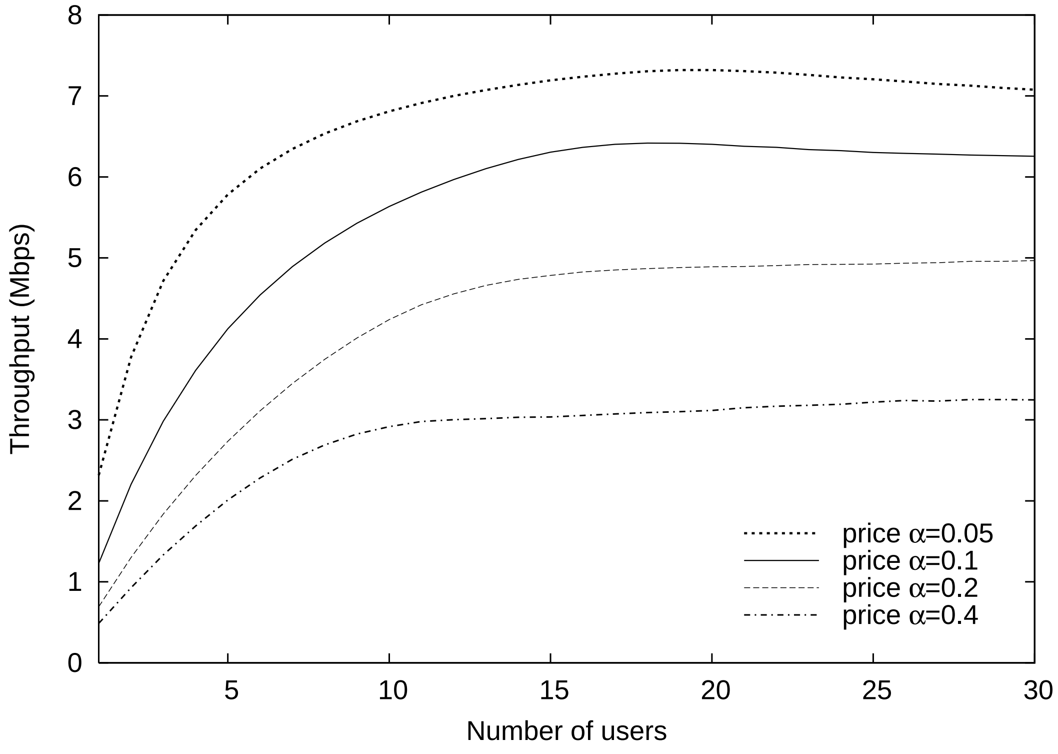


Figure 8.8: Network throughput as a function of the load

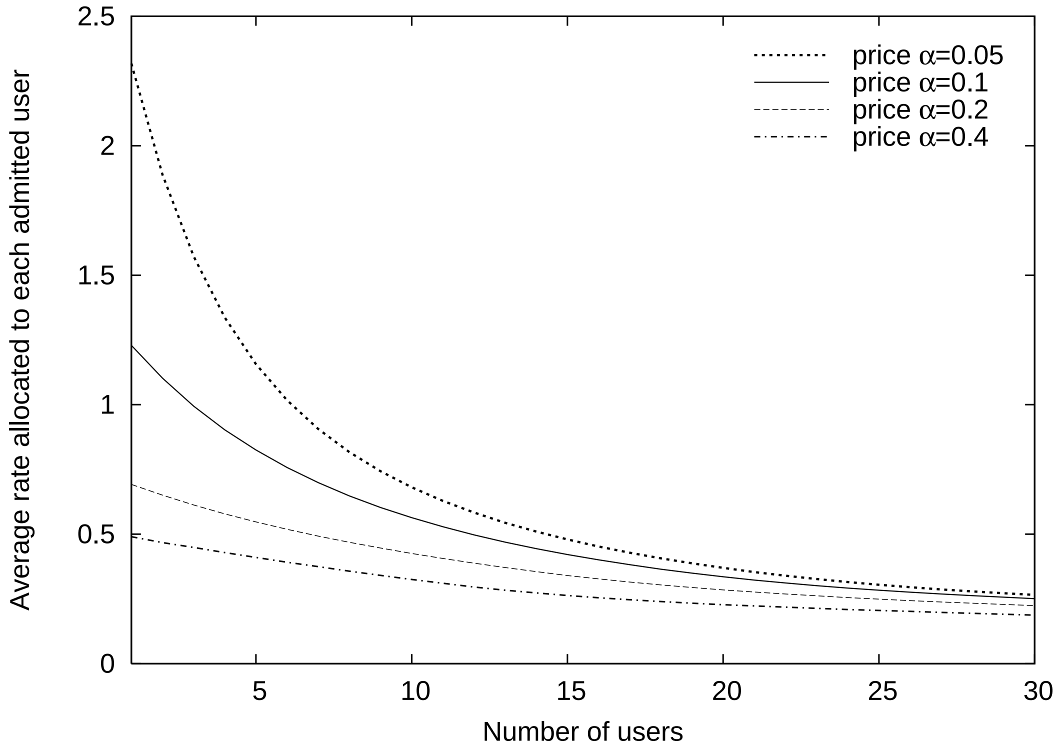


Figure 8.9: Average rate per user as a function of the load

actually taken into account in neglecting collided packets or overhead in the sum of the r_i's. It is highlighted that the throughput increases linearly at first in the number of users, then it saturates to a value which is decreasing as the price increase. It is also visible that the throughput saturation value can be significantly lower when the price is high, and this is due not only to the presence of network protocol overhead, but also to the fact that rate decreases implied by protocol inefficiencies cause the non-satisfaction of more users.

In Figure 8.9, we represent the same situation, but considering the ratio to the number of admitted users. With respect to Figure 8.8, here it is shown that the average allocated rate tends to be more or less similar when the number of users is sufficiently large. However, from the previous results it is also clear that the difference is in the number of satisfied users vs. the total allocated resource, i.e. when the price is high we allocate approximately the same rate to the users, but fewer users accept and then the throughput is lower.

Finally, we consider the revenue in Figure 8.10, whose behavior follows the throughput since the price is linear. However, the trend of the saturation value vs. the price choice is the opposite of Figure 8.8. Since doubling the price does not mean a halved throughput, the price increase improves the network management from the point of view of achieving high revenues. This means that the price choice is not trivial, and must be accurately checked according to the provider's objective. In fact, contrasting results can be found if the network management objective is either high resource allocation or high revenue. In general, both goals present pros and cons, and cutting the trade-off is needed.

As a general remark, we observe that the results are qualitatively similar to the ones obtained through the more approximate method described in section 8.2.1. Thus, we infer that for a simple qualitative analysis, a preliminary investigation, where the inherent MAC characteristics are neglected, is sufficient. On the other hand, for a more detailed analysis and also for a realistic numerical evaluation, the specific MAC protocol must be taken into account in some way.

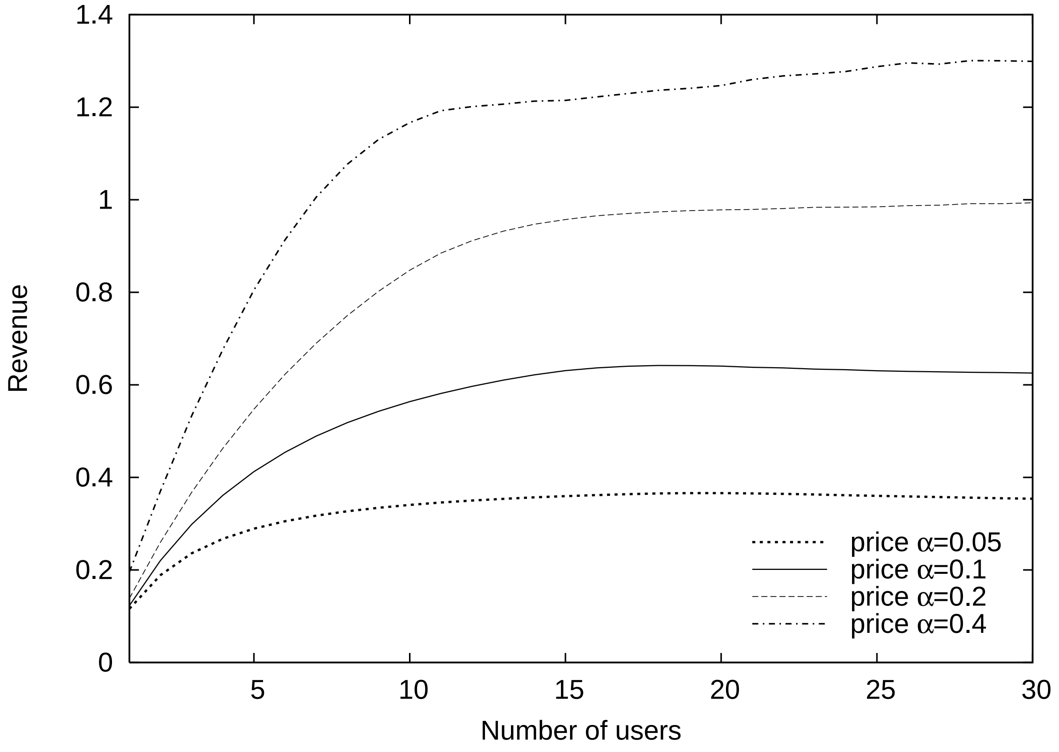


Figure 8.10: Total revenue as a function of the load

Chapter 9

User Mobility and Cooperation

In these years, given the always wider availability of high-capacity wireless networks (like WiFi hot-spots and 3G+ cells), much effort has been put into investigating the issue of selecting the best point of access in a multi-access and possibly multi-technology scenario. For the sake of clarity, we focus on the problem of Access Point (AP) selection in IEEE 802.11x wireless networks. However, we also provide discussion on feasible extensions to more complex scenarios.

The aim of AP selection techniques is working out new strategies devoted to assisting wireless devices in choosing the best AP among the many available, where "best" may be differently interpreted, e.g. in terms of throughput, delay or price, possibly summarized by some user-specific utility function. An application of utility functions is presented in the next chapter 8. Thus, it is not simply a matter of finding the strongest signal.

Even if widely used as a link quality estimator, RSSI (Received Signal Strength Indicator) is indeed defective if used as the only metric, since many other issues may arise (like congestion or service continuity). Actually, recent works invariably show the limitations of RSSI-based metrics [54] and then introduce smarter decision schemes.

They achieve notable improvements by either deeply exploiting existing IEEE 802.11x protocols [55], or relying on minor extensions [56]. Real-life [57], experimental or simulative tests prove the need and effectiveness of the approaches. However, none of them consider the potential advantage given by *user* cooperation.

Often associated to cognitive networks (as in [58]), cooperation in wireless networks is quite an active research topic [59]. Because of its manifold nature, there is no single definition for the term, which is relevant to different layers and entities. Our focus is on the *collective* aspect of cooperation: "establishing and maintaining a network of collaborating nodes aiming to a mutual benefit" [60].

Within the framework of AP selection, in order to improve users' overall Quality of Service (QoS), we propose a strategy of user cooperation and, by means of simulation, show strong clues of its rewarding features. In our proposal, no backing from the infrastructure is needed, and thus the APs need not to support new protocols or signaling. Although we consider a number of parameters (like AP throughput and delay), we are not concerned on how they are estimated: almost any of the cited methods, and possibly others, may be employed.

This contribution may be better appreciated in the light of the Always Best Connected networks (ABC) concept [61]. Fodor *et al.* propose mechanisms that will allow users to always select the "best" network on the basis of their preferences. Their comprehensive approach foresees a dense, multi-access environment which includes a strong network support, as providing an end-to-end bearer service for QoS distribution. However, at the present time, this architecture assumption cannot hold. Therefore, we argue that a simpler and readily available technique is needed, in order to push users' QoS beyond its present limits.

An advanced proposal, entirely user-based but not exploiting cooperation, is BreadCrumbs [62], whose basic idea is considering the "derivative of connectivity". Users build a history of the AP encountered along usual paths and optimize their QoS and battery consumption by forecasting AP availability and their related connection quality. Each user works independently and about a week of training is required in order to get some sensible improvements.

The problem of increasing users' QoS may also be addressed by reducing the congestion in the WiFi hot-spots and balancing the load among the available APs. [63–65] all focus on this aspect and may be used in conjunction with our approach. They

i	AP (p_i)	STA (q_i)
0	Residual capacity	Bandwidth required
1	Packet delay	Packet delay required
2	Service availability	Services required
3	Price offered	Price required

Table 9.1: Meaning of the parameters in the user cooperation model. Each AP or STA is characterized by a set of four parameters, identified by p_i or q_i respectively.

require an explicit support by the underlying networks, that may also work together in a coordinated way. Users are directed to switch channel or network, if this can improve the overall performance of the network without impairing them, and can negotiate a QoS on the basis of some admission control protocols.

Finally, from the game-theoretic point of view, the problem of designing distributed QoS-improving techniques has been addressed in [66]. There, independent WLANs act as players, whose interaction – but no explicit coordination – results in a better spectrum usage. Different multi-stage strategies are evaluated, especially with the aim of finding out whether and when they can pay off and their trade-offs with respect to the additional resources needed for such algorithms.

9.1 Scenario and Simulator Operation

Our scenario is constituted by N_{AP} APs and N_{U} users (or nodes, or stations, STAs). Each entity (AP or STA) is described by a set of four real parameters valued in $[0, 1]$, denoted by p_i for APs, and by q_i for users, $0 \leq i \leq 3$.

These four parameters describe bandwidth, delay, openness and price. For an AP, they are normalized quantities that respectively measure its current state in terms of residual capacity, average delay experienced by packets, availability of services to affiliated users (as open ports, static IP, $\dots$), potential fee for granting access or traffic. For STAs, they instead represent the corresponding service level required, where 1 means the highest importance and 0 is indifference. Table 9.1 summarizes the meaning of these parameters. We select here a set of four relevant characteristics, but similar considerations may well hold for different choices.

We assume that any generic node u can completely estimate any AP it sees, i.e. all of that AP parameters which interest the node. This is quite a reasonable assumption, since u can passively monitor the radio environment [57] or temporarily associate with the AP and then evaluate them [55]. Since we require $p_i \in [0, 1]$, user u must also choose convenient normalization constants $\hat{p}_i$, as in

$$p_i = \max(\frac{\tilde{p}_i}{\hat{p}_i}, 1) \quad ,$$

where $\tilde{p}_i$ is the physically measured quantity. Normalization constants should be determined by the particular scenario. For example, when $i = 0$ (rate) and we are in a cloud of 802.11b hot-spots, then $\hat{p}_i = 11$ Mbps can be a good choice.

If an AP has $p_0 \approx 1$, it means the node estimates that the AP is more or less idle, while a value of $p_0 = 0.5$ means that the AP is using about half of its capacity. It is clear that the AP estimation methods employed heavily impacts the effectiveness of the solution we are proposing.

In addition, we assume a single cooperation pool $\mathcal{C}$, to which a user may or may not belong. In other words, users can be either cooperative or non-cooperative. Lastly, other peculiar AP characteristics, though currently unused, are its SSID (Service Set Identifier, broadcast by the AP in WiFi networks) and load estimate, both useful for future model refinements.

A scenario is generated by completely specifying or randomly drawing all the four parameters of each AP or user. Each user u is then assigned a non-empty random set of visible APs, i.e. whose coverage area include user u. We denote such set with $\mathrm{AP}(u)$. Obviously, $1 \leq |\mathrm{AP}(u)| \leq N_{\mathrm{AP}}$, though it will usually be much smaller than its upper limit.

The simulator[1] operates at high level, without taking into account the actual underlying packet exchange. This results in extremely fast simulations (C++ implemen-

[1]Source code available upon request.

tation). In particular, we employ a hard capacity constraint, which may not always be realistic, but it is a good approximation in high-load conditions. A more precise approach could employ the Channel Utilization Estimate (CUE) proposed in [67], where also the packet size, and not just a raw bandwidth consumption, is included in computing the effective network usage. We also point out that our approach is not concerned on physical details, like the propagation model, even though we do include the possibility of incorrect packet reception, as explained later.

At the beginning, each user u is assigned a set of four normalized weights $-1 \leq w_i(u) \leq 1$, which represent the priority given by the user to the i-th parameter. They must be such that

$$\sum_{i=0}^{3} |w_i(u)| = 1 \quad .$$

Typically, we set positive values for the weights related to bandwidth and openness, since they are beneficial, while negative weights can well characterize undesirable quantities like delay and price. We stress the dependence of $w_i(u)$ on user u.

Each station u then computes, for each AP $j \in \mathrm{AP}(u)$ (i.e. for each AP j visible to u), a merit figure

$$\Gamma_{\mathrm{AP}}(u, j) = \sum_{i=0}^{3} p_i(j) w_i(u) + \max(-w_i(u), 0) \quad ,$$

which is intended to be a measure of how much AP j is fit with respect to what user u actually needs. Note in fact the dependence on j. In a similar fashion, user u also evaluates

$$\Gamma_{\mathrm{U}}(u) = \sum_{i=0}^{3} q_i(u) w_i(u) + \max(-w_i(u), 0) \quad ,$$

which summarizes its requirements. Note that our positions combined lead to having $0 \leq \Gamma_{\mathrm{AP,U}} \leq 1$ even for negative weights (we recall that $0 \leq p_i, q_i \leq 1$).

We admit that plainly mixing different physical quantities may be questionable, but the rationale behind the merit figures accounts for it: STAs assess the suitability of visible APs with respect to their own needs. In other words, a broad range of users and

APs is fairly and homogeneously handled. By appropriately weighing their needs, or sampling a suitable distribution, for example, we can simulate real-time or best-effort users. In the same way, a wide variety of access points can be taken into account, as low/high-rate, low/high-delay, cheap/expensive. The model can be further refined, e.g. by adding some penalty due to AP load.

In order to clear the topic yet more, we highlight that $\Gamma_{\mathrm{AP}}(u, j)$ is the measure of the AP j suitability, tailored on user u's priorities, as they are represented by the weight distribution $w(u)$. Being normalized, Γ_{AP} is a proper metric for highly different APs and even in the case of heterogeneous networks (see also the conclusions). On the other hand, $\Gamma_{\mathrm{U}}(u)$ measures the minimum user u's requirements, on the basis of the same weight distribution. Thus, Γ_{AP} and Γ_{U} are defined consistently and can be compared.

Once $\Gamma_{\mathrm{AP}}(j)$ and $\Gamma_{\mathrm{U}}(u)$ are both available for every user u and for every $j \in \mathrm{AP}(u)$, a single *affiliation round* follows. For every $0 \le u \le N_{\mathrm{U}} - 1$, node u selects and affiliates to AP $\tilde{\jmath}$, such that

$$\tilde{\jmath} = \operatorname*{argmin}_{j \in \mathrm{AP}(u)} \Gamma_{\mathrm{AP}}(u, j)$$

and under the constraints $\Gamma_{\mathrm{AP}}(u, \tilde{\jmath}) \ge \Gamma_{\mathrm{U}}(u)$ and that the candidate AP has enough residual capacity, as estimated by the node. No affiliation takes place if no suitable AP is found. Due to the bandwidth consumed by the new user u, and according to the stated approximation, AP $\tilde{\jmath}$ available bandwidth is decreased by the corresponding amount, for example $q_0(u)$. A cooperative strategy comes now into play.

9.2 Cooperative Strategy

Let's recall that the user population is parted in either cooperative (set $\mathcal{C}$) or non-cooperative users (set $\overline{\mathcal{C}}$), and that every STA is in coverage of at least one AP. In general, due to limited resources, not all users could affiliate in the first round detailed above. We denote with $\mathcal{B} \subseteq \mathcal{C}$ the set of unaffiliated cooperative users at this stage.

Following Fig. 9.1, our proposal is that all nodes in $\mathcal{B}$ switch to a *peek mode*, by

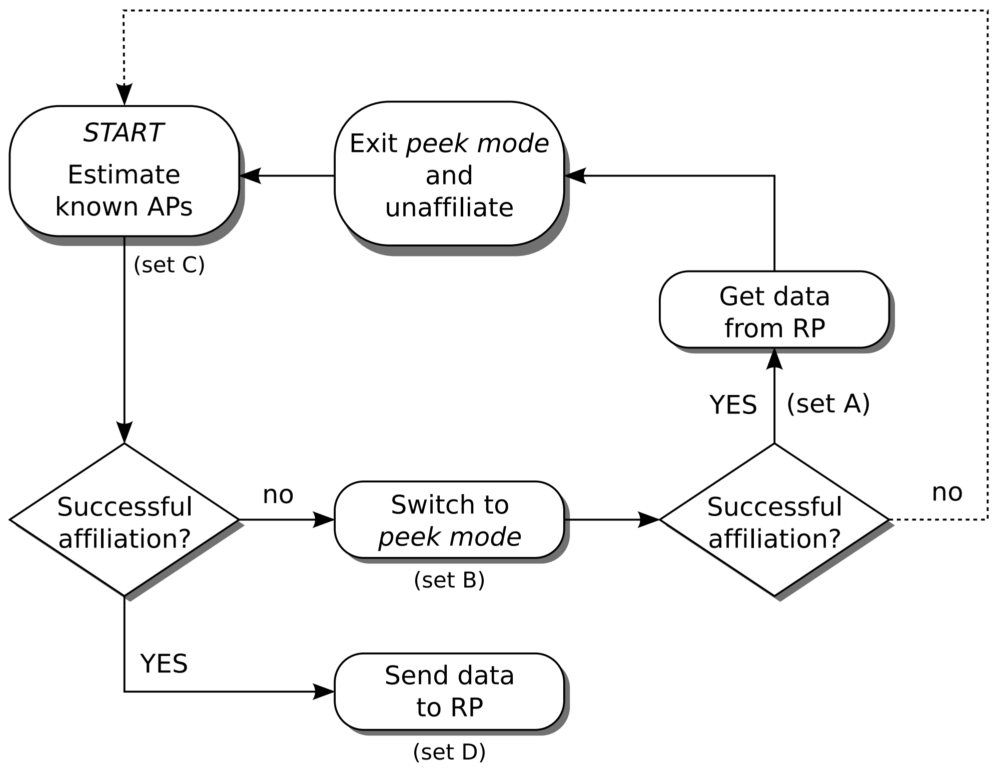


Figure 9.1: Flowchart of the affiliation strategy exploiting user cooperation. Note that simulations currently do not include the dotted path, reported here for completeness.

setting their requested bandwidth to a very small value (such as 0.01 in normalized units) and all others q_i to their minimum (i.e. 0). Given these reduced requirements, they may hope to find some AP to affiliate. They then proceed exactly as per the previous section, that is, they evaluate various Γ_{AP} (which will typically be changed, due to to other nodes' affiliation) and look for a suitable AP. Let's denote with A $\subseteq \mathcal{B}$ such accepted users.

We now introduce in the scenario a remote rendezvous point (RP), acting as an information collector, which can be identified e.g. by its Internet address. Such address must be known in advance by all nodes willing to exploit cooperation. The assumption is reasonable, e.g. users may subscribe to this service.

All users in $\mathcal{C}$ which were able to affiliate either way (i.e. every $u \in \mathcal{D}$, where $\mathcal{D} = (\mathcal{C} \setminus \mathcal{B}) \cup \mathcal{A}$) send RP all their measurement and visibility data. In particular, they send RP a value $\Gamma_{\text{AP}}(u, j)$ for each $j \in \text{AP}(u)$, along with j's identifier. In the simulator, the identifier is j itself, but in real-life it could be e.g. a combination of the AP SSID and MAC address, or some other unique feature.

As far as the AP identification is concerned, we are aware that even combining BSSID and MAC address may not give a worldwide unique identifier. Localizing nodes can then be of help, but we cannot assume GPS availability. Thus, users may be asked to manually identify their position, or they could pinpoint themselves by relying on their public IP address or on *traceroute* output. Another interesting solution is exchanging locally available information (such as AP visibility) among neighboring nodes, in order to infer AP approximate location.

Going back to the allocation strategy, as a next step, affiliated peeking users (i.e. those in $\mathcal{A}$) acquire all data collected by the RP. In the basic hypothesis that they are able to move in coverage of any AP, all $u \in \mathcal{A}$ quit peek mode by restoring their former q_i and a third affiliation round takes place in the same way as before. Note that, differently from the former cases, users in $\mathcal{A}$ have now knowledge of other APs. The whole information exchange is depicted in Fig. 9.2, while a clear view of the overall rationale is depicted in Fig. 9.3.

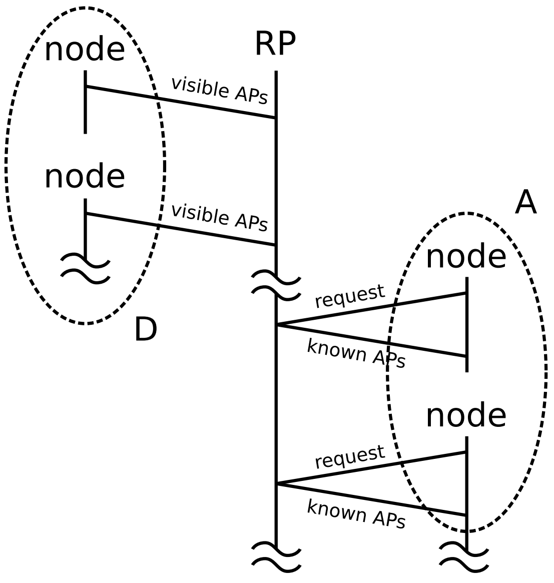


Figure 9.2: Communication model between nodes and the rendezvous point RP. We recall that $\mathcal{A}$ is the set of peeking users, while $\mathcal{D}$ includes both $\mathcal{A}$ and regularly affiliated users (i.e. $\mathcal{C} \setminus \mathcal{B}$).

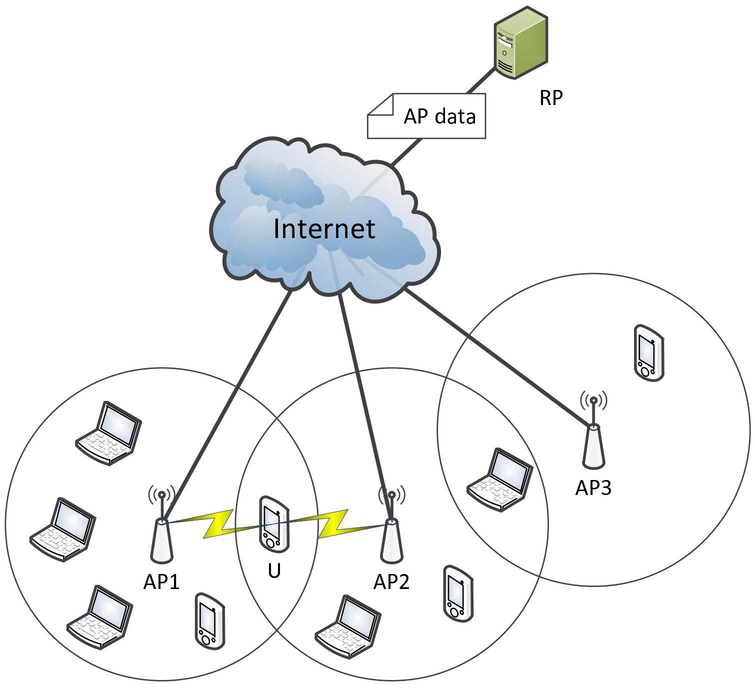
Acknowledgments are presently left out, but can be easily added to the scheme and would not constitute a significant overhead. We also observe that, since the size of the data sent to and from the RP is quite small, it makes sense that the communication can take place with a single packet and rely even on a very low-rate channel, as is the case for the stations in peeking mode, which could be using a very bad channel.

In order to take into account more realistic conditions, we introduce:

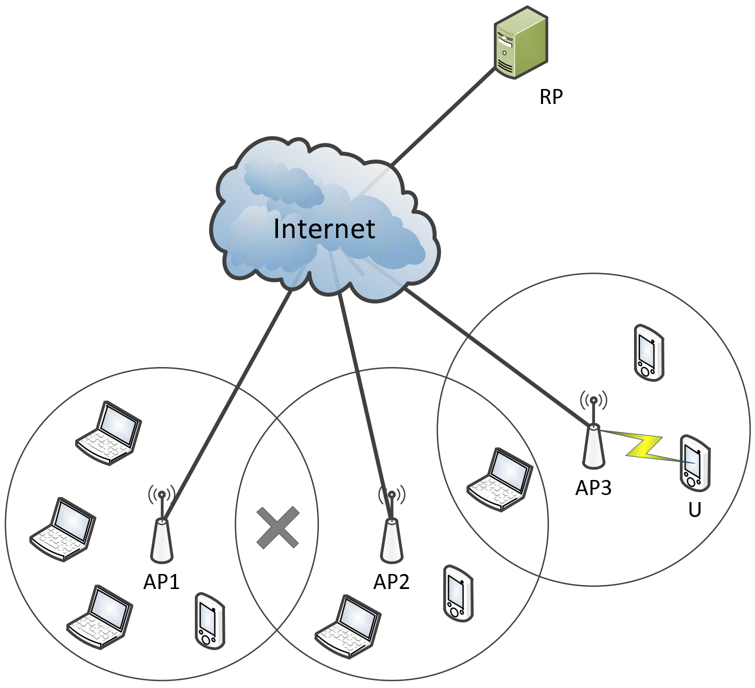
- P as the probability of a correct packet exchange (i.e. the probability with which a node correctly sends RP or obtains from RP the relevant data);
- m as the probability that a user can move in coverage of a given AP.

We are aware that this mobility model may not be adequate in some contexts, but we believe it is nevertheless helpful in having a more insightful view of the topic.

Finally, we note that we are examining a best-case scenario, since we suppose that information are retrieved from the RP after they have been submitted by all the pertinent nodes. Should this be not the case, worse performances are to be expected.



(a)



(b)

Figure 9.3: In (a), the mobile device U is quite unlucky, since it finds itself in a very crowded area, and can't see AP3. (b) After exchanging information with the rendezvous point RP, it discovers and access a new hot-spot AP3, which provides a better service.

9.3 Results

Simulation results are currently constituted by the fraction of affiliated users, with respect to their total number. We highlight that, according to our allocation policy, affiliated users are only those completely satisfied. Due to the strong dependence on the particular affiliation order, many simulations are carried on in a Monte Carlo fashion, so to take into account a very large number of circumstances. The number of runs is ten thousands for single outcome.

The cooperative strategy described above is applied for increasing levels of cooperation, expressed as the fraction of cooperative users. Thus, no cooperation is exploited when this value is zero. The scenario parameters are $N_{\mathrm{AP}} = 60$ and $N_{\mathrm{U}} = 360$. The same results can be found by upscaling these figures, while smaller values may lead to unreliable trends. Without loss of generality, the bandwidth parameters are the only ones taken into consideration, with users requesting 0.15 and each AP offering 1 (all expressed in normalized units). Up to 6 users may thus be allocated to a single AP, allowing to completely saturate the system, should the optimal conditions occur.

Curves in Fig. 9.4 depend on the number of visible APs per user, ranging from 1 to 3 (uniform distribution), while $P = 1$ and $m = 1$. As expected, the availability of more than one choice is significantly beneficial. Moreover, the proposed allocation strategy for cooperative users looks suitable (monotonically increasing with the cooperation level) and highly effective: it is possible to allocate all users in every case.

In Fig. 9.5, $P = [0.5, 1]$ and the number of visible APs is 1 to 3, both uniformly chosen, and again $m = 1$. Since P is the probability of a successful packet exchange, and the proposed strategy involves two packet exchanges, cooperation is actually exploited only for a fraction P^2 of the cooperative users. It is interesting to note that even when cooperation is very heavily impaired, it is anyway appreciably better than no cooperation at all, though errors on the channel impose an upper bound to the overall performance, as lower as the channel is more unreliable.

We investigate the effect of users' mobility (parameter m) in Figs. 9.6 and 9.7. First

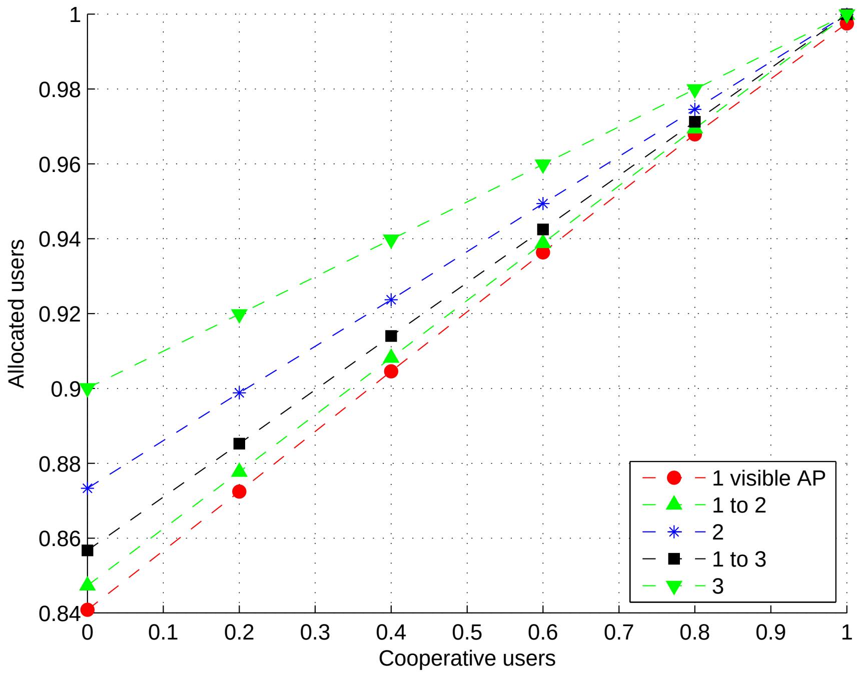


Figure 9.4: Impact of initial access points visibility on overall performance.

of all, the simulated scenario requires just a basic level of mobility for nodes in order to fully exploit a given cooperation level (which may not be enough for saturating the network). The only exception is when all nodes cooperate: in this case, the network can be fully saturated (as possible for this level of cooperation) only if the users are highly mobile.

The same conclusions may drawn by Fig. 9.7, where the focus is on the dependence from the cooperation level. It is clear that the proposed strategy is greatly rewarding even when the probability of moving in coverage of a given AP is as low as one tenth.

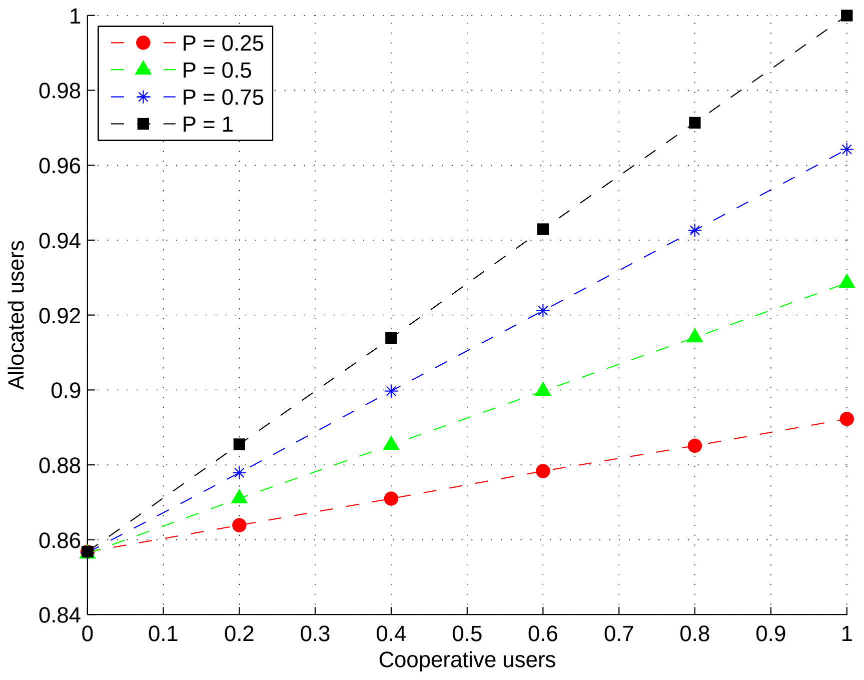


Figure 9.5: Performance as a function of transmission success probability.

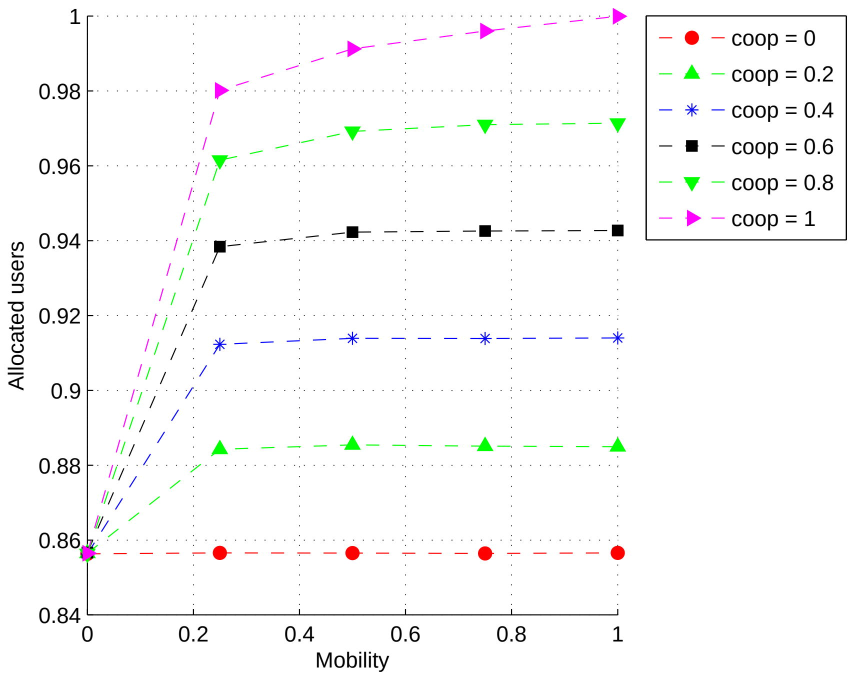


Figure 9.6: Performance as a function of users' mobility.

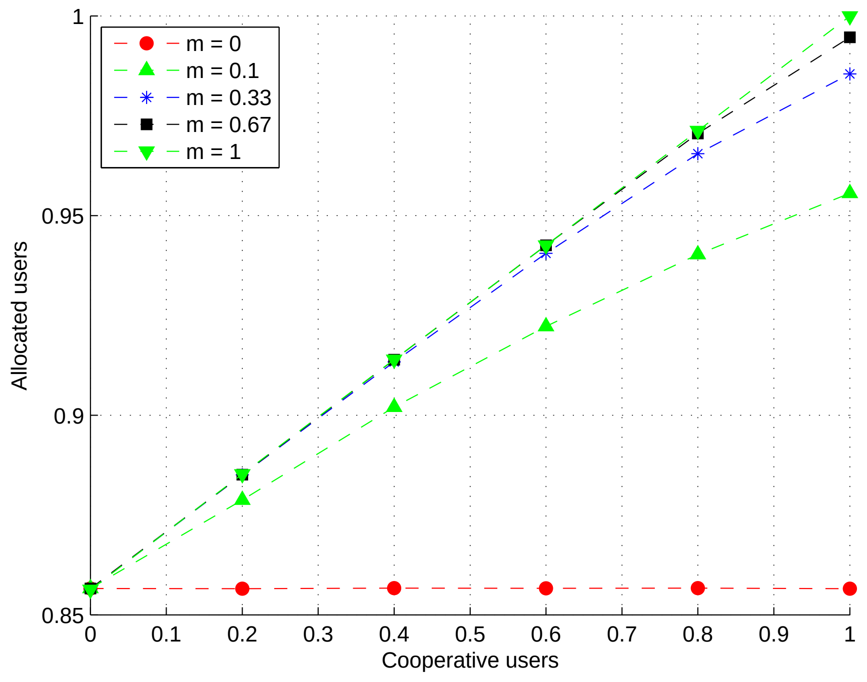


Figure 9.7: The importance of users' mobility for affiliating good APs.

Conclusions

In this second part of the thesis, we followed two lines of thought, both aimed at improving the user's service experience. As far as the pricing strategy is concerned, we investigated the case of a WLAN hot-spot regulated by the CSMA/CA capacity constraint, as in the IEEE 802.11b DCF standard. We applied to this scenario a micro-economic model where users' choices are tuned by means of two knobs: utility and price. From our analysis, it emerges that both the price setting and the number of potential users in the network have a strong impact, and they are also heavily related and imply non-trivial effects, in particular on the throughput and on the total revenue earned by the provider.

Indeed, the pricing policy plays a key role for the correct evaluation of the system performance, since it can tune the number of users accepting the provided service, but the users' sensitivity to this phenomenon can be very high. Even if the number of users is key in achieving an efficient performance, the CSMA/CA-based multiple access exhibits good behavior in multiplexing users' requests in an elastic manner. Further research may be needed to optimize design choices on the basis of the many points of trade-off identified.

More in general, there are inter-dependencies between economic parameters and protocol efficiency, which can be studied through the presented model. A possible application is then useful not only for measurement purposes, but also to search for possible protocol improvements which can be achieved by exploiting economic knowledge.

Shifting to the field of AP selection, our proposal gives good clues on the rewarding nature of user cooperation in wireless networks, with a particular stress on the WiFi case. We point out the immediate applicability of the method, since it does not require any modification on the AP side. However, it does need a remote information collector (which we named rendezvous point, RP) that every cooperative node must know and be able to reach. In our scenario, stations have also to carry on non-trivial estimations and measurements on the AP and connection quality, but the cooperative strategy may be applied regardless of how this information is obtained.

We believe the past sections show that our approach is quite general and could be extended to multi-access networks. Indeed, we do not rely on any hypotheses specifically related to IEEE 802.11x protocols. The only requirement, again, is the availability of methods that allow users to estimate the network quality parameters they are interested in. Points of access with different technologies can then be easily modeled by appropriately setting their respective p_i.

Recent works, like [68–70], investigated these hybrid scenarios, and in particular the coexistence of WiFi and 3G networks. Many strategies for selecting the best association are evaluated, either based on cooperative or non-cooperative principles, and working in a centralized or in a partially distributed fashion. Some form of admission control may be included (the price paid being one). However, in order to work, they have to know the current network state (possibly on a local base) to a great extent, which is not always realistic or feasible and requires special coordination at the infrastructure level.

The underlying principles of our work can be well taken into consideration to deepen the understanding of user cooperation in wireless networks. Further research may focus on the non-trivial problem of unique identification of access points, and on considering more realistic model for user mobility and localization or for resource allocation. Also the limit of the single cooperation pool could be lifted and generalized. An interesting question is whether the presented results can be regarded as universal, e.g. whether they are always good approximations for a given congestion level.

Bibliography

[1] J. Blom, J. Chipchase, and J. Lehikoinen, “Contextual and cultural challenges for user mobility research,” 2005.

[2] D. Tennenhouse and D. Wetherall, “Towards an active network architecture,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 5, p. 94, 2007.

[3] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, “A review of audio fingerprinting,” *The Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 271–284, 2005.

[4] R. Typke, F. Wiering, and R. Veltkamp, “A survey of music information retrieval systems,” in *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[5] P. Cano, E. Batlle, H. Mayer, and H. Neuschmied, “Robust sound modeling for song detection in broadcast audio,” *Proceedings of the 112th AES Convention*, pp. 1–7, 2002.

[6] J. Haitsma, T. Kalker, and J. Oostveen, “Robust audio hashing for content identification,” in *International Workshop on Content-Based Multimedia Indexing*. Citeseer, 2001.

[7] J. Haitsma and T. Kalker, “A highly robust audio fingerprinting system with an efficient search strategy,” *Journal of New Music Research*, vol. 32, no. 2, pp. 211–221, 2003.

[8] H. Özer, B. Sankur, N. Memon, and E. Anarim, “Perceptual audio hashing functions,” *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 1, pp. 1780–1793, 2005.

[9] D. Jang, S. Lee, J. Lee, M. Jin, J. Seo, S. Lee, and C. Yoo, “Automatic Commercial Monitoring for TV Broadcasting Using Audio Fingerprinting,” in *Proceedings of the 29th Audio Engineering Society Conference*, 2006, pp. 38–43.

[10] C. Burges, D. Plastina, J. Platt, E. Renshaw, and H. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 2005.

[11] A. Ghias, J. Logan, D. Chamberlin, and B. Smith, "Query by humming: musical information retrieval in an audio database," in *Proceedings of the third ACM international conference on Multimedia.* ACM New York, NY, USA, 1995, pp. 231–236.

[12] Melodis Corporation, "midomi," http://www.midormi.com, 2008.

[13] V. Venkatachalam, L. Cazzanti, N. Dhillon, M. Wells, M. Inc, and W. Kirkland, "Automatic identification of sound recordings," *IEEE Signal Processing Magazine*, vol. 21, no. 2, pp. 92–99, 2004.

[14] C. Burges, J. Platt, and S. Jana, "Extracting noise-robust features from audio data," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2002*, vol. 1. Citeseer, 2002.

[15] A. Ramalingam, "Gaussian mixture modeling of short-time Fourier transform features for audio fingerprinting," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 4, pp. 457–463, 2006.

[16] C. Burges, J. Platt, and S. Jana, "Distortion discriminant analysis for audio fingerprinting," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 3, pp. 165–174, 2003.

[17] L. Shen, Y. Guan, Y. Wu, and Y. Zhao, "Fast audio fingerprint search strategy for song identification," *Networking and Digital Society, International Conference on*, vol. 2, pp. 259–262, 2009.

[18] N. Orio, "Automatic identification of audio recordings based on statistical modeling," *Signal Processing (Elsevier)*, 2010.

[19] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognition (Elsevier)*, vol. 41, no. 11, pp. 3467–3480, 2008.

[20] S. Sukittanon and L. Atlas, "Modulation frequency features for audio fingerprinting," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 2002.

[21] L. Ghouti and A. Bouridane, "A robust perceptual audio hashing using balanced multiwavelets," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, May 2006.

[22] Y. Liu, K. Cho, H. S. Yun, J. W. Shin, and N. S. Kim, "Dct based multiple hashing technique for robust audio fingerprinting," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 0, pp. 61–64, 2009.

[23] F. Kurth and M. Clausen, "Full-text indexing of very large audio data bases," in *Proceedings of the 110th AES Convention.* Citeseer, 2001.

[24] F. Kurth, A. Ribbrock, and M. Clausen, "Identification of Highly Distorted Audio Material for Querying Large Scale Data Bases," in *Proceedings of the 112th AES Convention.* Citeseer, 2002.

[25] J. Goldstein, J. Platt, and C. Burges, "Redundant bit vectors for quickly searching high-dimensional regions," *Lecture notes in computer science (Springer)*, vol. 3635, p. 137, 2005.

[26] J. Haitsma and T. Kalker, "Speed-change resistant audio fingerprinting using auto-correlation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 2003.

[27] F. Balado, N. Hurley, E. McCarthy, and G. Silvestre, "Performance analysis of robust audio hashing," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 2, pp. 254–266, 2007.

[28] P. Doets and R. Lagendijk, "Distortion Estimation in Compressed Music Using Only Audio Fingerprints," *IEEE Transactions on Audio Speech and Language Processing*, vol. 16, no. 2, p. 302, 2008.

[29] A. Wang, "An industrial strength audio search algorithm," in *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003.

[30] S. Kim and C. Yoo, "Boosted binary audio fingerprint based on spectral subband moments," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, April 2007, pp. I–241–I–244.

[31] D. Jang and C. D. Yoo, "Fingerprint matching based on distance metric learning," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 0, pp. 1529–1532, 2009.

[32] C. Bellettini and G. Mazzini, "On audio recognition performance via robust hashing," in *International Symposium on Intelligent Signal Processing and Communication Systems*, 2007, pp. 20–23.

[33] T. Wang, "Integer hash function," http://burtleburtle.net/bob/hash/integer.html, March 2007.

[34] R. Jenkins, "Hash table lookup," http://burtleburtle.net/bob/c/lookup3.c, May 2006.

[35] T. Li, M. Ogihara, and Q. Li, "A comparative study on content-based music genre classification," in *Proceedings of the 26th International Conference on Research and Development in Information Retrieval.* ACM New York, NY, USA, 2003, pp. 282–289.

[36] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[37] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[38] C. Panagiotakis and G. Tziritas, "A speech/music discriminator based on RMS and zero-crossings," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 155–166, 2005.

[39] C. Bellettini and G. Mazzini, "Reliable Automatic Recognition for Pitch-Shifted Audio," in *Proceedings of 17th International Conference on Computer Communications and Networks*, 2008, pp. 1–6.

[40] Bernsee, "Pitch shifting using the fourier transform," http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/, May 2008.

[41] Audacity, http://audacity.sourceforge.net/.

[42] C. Luna, L. Kondi, and A. Katsaggelos, "Maximizing user utility in video streaming applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 141–148, 2003.

[43] M. Xiao, N. Shroff, and E. Chong, "Utility-based power control in cellular wireless systems," in *IEEE INFOCOM*, vol. 1. Citeseer, 2001, pp. 412–421.

[44] C. Saraydar, N. Mandayam, and D. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 291–303, 2002.

[45] T. Heikkinen, "On congestion pricing in a wireless network," *Wireless Networks*, vol. 8, no. 4, pp. 347–354, 2002.

[46] I. S. 802.11b, "Wireless lan medium access control (mac) and physical layer (phy) specifications. higher-speed physical layer extension in the 2.4 ghz band," in *Supplement to IEEE 802.11 Standard*, 1999.

[47] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: A price-based approach," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 347–364, 2006.

[48] R. Lam, D. Chiu, and J. Lui, "On the access pricing and network scaling issues of wireless mesh networks," *IEEE Transactions on Computers*, pp. 1456–1469, 2007.

[49] F. Meshkati, H. Poor, and S. Schwartz, "Energy-efficient resource allocation in wireless networks: An overview of game-theoretic approaches," *IEEE Signal Processing Magazine*, vol. 24, pp. 58–68, 2007.

[50] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on selected areas in communications*, vol. 18, no. 3, pp. 535–547, 2000.

[51] L. Badia, M. Lindström, J. Zander, and M. Zorzi, "Demand and pricing effects on the radio resource allocation of multimedia communication systems," in *Proceedings Globecom*, vol. 7, December 2003.

[52] L. Badia and M. Zorzi, "A technical and micro-economic analysis of wireless LANs," in *IEEE Wireless Communications and Networking Conference*, vol. 3, 2005, pp. 1632–1637.

[53] F. Calí, M. Conti, E. Gregori *et al.*, "IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement," in *IEEE INFOCOM*, vol. 1. Citeseer, 1998, pp. 142–149.

[54] G. Judd and P. Steenkiste, "Fixing 802.11 access point selection," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, p. 31, July 2002.

[55] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley, "Facilitating access point selection in IEEE 802.11 wireless networks," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement.* Berkeley, CA, USA: USENIX Association, October 2005, pp. 293–298.

[56] Y. Fukuda, A. Fujiwara, M. Tsuru, and Y. Oie, "Analysis of access point selection strategy in wireless LAN," in *IEEE 62nd Vehicular Technology Conference*, vol. 4. Dallas, TX, USA: IEEE, September 2005, pp. 2532–2536.

[57] A. Nicholson, Y. Chawathe, M. Chen, B. Noble, and D. Wetherall, "Improved access point selection," in *Proceedings of the 4th international conference on Mobile systems, applications and services.* Uppsala, Sweden: ACM, June 2006, pp. 233–245.

[58] G. Ganesan and Y. Li, "Cooperative spectrum sensing in cognitive radio networks," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks.* Baltimore, MD, USA: IEEE, November 2005, pp. 137–143.

[59] A. Nosratinia, T. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 74–80, October 2004.

[60] F. Fitzek and M. Katz, Eds., *Cooperation in Wireless Networks: Principles and Applications.* Dordrecht, The Netherlands: Springer, 2007.

[61] G. Fodor, A. Eriksson, and A. Tuoriniemi, "Providing quality of service in always best connected networks," *Communications Magazine, IEEE*, vol. 41, no. 7, pp. 154–163, July 2003.

[62] A. J. Nicholson and B. D. Noble, "BreadCrumbs: forecasting mobile connectivity," in *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking.* New York, NY, USA: ACM, 2008, pp. 46–57.

[63] A. Balachandran, P. Bahl, and G. Voelker, "Hot-spot congestion relief in public-area wireless networks," in *Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on*, 2002, pp. 70–80.

[64] G. Sawma, R. Ben-El-Kezadri, I. Aib, and G. Pujolle, "Autonomic management for capacity improvement in wireless networks," in *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, Jan. 2009, pp. 1–6.

[65] G. Sawma, R. Ben-El-Kezadri, I. Aib, G. Bezalel, and G. Pujolle, "Proactive traffic engineering for IEEE 802.11 mobile wireless networks," in *IEEE 70th Vehicular Technology Conference*, Sep. 2009, to appear.

[66] L. Berlemann, G. Hiertz, B. Walke, and S. Mangold, "Strategies for distributed qos support in radio spectrum sharing," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 5, May 2005, pp. 3271–3277 Vol. 5.

[67] S. Garg and M. Kappes, "Admission control for voip traffic in ieee 802.11 networks," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, vol. 6, Dec. 2003, pp. 3514–3518 vol.6.

[68] D. Mariz, I. Cananea, D. Sadok, and G. Fodor, “Simulative analysis of access selection algorithms for multi-access networks,” in *WOWMOM ’06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 219–227.

[69] D. Kumar, E. Altman, and J.-M. Kelif, “User-network association in an 802.11 WLAN & 3G UMTS hybrid cell: Individual optimality,” in *Sarnoff Symposium, 2007 IEEE*, 30 2007-May 2 2007, pp. 1–6.

[70] K. Premkumar and A. Kumar, “Optimum association of mobile wireless devices with a WLAN & 3G access network,” in *Communications, 2006. ICC ’06. IEEE International Conference on*, vol. 5, June 2006, pp. 2002–2008.

List of Publications

C. Bellettini, G. Mazzini, “A Framework for Robust Audio Fingerprinting”, *Journal of Communications, special issue on Multimedia Computing and Communications*, Academy Publisher, 2010 (to appear).

C. Bellettini, G. Mazzini, “User Cooperation in Access Point Selection”, *IEEE International Conference on Software, Telecommunications & Computer Networks (SoftCOM): Workshop on ICT*, Split (Croatia), September 2009.

C. Bellettini, G. Mazzini, “Scalability Issues in Fingerprinted Audio Searching”, *IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Bangkok (Thailand), February 2009.

C. Bellettini, G. Mazzini, “Reliable Automatic Recognition for Pitch-Shifted Audio”, *IEEE International Conference on Computer Communications and Networks (ICCCN): International Workshop on Multimedia Analysis and Processing (IMAP)*, St. Thomas, Virgin Islands (USA), August 2008.

C. Bellettini, G. Mazzini, “On Audio Recognition Performance via Robust Hashing”, *IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Xiamen (China), November 2007.

S. Blom, C. Bellettini, A. Sinigalliesi, L. Stabellini, M. Rossi, G. Mazzini, “Transmission Power Measurements for Wireless Sensor Nodes and their Relationship to the Battery Level”, *IEEE International Symposium on Wireless Communication Systems (ISWCS)*, Siena (Italy), September 2005.

L. Badia, C. Bellettini, M. Zorzi, “A Radio Resource Management Scheme Driven by Users’ Preferences under the CSMA/CA Capacity Constraint”, *IEEE Vehicular Technology Conference (VTC)*, Stockholm (Sweden), May 2005.