Metadata Independent Hashing for Media Identification & P2P Transfer Optimisation

Jeff Warren, Michael Clear & Ciaran McGoldrick
School of Computer Science and Statistics
Trinity College Dublin
Dublin 2. Ireland
Email: warrenjp@tcd.ie, clearm@tcd.ie, cmcgldrk@cs.tcd.ie

***Abstract*—Swarms within peer-to-peer networks are hindered by content containing incorrect metadata. After publishing, any incorrect metadata requires either a complete republish/swarm recreation or for each peer to manually make corrections (causing them to leave the swarm, decreasing performance). We present an approach which enables a swarm to both collaboratively upgrade embedded data to reflect changes in metadata, and identify additional candidates which contain metadata errors but a correct payload. Swarm degradation due to peer drop-off resulting from edits is eliminated, and additional peers can be identified in a fully automated fashion, increasing swarm lifetime and performance.**

Stemming from this metadata abstraction, automatic purification can be realised in situations where multiple incomplete/incorrect versions are available within one or more unconnected swarms. Variations associated with a content set are processed associatively using a knowledge discovery rule set to extrapolate a canonical tag set, which can also be reinforced using data from external corpora. After any update, these changes can again be automatically disseminated in a peer-to-peer swarm. The system presented enables context-aware P2P data transfers which abstract metadata optimally, while also maximising swarm size and enabling cataloguing of content. A proof-of-concept implementation is presented, and its impact on swarm purification/optimisation is evaluated.

I. Introduction

Accurate identification and recognition of media files remains an unsolved problem. Consumers routinely stream and download audio and video data from the Internet on a daily basis from a variety of sources, yet the majority of this comes with incomplete, incorrect, or simply no metadata. This presents a major obstacle when it comes to organisation, processing, and searching [1]. This paper proposes the application of an existing algorithm used for identifying regular data files, in order to rapidly identify media with high accuracy.

Existing implementations make use of a number of variations on a core method to identify media [2]–[5]. First we will examine some audio oriented solutions - these include Acoustid, Echoprint, last.fm and Open Fingerprint Architecture. The audio file is decoded into uncompressed PCM data, and a Discrete Fourier Transform is performed. This converts the data into a frequency domain representation, from its original time domain representation. Usually some post-processing also takes place, and the resulting output can be used for recognition of the song in an accurate, noise tolerant fashion.

We introduce a completely different method of identification, which draws inspiration from more traditional method of verifying a file is intact and has not experienced any corruption - the hash algorithm. The hash process is quite simple - some input is fed into a function which produces a random, but deterministic output identifier [6]. The input can be split up into evenly sized chunks, or taken as a whole - the former approach can be more attractive if it is necessary to parallelise the computation, or if the payload is likely to be sent in packets over a network.

Until now, hashing as a method of identifying media files has not been very effective, for two primary reasons:

- Popular protocols use chunking approaches, creating a hash for a subset of the file in question, without specifying a standardised chunk size. This means that even an identical file will not be matched without rehashing, since the chunks will not produce the same hash [7].
- Media files, in particular audio files, tend to contain a plethora of metadata [3]. Even if the media data is the same in two versions of an audio track, they will possibly have been tagged or retagged differently according to the information available to the human/machine tagger. Human error will also cause a problem [8], in addition to the fact that many popular metadata encoders do not embed data in full accordance with the appropriate standards. Since the files will now be different, a simple hash check will fail [9].

The first drawback is overcome simply by hashing the entire input data set. A formula which outputs different chunk sizes based on the characteristics of the input (e.g. size) could also be used here. This paper focuses on tackling the second issue, as it is less trivial to overcome. "Metadata independent hashes" are introduced. A media file is examined, and only the stream data is hashed (audio/video streams, in addition to subtitles). The result is a hash which can be matched against differently tagged versions of the same encoded data. This identifier is far more versatile than a blind hash for several reasons. Any retagging, edits to metadata, addition or removal of embedded artwork will not affect the integrity of the stream data, and will therefore not cause an unwanted hash mismatch. At the same time, any transfer error, disk corruption, or other unwanted change of the file will cause a hash mismatch, indicating that the integrity of the item has been compromised.

This approach is format agnostic - it will work on any stream encoded in an existing or future codec. Metadata independent hashes have not previously been implemented for media identification - despite their relative simplicity, existing systems have opted for more computationally expensive approaches.

A. Implications of this work

Users can scan their media collections, and generate a database of hashes for their files, in addition to extracting metadata as appropriate. This information is then uploaded to a central database for examination, as is common with such systems.

Immediately it becomes possible to identify identical files in a person's collection, especially in cases where a badly tagged or untagged version would not have been flagged by conventional methods. It also becomes possible to estimate what percentage of media files available on the Internet are popular, as the central database can track how many different users possess exactly the same version of a media item.

As from an archival point of view, having only a single high integrity version mirrored in many locations is desirable, investigating what percentage of files are highly available and what percentage are not will be a useful contribution. Older P2P networks tended to have many disassociated copies of a single item, with peers tending to have single, relatively short availability sessions [10]. However, recent improved protocols such as BitTorrent have shown that having a single version of a content item available boosts swarm performance and longevity significantly; the work presented in this paper attempts to investigate the model further. The correlation between media items which have single/multiple versions available and how many copies of each are present, if any, will also be investigated. Initially crawling a large media collection (which may be in the order of hundreds of gigabytes, or even terabytes) is an extremely slow process, and thus finding participants for the research proves a challenge. By allowing users access to derived metadata, and a means to embed it back into their collection, there is an incentive to upload data back for further central study. Since the user can also be informed if their files are corrupt, contain illicit/illegal content (provided this has previously been flagged), or if a higher quality version is available from some source, there is further scope for raising participation.

By virtue of these metadata independent hashes being far faster to compute than spectral transform based fingerprinting methods, it becomes more viable for users with large collections to submit a full data set - at present, it can take weeks to process a few hundred gigabytes on a single computer.

In addition to providing media consumers with correct metadata, there are other benefits to be had from a derived central database of metadata. Using multiple submissions of the same items from different users, it will become easy to extract the "correct" value for each appropriate field. Majority rule is expected to be sufficient, but this has yet to be seen. Radically different submissions can be easily flagged, and approved manually if required.

Since a simple rule-based approach can be taken, it can evolve over time to suit edge cases. Existing file taggers are very insular - they may query online databases, but feedback is rarely if ever submitted back. A tagging engine may make a mistake on an edge case which will be noticed by the human user and corrected - but this information is not published for re-use. A centralised database, such as the one proposed by this research, allows for feedback into the rule-based algorithm, resulting in improvements which can then be felt through the whole network. Previous studies have proposed the use of peer-to-peer based metadata storage [11], however a traditional centralised approach has proved superior.

Future plans would see the system upgraded to allow for a hybrid identification approach - in cases where hashing / comparison of metadata could not be used to match files, audio fingerprints could be generated in the conventional, slower way.

A centralised database can also be used to flag fake or illegal versions of media files, allowing users to purge them from the Internet [12].

Using hashes for accurate identification is superior to intelligent fingerprinting for several reasons - the algorithms are simple, fast and portable, and on modern hardware are bound only by the rate at which the data can be read off disk [6]. By comparison, short audio files may take several seconds to fingerprint, and the resulting identifier will be significantly larger than a simple hash.

Treating a file in this way is also useful for transferring it or distributing it among a large audience. One problem noted in Peer-to-Peer file transfer implementations is that files are treated blindly - there is no awareness of what the content is [9]. If media files are identified at publishing time, this can be leveraged usefully. Metadata and header information could be completely abstracted - i.e. only stream data would be transferred. On the receiving end, an appropriate container can be constructed, and the stream data fed in as it is received. Finally, metadata, from some other source, could be added back to the file. The result is a published file which is effectively version-controlled. Stream data will not change, but updates to the metadata can easily be distributed after publication. Users who downloaded a file could have their client check periodically for updates, and have their metadata altered without needing to supervise the process.

Building on top of this system, an application has been implemented which allows context-sensitive transfer of MP3 files in a peer-to-peer swarm, leveraging the benefits described above. Stream data (i.e., data containing the audio stream only) contained in a file is extracted, tagged with sequence information, and disseminated in a similar fashion to BitTorrent. Upon completion of a transfer, a downloading peer will generate a new MP3 file containing the data, and no metadata. This missing metadata can be embedded either by querying one or more peers in the swarm, or by requesting from a canonical authority using its identifying hash.

II. Background

This section examines the Discrete Fourier Transform (DFT) approach used by many existing audio identification systems, and discusses the advantages and disadvantages that go along with it.

Computing a DFT using a Fourier transform method is a computationally expensive operation [2]. For the most useful output to be computed, a DFT must be generated from the entire input dataset - i.e. an entire song. The output is a frequency-variant representation (the input being time-variant). This can be matched against the output from another audio file, usually taking into account a similarity threshold, which allows for the presence of noise in one of the two files.

Comparing media items in the frequency domain is useful as it is strictly not necessary to process the entirety of both files - a subset may be enough to give an accurate enough representation for matching. The resolution of the generated spectrogram can also be reduced for storage, useful when hundreds of thousands of items must be archived for a match-against library.

Identification of video streams is quite similar to the process described above. Keyframes (full frames) can be extracted out of a video stream, and put through a similar transforming function. Then, keyframes from different versions of a video can be matched up [13]. Further matching can be performed to ascertain whether the video streams are equivalent, or if one simply contains a clipping of another. While audio processing can be performed far faster than real-time, fingerprinting of thousands of tracks is still a slow process. Video streams are slower again, as image processing over a clip of several minutes or even hours is an expensive task.

Aside from content identification and intellectual property law enforcement, the vast majority of the work in this domain has been undertaken for the purposes of detecting, correcting and adding meta data associated with audio files. Traditionally video files do not embed any identification, apart from anything contained in the file name, excepting subtitle language. No authoritative system has superseded the plethora of different approaches, hence tagging applications often try to make use of disjoint algorithms concurrently. This is because each approach usually has its own fingerprint library associated with it for lookup - and the identifiers are incompatible. MusicBrainz, a popular MP3 tag correction application, is a good example of this trend [14].

A traditional fingerprinting library will return the most similar track it matches, along with a certainty rating. This leeway has advantages and disadvantages. The primary advantage is that streams subject to corruption, artifacting caused by compression, range compression, or clipping introduced by the copying or mastering processes can be matched. There are several disadvantages - audio tracks on DJ mixes tend to be overlapped on top of each other, which confuses the spectral patterns, and sabotages the certainty ratings by a large amount. Extended mixes and shortened versions of audio productions will still match together, which is not always desirable when pursuing completely accurate meta data.

In addition, there is one overhanging concern when designing a media implementation system - the accuracy / resource trade-off of the library. The higher the resolution of a fingerprint, the more disk space it requires to store as part of the library. Some work has been done to optimise the usefulness of a fingerprint, while minimising size [15], though compared with a hash, spectral fingerprints are still not ideal. Higher resolution fingerprints also require more CPU time to match against queries. If the resolution is reduced too much, though, the higher the chance of a mismatch, and the reliability of matches will also suffer. Since our solution only requires the storage of a number of SHA-1 hashes, which are computationally trivial to match against, this compromise need not be made.

III. Methodology

A media file cannot simply have headers and metadata blindly stripped from it. Metadata is littered throughout many containers, particularly with MP3. There are potentially different tags in the header of the file, stored at the end, and even inside the stream data. If this is not all handled correctly, then the resulting data hash will not be fully metadata independent. As a desirable property of a hash algorithm is that a one bit flip in the payload will cause a random selection of half of the output hash's bits to flip, even the smallest error when identifying metadata will damage the hash integrity, leading to mismatches. This would effectively render the system useless for our purposes.

In addition to this delicate problem, a useful system would need to support all of the common containers, and be able to semi-intelligently cater for the quirks associated with each. Each media file is opened and treated as a media file. Headers are processed, and each stream in the file is identified. A SHA-1 hash is identified for each stream, and can be used to uniquely identify that stream. To compute an overall identifier for the file as a piece of media, the collection of hashes is

XOR'ed[1] together. While a synchronised hash dependent on the packet order could be computed, it would be extremely wasteful to recompute this for big files. If certain streams are identified as dummy, non important, or containing metadata, it is trivial to exclude them from the XOR process, even at a later stage.

The output a unique identifier for a given media file, as well as inner identifiers for each component stream. The latter can be used to detect the presence of identical streams inside otherwise different files (e.g. two copies of a video clip which have identical encoded audio, but different resolution video). If the source file is retagged (the metadata is changed) a rehash will now yield the same result.

Since a lot of media is highly redistributed, matching these hashes against a database can immediately provide a reference point

IV. Implementation

An initial approach taken was to attempt parsing an MP3 file and simply zeroing or ignoring all meta data. Although easy to implement, due to various reasons, it was useless. Firstly, fields padded with spaces, or superfluous nulls, were impossible to cater for - two matching files might have been tagged differently in this regard. Secondly, identifying every possible metadata field in every file would have required implementation of every desired format. Since an eventual goal was identification of a most correct set of meta data for a single file, every unsupported format reduces the potential quantity of collectable data. In addition to this, many multimedia containers are badly documented, badly implemented by the software with generates them, and often ambiguously standardised. Failure to cater for every potential quirk would result in an erroneous hash mismatch - something which it is crucial to avoid.
The MP3 container is a good example of this - unofficial extensions, such as the Xing and Lame standards, insert additional meta data into a file inside a fake stream frame located near the beginning of the file. Legacy decoders simply play a few unnoticeable milliseconds of noise, and newer players which are aware of them can skip over them. Including them for our purposes would mangle the results - and this is but one example of many. It followed that the only useful way to proceed was to treat the files as stream containers - extracting the stream data out of the file, rather than disregarding the meta data.

The FFMPEG library, which incorporates container interpreters for most common multimedia formats [16], is firstly used to identify the codecs which were used in creating a file. This can be important, since quite often a filename's extension cannot be relied on as accurate. Next, the streams are identified, and the data is streamed out. Streams are packetized, and inside a multi stream file, different stream packets are ordered serially in roughly the order they should be needed. A video file will almost always have an accompanying audio track synchronised to it. Given the design of rotational storage devices, and the performance penalty associated in seeking across a disk, thus interleaving frames from different streams in a temporally suitable fashion was sensible. Each contains an identifier, which is used to pass them to their respective decoders upon playback.

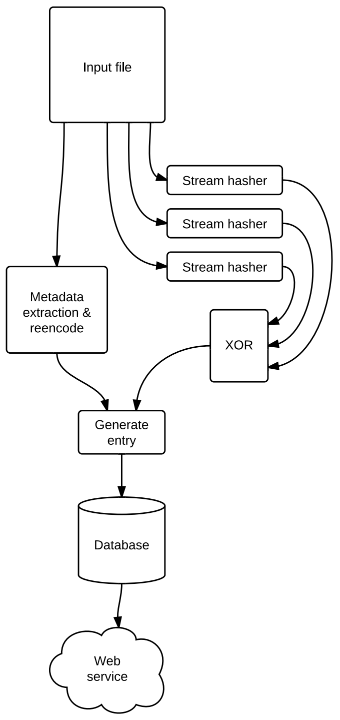

Fig. 1. High-level System Layout

Two different files could be equivalent - i.e. contain byte-for-byte identical streams. However, if they were interleaved differently at mux time, again a hash mismatch would be possible. As a result, and because it is also useful to uniquely identify streams within a file, we hash each different stream independently. The resulting set of hashes are combined to yield a hash deterministically identifying the stream collection.

The resulting hashes are stored into a local database, along with any available metadata contained within the files. Additional information regarding the file's modification and the hasher's version allow for future version control and rescanning of files which are modified externally between scans.

The system currently uses the SHA-1 algorithm, although any other algorithm could easily be switched in. SHA-1

[1]The authors are aware of the implications of this approach with respect to cryptographic strength. Although XOR does not upset the uniformity of the distribution, care must be taken that two streams are not identical, as their existence would be eliminated from the result. Moreover, the ordering of the streams is naturally ignored as well. Although there are several secure ways to combine hashes, XOR was chosen for efficiency because collision resistance against an attacker was not a primary design goal.

was selected on a "good enough" basis - it is quite fast, and resistant to practical collision attacks. It is essential that files are handled correctly - any bugginess in the hashing algorithm, such as treating a placeholder frame as stream data, or any buffer overflow between streams, would result in an inaccurate hash. The consequences of this would be that all existing data would have to be marked invalid, and each user would require a full rescan, which is extremely undesirable.

The resulting fingerprint is stored as a string - lookups only require a computationally trivial string comparison. Rival approaches require some or all of the file to be computed using a variation of a FFT, with the result sent over a network, and compared with the remote database's fingerprint collection. The higher accuracy required, the higher the resolution the spectral must be, and thus the slower the file transfer (network bandwidth) and comparison (CPU time) will be. Conversely, hash based matches are binary, they either match as identical or must be treated as different. Thus, any file sourced corruption is highly undesirable.

Finally, the local database is synchronised with a remote, centralised system. All information, from file path (important, the lack of embedded support for metadata in many containers forces archivists to record data in a hierarchy of folders/filenames) to stream information and embedded metadata is sent, and associated with an identifying user account. Embedded artwork, common in music files, is also uploaded.

We now have a system which can open almost any multimedia file, and separate metadata and stream data. It proved possible to serialise stream data out separately, hash it, and derive a matching hash back - thus proving that metadata independent hashes can be used for identification, and that stream based multimedia files can be transferred in a swarm based context sensitive fashion. A prototype implementation was used to demonstrate this secondary hypothesis, modelled on the BitTorrent paradigm. A "tracker"-like coordinator allowed audio files to be published to it, using the hash only. No other data about the file needs to be known by the coordinator. Peers, including one or more with the original file, connect to the tracker, and download lists of other peers (IP addresses and associated ports) who also have the file of interest. They then interconnect, similarly to BitTorrent [17], and download stream packets from each other. When a downloader has successfully obtained all packets from other peers, they are serialised back out to disk. The prototype queries a web service using the identifying hash, and receives all embeddable metadata, as well as an appropriate file name. This is then tagged in appropriately.

V. Deriving data

The major advantage of abstracting the metadata to a central store is that it can be error corrected. Many different versions of the same stream data often carry different versions of the metadata which should be associated with them - some fields are incomplete, truncated, missing, or simply incorrect. Frequent inconsistencies include capitalization, spelling and word order [18], [19]. The metadata authority can combine submissions from many different sources and attempt to extract a verified canonical set of data for each file.

Currently, for a given datum, the best candidate is chosen by majority rule. For every unique file submitted, the entries for all users are examined, and from all non blank entries, the most frequently occurring candidate is selected. This can easily be customised per datum - for example, when processing the title for an audio track, a customisation could be made which preferred candidates containing spaces where others contain underscores (a common artifact). Spell checking, or illegal character detection could also be considered. Invalid entries could be ignored; for example a non integer entry in a "track number" or "release year" field could be disregarded. The extracted canonical data is saved separately to the user submissions, and can then be pushed out to scanner clients for embedding back into library files. A key point to note is that if an extracted entry is flagged as erroneous, or a missing piece of data is later added, the updated canonical version can easily be pushed out to clients next time they synchronise. The alternative is each client manually tagging their files, a process which has proved to be too tedious to carry out by most collectors.

Unavoidably, any form of corruption affects the integrity of the data set. Corruption commonly occurs due to storage device error, network transmission error, and software-driven error. CRC check failures, line noise, and faulty tagging software respectively are some examples. Files may also have been "double encoded", or re-encoded in such a way that fidelity is lowered [20]. Users can manually remove such files from their collections, which will result in them being synchronised out of the database, or alternatively they can flag them for blacklisting on the centralised authority.

Entries are mapped in such a way that multiple unique streams can point to the same metadata. This allows multiple legitimate versions of the same multimedia item to share metadata, thus increasing the likelihood of having a complete, correct entry. Detecting legitimate cases for combination is computationally expensive and prone to error, and is only automatically considered when the probability of it being a true positive is extremely high. The current strategy requires a fully matching "artist", "album" and "track number" data set.

Any other missing data can be corrected/added externally in two fashions. Firstly, human users can use an online tool to edit tags associated with their collection on the metadata authority server. Upon synchronisation, these changes can then be pushed out to their local collection. Secondly, existing external corpora can be leveraged to incorporate missing data. Some examples of suitable corpora would be last.fm, Gracenote and Discogs. Combining external and internal data made it possible to rapidly build a consolidated canonical corpus with extremely high correctness/accuracy. Relevant artwork can also be imported from external sources, and low resolution submissions can be automatically replaced with higher resolution candidates.

VI. Findings & Experimental Results

Presented below are some benchmark results, and statistics regarding data collected thus far. Some performance data about the implementation is given, which should clarify the advantage of this approach over traditional fingerprinting.

A. Performance

Some input data sets were used to benchmark throughput. The host machine contains an Intel Core i7 CPU @ 2.8GHz (HyperThreading disabled), and both an SSD and standard rotational hard disk. Sequential filesystem read throughput tests were run on the two devices to determine what the performance ceiling should be for the system. A data set of 4.6GB in 675 MP3 files was used for the audio benchmark, with 4.6GB in 4 MKV files for the video.

	Filesystem	Audio scan	Video scan
SSD	202.9 MB/sec	40.4 MB/sec	90.7 MB/sec
HDD	83.0 MB/sec	35.1 MB/sec	31.5 MB/sec

TABLE I
THROUGHPUT BENCHMARKS FOR THE MEDIA SCANNER, WITH REFERENCE FILESYSTEM VALUES

Performance on the data set read from the SSD is noticeably faster, especially with the video files, which contain no explicitly tagged data (TableI). The files are read sequentially back, thus the increase in throughput cannot be attributed to the lack of a seek penalty on the SSD. Though the system was thought initially to be I/O bound, the fact that files are only read or processed at a given time is a limiting factor. A future alteration to allow files to be streamed in by a reader thread, and then processed (possibly several at once) by a pool of worker threads could yield significantly better timings.

The disparity between the audio and video gains the SSD has on the HDD are attributable to the fact that video files have no (explicit) tagged meta data. The specification of the MP3's tags means that searching around inside the file for extraction is relatively expensive. It is difficult to draw a comparison between this approach and other meta data tagging systems. The quantity of data which must be collected for a reference database means that establishing a new system requires significant investment. At time of writing no similar research systems have been made available. However, two commercialised pieces of software have become well known for this task - MusicBrainz [14], and MediaMonkey. The former was able to identify the test audio data set in just over 12 minutes (6.3MB/sec), and the latter could not be compared, as it forces the user to manually step through each song and verify that the identity assigned is correct. In addition, neither perform any verification, so it cannot be guaranteed that the tracks are intact. Video files are also not supported by either system.

B. Submitted data

Forty volunteer users were asked to use the tool to scan their video and audio collections. While far more would be needed to draw realistic conclusions as to how many duplicate copies of files exist in different libraries, it is evident that there is significant crossover. This will likely increase as more submissions are made from future users.

In total, 51.04TiB of data was scanned - user library size ranged from 6.5GiB up to 7.09TiB. This comprised 771,209 files. The mean number of per-user submissions was 25,707.

	Total	Video	Audio
Submissions	771,209	69,683	701,526
Unique submissions	634,595	54,502	580,093
Duplicate submissions	136,614	15,181	121,433
Duplicate submissions (%)	17.714%	21.785%	17.309%

TABLE II
STATISTICS REGARDING DUPLICATE SUBMISSIONS

Examining just 40 libraries, all independently sourced, we can see that there is a significant portion of duplicate files. In cases where meta data does not match between duplicate files, it would be possible to correct, or at least flag the errors, allowing them to be semi or fully automatically corrected.

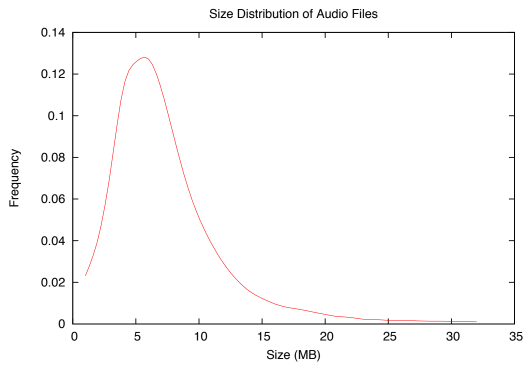

Fig. 2. Size Distribution for Audio Files

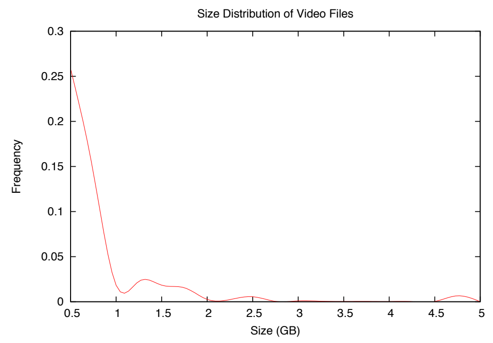

Fig. 3. Size Distribution for Video Files

The data examined is a representative cross section of media currently available in the wild. The majority of audio data is sized between 0 and 16MB, and is normally distributed, as can be seen in Figure 2. This is attributable to the fact that most users favour lossy encodes, using codecs which perform similarly. A small amount of audio files are radically different, and may be in the order of 100-200MB, usually representing a substantial live recording in one track. No such pattern emerges in video file sizes, however - certain size intervals, which correspond with rules, guidelines, or other factors related to video encoding, contain almost all submitted data (Figure 3). Some examples would be 175MB, 350MB, 700MB (standardised sizes which fill a 700MB CD), 4.3GB and 8.3GB (standardised sizes which fill a DVD), as well as 1.2GB and 2GB (sizes which are believed to give optimal results for a 44 minute TV episode in high definitions using the most efficient codecs). A relatively small number of other video files which do not fit into these known ranges of interest are also present.

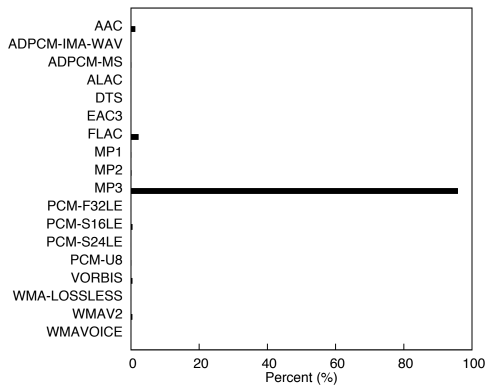

Fig. 4. Frequency of Audio Codecs

The distributions of codecs used to create files in the users' libraries are also worth mentioning, as they indicate which are currently popular among media authors. For audio, MP3 remains the most popular, with 95.7% of files examined using it (Figure 4). Its only competitor, AAC, ranked second among lossy codecs with 1.13% of files being encoded using it. This suggests that even though AAC and Vorbis have been shown to be higher performance codecs, the existing prevalence of MP3 has made it hard to dislodge. FLAC, a popular lossless codec, was used for 2.1% of audio files.

A codec monopoly in video files was less evident - 61.5% of files were encoded using MPEG4, and 23.1% with h.264 (Figure 5). This reflects that while h.264 has been embraced as the current best choice for video distribution, a large quantity of media using MPEG4 still exists in user collections.

C. Disagreement in Metadata

It was found that of the distinct audio tracks processed by the system, approximately 17.309% were associated with

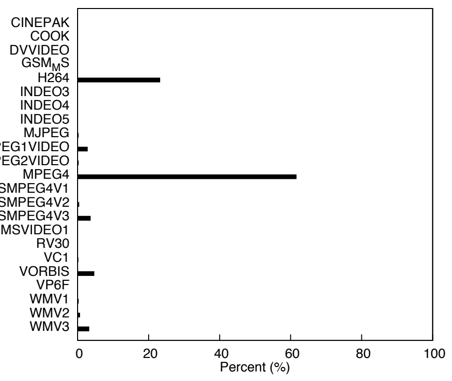

Fig. 5. Frequency of Video Codecs

more than one user. It is interesting to examine the disparity in metadata between the copies of these shared files. The focus was on MP3 as such files constitute the majority of submissions from users, and their ID3 tags are widely used. A sizable proportion of the set of shared tracks had at least two copies with different metadata. A distribution capturing the number of disagreements per tag is shown in Figure 6.

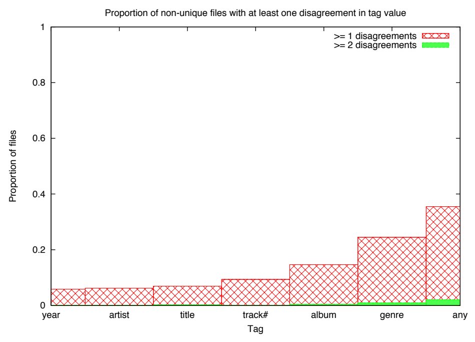

Fig. 6. Degree of disagreement of tags

It would be useful to investigate whether the results in Figure 6 are representative of a larger data sets. The strategy for pointing multiple unique streams to the same metadata proposed in Section V (based on matching Artist, Album and Track Number tags) may thus not be satisfactory - especially since it seems that track numbers are missing from a large percentage of files.

VII. Conclusions & Future Work

The systems detailed in this article have demonstrated that it is both possible and advantageous to leverage media specific context awareness when disseminating files across a peer to peer network, and reconstruct them at the receiving end. Files can both arrive intact, anonymously, and embed highly accurate metadata.

We have also demonstrated how a centralised authority for media metadata can, in a semi- or fully-automated fashion, calculate correct metadata based on multiple user submissions, as well as external sources. Media which has been previously processed by the system can be identified as matching with 100% accuracy, and requires a trivial identifier to do so. Disadvantages include the fact that a media source, extracted almost identically by two different human users on differing hardware, are unlikely to match byte-for-byte, and must be combined. Fortunately, the vast majority of media files possessed by users are encoded by a relatively small subset - i.e. most files have multiple copies. We have also demonstrated how effectively identical variations can be handled and combined.

Traditional approaches require significantly more computational power both on the user and server sides, in addition to bandwidth. Hash based matching requires a tiny fraction of the resources, and as a result provides a better user experience, making it possible to verify and optimise a large media collection in mere hours.

Extracting canonical metadata centrally also proved to be optimal - the data yielded edge cases which had not been considered, and would only become apparent when combining such large quantities of data, something which would never occur on a per-user level. The extraction algorithms can be edited on the central authority, and no user software update rollout is needed. Changes to their files can simply be pushed out upon synchronisation. Region-specific data, special cases and temporal exceptions could also be integrated if required.

Future work includes collection of significantly more data than was used to draw the conclusions in this report, and in addition, data extraction algorithms based on an expanded data set could be optimised. One possible additional use for the corpus presented would be as a queryable database for decoding software. Users could be presented with online data for their media as it is played. A suggested testbed for this would be a HTPC solution such as Boxee or XBMC [21], [22]. Future work will also include an examination of the security implications of the proposed system, in particular user privacy and prevention against injection of rogue data.

References

[1] P. Cudré-Mauroux, A. Budura, M. Hauswirth, and K. Aberer, "Picshark: mitigating metadata scarcity through large-scale p2p collaboration," *The VLDB Journal*, vol. 17, no. 6, pp. 1371–1384, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1007/s00778-008-0103-4

[2] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *J. VLSI Signal Process. Syst.*, vol. 41, no. 3, pp. 271–284, Nov. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11265-005-4151-3

[3] W. Li, Y. Liu, and X. Xue, "Robust audio identification for mp3 popular music," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '10. New York, NY, USA: ACM, 2010, pp. 627–634. [Online]. Available: http://doi.acm.org/10.1145/1835449.1835554

[4] S. Zmudzinski and M. Steinebach, "Psycho-acoustic model-based message authentication coding for audio data," in *Proceedings of the 10th ACM workshop on Multimedia and security*, ser. MM&Sec '08. New York, NY, USA: ACM, 2008, pp. 75–84. [Online]. Available: http://doi.acm.org/10.1145/1411328.1411343

[5] B. Zhu, W. Li, Z. Wang, and X. Xue, "A novel audio fingerprinting method robust to time scale modification and pitch shifting," in *Proceedings of the international conference on Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 987–990. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874130

[6] C. Teat and S. Peltsverger, "The security of cryptographic hashes," in *Proceedings of the 49th Annual Southeast Regional Conference*, ser. ACM-SE '11. New York, NY, USA: ACM, 2011, pp. 103–108. [Online]. Available: http://doi.acm.org/10.1145/2016039.2016072

[7] H. Pucha, D. G. Andersen, and M. Kaminsky, "Exploiting similarity for multi-source downloads using file handprints," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973432

[8] T. Vainio, K. Väänänen-Vainio-Mattila, A. Kaakinen, T. Kärkkäinen, and J. Lehikoinen, "User needs for metadata management in mobile multimedia content services," in *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, ser. Mobility '09. New York, NY, USA: ACM, 2009, pp. 51:1–51:8. [Online]. Available: http://doi.acm.org/10.1145/1710035.1710086

[9] M. Wojciechowski, M. Capotă, J. Pouwelse, and A. Iosup, "Btworld: towards observing the global bittorrent file-sharing network," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 581–588. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851562

[10] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," 2002.

[11] G. Tzanetakis, J. Gao, and P. Steenkiste, "A scalable peer-to-peer system for music information retrieval," *Comput. Music J.*, vol. 28, no. 2, pp. 24–33, Jun. 2004. [Online]. Available: http://dx.doi.org/10.1162/014892604323112220

[12] N. Michalakis, R. Soulé, and R. Grimm, "Ensuring content integrity for untrusted peer-to-peer content distribution networks," in *Proceedings of the 4th USENIX conference on Networked systems design & implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 11–11. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973441

[13] J. Zhou and X.-P. Zhang, "Automatic identification of digital video based on shot-level sequence matching," in *Proceedings of the 13th annual ACM international conference on Multimedia*, ser. MULTIMEDIA '05. New York, NY, USA: ACM, 2005, pp. 515–518. [Online]. Available: http://doi.acm.org/10.1145/1101149.1101265

[14] A. Swartz, "Musicbrainz: A semantic web service," *IEEE Intelligent Systems*, vol. 17, no. 1, pp. 76–77, Jan. 2002. [Online]. Available: http://dx.doi.org/10.1109/5254.988466

[15] C.-C. Liu and P.-F. Chang, "An efficient audio fingerprint design for mp3 music," in *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*, ser. MoMM '11. New York, NY, USA: ACM, 2011, pp. 190–193. [Online]. Available: http://doi.acm.org/10.1145/2095697.2095732

[16] S. Tomar, "Converting video formats with ffmpeg," *Linux J.*, vol. 2006, no. 146, pp. 10–, Jun. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1134782.1134792

[17] B. Cohen, "Incentives Build Robustness in BitTorrent," 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.1911

[18] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-Based Music Information Retrieval: Current Directions and Future Challenges," vol. 96, no. 4, Apr. 2008, pp. 668–696.

[19] A. Freed, "Music metadata quality: a multiyear case study using the music of skip james," in *AES 121*, San Francisco, CA, 2006. [Online]. Available: http://cnmat.berkeley.edu/publications/music_metadata_quality_multiyear_case_study_using_music_skip_james

[20] M. Qiao, A. H. Sung, and Q. Liu, "Revealing real quality of double compressed mp3 audio," in *Proceedings of the international conference on Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 1011–1014. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874137

[21] "Boxee personal media streamer," http://boxee.tv, Mar. 2012.

[22] "Media center solution for windows, osx, linux," http://xbmc.org, Mar. 2012.