Distributing Workflows over a Ubiquitous P2P Network

Eddie Al-Shakarchi[1†] Pasquale Cozza[2]

Andrew Harrison[1] Carlo Mastroianni[3] Matthew Shields[1]

Domenico Talia[2] Ian Taylor[1 4]

[1] School of Computer Science, Cardiff University, UK

[2] DEIS University of Calabria, Rende (CS), Italy

[3] CNR-ICAR, Rende (CS), Italy

[4] Center for Computation and Technology, Louisiana State University, USA

[†] Corresponding author

Eddie Al-Shakarchi E.Alshakarchi@cs.cardiff.ac.uk

Tel: +44 (0)29 2087 4812

Fax: +44 (0)29 2087 4598

Cardiff School of Computer Science, Cardiff University, Queen's Buildings, 5 The Parade, Roath, Cardiff CF24 3AA, UK

Abstract

This paper discusses issues in the distribution of bundled workflows across ubiquitous peer-to-peer networks for the application of music information retrieval. The underlying motivation for this work is provided by the DART project, which aims to develop a novel music recommendation system by gathering statistical data using collaborative filtering techniques and the analysis of the audio itself in order to create a reliable and comprehensive database of the music that people own and which they listen to. To achieve this the DART scientists creating the algorithms need the ability to distribute the Triana workflows they create, representing the analysis to be performed, across the network on a regular basis (perhaps even daily) in order to update the network as a whole with new workflows to be executed for the analysis. DART uses a similar approach to BOINC but differs in that the workers receive input data in the form of a bundled Triana workflow, which is executed in order to process any MP3 files that they own on their machine. Once analysed, the results are returned to DART's distributed database that collects and aggregates the resulting information. DART employs the use of package repositories to decentralise the distribution of such workflow bundles and this approach is validated in this paper through simulations that show that suitable scalability is maintained through the system as the number of participants increases. The results clearly illustrate the effectiveness of the approach.

1 Introduction

This paper describes a novel Music Recommendation System (MRS) that employs the use of the underlying Distributed Audio Retrieval using Triana[1] (DART) peer-to-peer (P2P) subsystem in order to provide a fan-out mechanism for distributing workflows across the network and for retrieval and aggregation of results. Mrs Dart utilises a combination of distributed systems technologies ranging from Grid computing, peer-to-peer, Web Services and workflow. The UK Science and Technology Facility Council PIPSS [2] (PPARC Industrial Programme Support Scheme) funded project at its core is based on the volunteer computing paradigm that typically employs the use of home computers for the analysis of data, e.g. radio-telescope data for SETI@home[3], gravitational wave data for Einstein@home[4], and so on. In the DART scenario however, the home users, or workers as they are referred to, perform analysis of their own music collection by executing Triana workflows that encompass the analysis to be performed at that time.

It is crucial to the project that the DART environment is capable of self-updating as it is intended not only to provide music recommendations for end users but also to establish itself as a research platform that music information retrieval (MIR) scientists can use in order to test out new ideas in the MIR field. To this end, scientists will want to design new methods for audio analysis, both for statistical correlation based on tags and more interestingly for the analysis of the actual audio itself to extract things like tempo, pitch, mood and so forth. We believe these *high-fi* parameters will allow DART to stand out from the existing recommendation systems that generally are based on grouping what people have bought, and do not take into account the actual audio nor do they take into

[1] see MRS DART website `www.mrsdart.com`

[2] see `http://www.pparc.ac.uk`

[3] see SETI@Home website `http://setiathome.berkeley.edu/`

[4] see Einstein@Home website `http://einstein.phys.uwm.edu/`

account what people actually have and what they have listen to. Further, we intend to build a DART community, rather similar to those hosted on Web sites like mySpace and Facebook, to allow scientists to share workflows, which will help to create a collaborative research space to help advance the state of the art in this area. Such an environment is being developed through the WHIP[5] project, based around tools developed in the myExperiment project [8].

In order to distribute such worflows, a complex uploading, packaging, and deployment subsystem must exist; not only to bundle the XML document specifying the workflow along with all the tools, resources and Java class files for execution on the remote platform, but also to be able to fan-out these bundles potentially to millions of peers. For this to be achieved, we cannot rely on the current, centralised mechanism employed for data distribution in BOINC[6] (Berkeley Open Infrastructure for Network Computing, an open-source software platform for computing using volunteered resources) as this is not only expensive in both cost (i.e. we would need to buy a fairly heavyweight server for this purpose) and time (for administration) but also it would be extremely slow for updating the workflows to the network participants as existing Triana toolboxes for the analysis of such audio are over 10 megabytes in size. Further, such latency is simply not acceptable for the scientists who would like to be able to prototype their ideas quickly and not have to wait a few days for everyone to be updated before they can view the results.

To address this problem, we have chosen a peer-to-peer approach that is based on the super peer architecture but extends this idea to employ the use of secure data servers (called package repositories) that cache the workflow bundles for DART, to be able to replicate and decentralise the distribution of the workflows. Other techniques were considered, such as bittorrent but this is unacceptable within the BOINC framework because of security constraints. First,

[5] http://www.whipplugin.org/

[6] see BOINC website `http://boinc.berkeley.edu/`

it requires every user on the network to open a port for serving the data and secondly, it provides no scoping environment for DART to be able to restrict which servers can act as a data provider for their workflows. We would like scientists to be able to develop their ideas in confidence (and also to support commercial products) and therefore a user must be able to impose control on who is authorised to distribute the data. For this we employ the use of X.509 certificates that are issued by the Certificate Authority (CA), in this case the Dart manager, and made available to certain participants for authentication to become package repositories.

DART is a realisation of a Cardiff University research framework called Alchemist, which provides the underlying P2P infrastructure and mechanisms to create secure data caching groups for implementing the distributed package repositories. This framework is also used by another project that implements data distribution techniques for scientific data. More information about this framework can be found in publications [14] [3]. The scheme described here for distributing workflows therefore has many applications in distributed systems as a whole.

The rest of the paper is organised as follows; in the next section we discuss some of the background technologies and related work within this paper, including an overview of Triana and some workflow examples. In Section 3, we describe the DART framework itself, followed by a discussion of our simulation results for the application and network in Section 4, which shows how the system will scale as the number of peers (users) in the network increases. Finally the conclusion will argue that this approach to distributing dynamic computational workflows is valid, and outline the future work.

2 Background Technologies

DART represents a fusion of recent Internet scale distributed systems technologies. It encompasses security techniques from Grid computing, which has evolved from toolkits such as Globus[7] that provide the core mechanisms for managing the execution of jobs and transfer of data. The Alchemist framework, on which DART is built, is also based on Web services with support for the Open Grid Services Architecture (OGSA) [7] approach to expose resources using Web services standards[8] within a service oriented architecture (SOA). The recent implementation of OGSA in the form of WS-RF [6] provides decoupled mechanisms for representing stateful resource capabilities through stateless Web services interfaces, which allow a system to manage lifetime without using a tightly coupled approach such as those used in previous distributed object systems, e.g. Corba[9] or Jini[10]. Grid computing is built on standardised technologies and extended through other standardisation efforts often initiated through the various research and working groups hosted within the Open Grid Forum[11] (OGF). Both Alchemist and therefore DART intend to adhere to these standards as they evolve.

DART also employs the use of Peer-to-peer (P2P) technologies, which have grown through the development of popular applications targeted at specific services, such as Napster[12], Gnutella[13], and CPU sharing systems like SETI@home, with no desired path to their interoperability or standardisation. P2P designers have been more concerned about solving practical problems, such as discovery and communication between huge numbers of unreliable edge peers, and developing scalable and robust mechanisms to tackle frequently disconnecting peers

[7] http://www.globus.org/
[8] http://www.w3.org/TR/ws-arch/
[9] http://www.omg.org/
[10] http://www.jini.org/
[11] http://www.ogf.org/
[12] http://www.napster.com
[13] http://www.gnutella.com/

and unfriendly network application environments, e.g. Network Address Translation (NAT) systems and firewalls. More often, the society of peers cannot be trusted and therefore the content of a file or service cannot be the guaranteed. The data distribution techniques employed here build upon the super peer network structure that has been shown to cope with highly transient participants on a massive network scale.

Thirdly, DART builds on tools within the workflow domain for e-Science applications, where there has recently been a wealth of activity from specific application domains, such as the support for scientists to conduct *in silico* experiments in biology using Taverna [13] to more generalised systems, such as Kepler [12] or the VDS [4]. For a state-of-the-art overview of current workflow environments and toolkits, see [15].

Audio analysis algorithms and frameworks for Music Information Retrieval (MIR) are expanding rapidly, providing new ways to garner information from audio sources well beyond what can be ascertained from ID3 tags, the standard format for storing artist and song meta-data within an MP3 audio file. Modest successes have been made in audio-based musical genre classification audio-analysis algorithms such as musical genre classification [18][1], beat detection and analysis [5], similarity retrieval [11][1][21], and audio fingerprinting [9]. This work uses Short-Time Fourier Transforms to track the means and variances of the Spectral Centroid, standard deviations of the spectrum around its centroid, spectral envelopes, and signal power to represent sound textures, beat and pitch content [17]. These values are then transformed into attribute–value pairs for pattern matching and semantic retrieval. There is still much to be done in this field. Refinements to existing strategies, as well as new strategies are still needed.

The analysis component of MIR requires extensive computational resources. Distributed environments and P2P networks are already being used for this purpose [19]. The idea of using MIR over P2P was proposed in Wang et. al. [20],

however this system suffered from problems with scalability. More recently, the JXTA programming framework was used by Baumann [2] to aid in the content-based retrieval over a P2P network. The proposed DART system differs from the distributed MIR system proposed in [19] however, in that only metadata is returned to the main Triana server for analysis, as opposed to actual audio data files, and has a different overall goal; DART is not intended to act as a file sharing system, but instead a distributed P2P MIR system with the main application scenario focussing on the recommendation of music based on the audio files already stored on the user's hard drive.

Pandora[14] and is a novel music recommendation system from the makers of the Music Genome Project[15]. Pandora allows users to enter the names of artists or songs they like, and Pandora will consult return a play list of artists and songs that the user may like. DART hopes to build on these concepts by using the users actual audio files, extracting information based on the characteristics of the audio, and using these attributes to base the recommendations on

Last.fm[16] is an internet radio and music community website, and uses an MRS known as an *Audioscrobbler*[17]. Last.fm builds a profile of each users' musical taste by recording details of all the songs the user listens to using their media player (iTunes, Winamp, etc) or iPod. This information is 'scrobbled' (transferred to Last.fm's central database) via a plugin installed into the users' music player, and the users' profile data is displayed on a personal web page. The Last.fm website offers numerous social networking features and can recommend and play artists similar to the user's favourites. Again, DART hopes to build on this by performing audio analysis in addition to relying on statistical data, and will also employ a decentralised distributed P2P model. However, the Last.fm Audioscrobbler system also exposes its data via webservices so other projects

[14] `http://www.pandora.com/`

[15] `http://www.pandora.com/mgp.shtml`

[16] `http://www.last.fm/`

[17] `http://www.audioscrobbler.net`

can make interesting use of the data and statistical results and recommendations in the database. The feasibility of making use of this statistical data in DART in addition to the audio analysis workflows, is currently being researched by the DART team.

In summary, DART's P2P architecture aims to build upon all of these developments to provide an advanced, fully scalable platform for developing, testing and deploying new search and analysis algorithms on an Internet scale. Furthermore, as explained later in the paper, the DART system can be adapted to fulfil a variety of applications other than music recommendation by modifying the Triana workflow that is distributed to the worker nodes. The Triana framework and Triana workflows are discussed in the next section.

2.1 The Triana Workflow Environment

Triana[18] is a graphical Problem Solving Environment (PSE) for composing data-driven applications by executing workflows. Workflows (data or control-flows) are constructed by dragging programming components, called tools or units, from the toolbox onto a workspace, and then drawing cables between these components to create a graph. Components can be aggregated visually to group functionality and compose new algorithms from existing components. For example, to add a digital Schroeder reverb to a piece of audio (see Figure 1), the file could be loaded (using the LoadSound unit), then passed to a SchroderVerb group unit before being passed to the Play unit to hear the result. The SchroderVerb unit is itself a group, which consists of a number of summed comb delays and all-pass filters, representing the inner workings of such an algorithm.

Within Triana, a large suite of Java tools exist in a range of domains including signal, image and text processing. The most advanced tools in Triana are

[18] `http://www.trianacode.org`

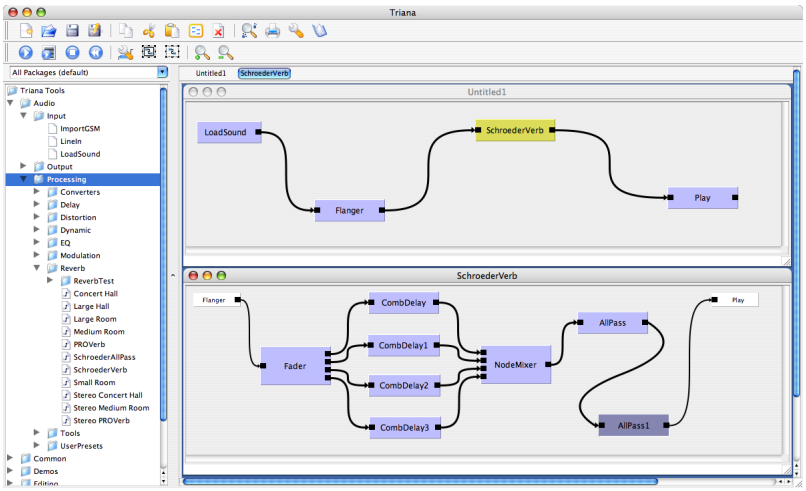


Figure 1: A simple audio processing work-flow, showing how a Schroeder reverb is applied to a signal by using a group, which contains the underlying algorithmic details.

created for signal processing, as Triana was initially developed for signal analysis within the GEO600 Gravitational Wave Project[19], who use the system to visualize and analyse one-dimensional signals (rather like an audio channel but sampled at a lower rate). Therefore a number of core mathematical, statistical and high-quality digital-signal processing algorithms already exist.

Triana integrates Grid, Web service, and P2P technologies, and has been used in a number of domains, from bioinformatics, investigating biodiversity patterns, to gravitational wave observation, using computational Grids to process one-dimensional signals using standard digital signal-processing (DSP) techniques. The goal of the DART project is to leverage this technology such that the same kind of DSP processing can be achieved with audio rate signals for the purposes of signal analysis, feature extraction, synthesis, and music information retrieval.

[19] `http://www.geo600.uni-hannover.de/`

Given its modularity, its support for high quality audio, and its ability to distribute processes across a Grid of computers, Triana has the potential to be an extremely useful piece of software that allows users to implement custom audio processing algorithms from their constituent elements, no matter their computational complexity.

Triana's data type classes are fundamental to Trianas flexibility and power. Data Types are containers for the data being processed by the units or tools. The two main types in Triana that are relevant to the processing of digital audio in Triana are *MultipleChannel*, a base class for representing multiple channelled data. The *MutipleAudio* Data Type stores many channels of sampled data, again, each channel can have its own particular audio format of the data e.g. the encoding format, sample rate and number of bits used to record the data. These data types are essential for reading and writing sound files, and analysing audio data. In essence, MultipleAudio provides the support for high quality audio to be processed within Triana.

The Triana audio toolkit consists of several categorized hierarchical folders, each with an assortment of units based on the MultipleAudio Triana data type. This type utilises the JavaSound API classes in order to allow the use of high fidelity audio. The Audio toolkit tree is split into three main folders: Input, Output, and Processing.

The sub-categories inside these folders should be recognizable to regular users of audio processing or production software, with sub-folders such as Converters, Delay, Dynamics, Modulation, MIR, and EQ units, to name a few. Each folder contains units which are in themselves effects but also allow the user to create their custom algorithms from the smaller building blocks supplied. This unit aggregation gives the user more freedom to take advantage of Trianas modularity.

One feature of interest resides in the Converters folder; two units are available that allow the user to convert from MultipleAudio to a SampleSet Triana

data type – and back again (MAudioToSSet and SSetToMAudio respectively). This opens up a whole range of possibilities to the user, enabling them to utilise many of the numerous math and signal-processing units that work with the SampleSet data type) to process the audio, and then convert data back to a MultipleAudio data type for playback. One example of how this technique could be used is shown in Figure 2. In this example, a Stereo2Mono unit is used to split the stereo channels of an audio file or stream into two distinct mono channels. Each side is then converted to a SampleSet and fed into a Subtractor unit from the Math folder. This subtracts the left from the right stereo channel, which results in the removal of sound that is contained in both i.e. those panned in the middle of the stereo field. This is a simple way of removing vocals from many songs and leaving the (majority) of the backing track (as vocals in particular are normally panned down the centre). This is just a simple example of how a few of the converter units could help users create their own algorithms.

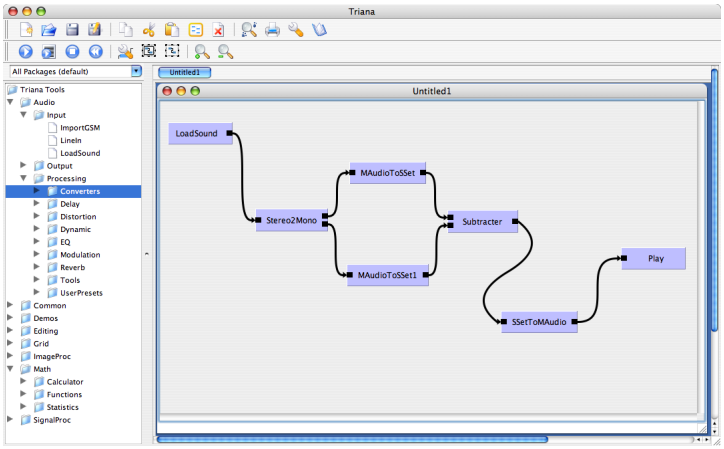


Figure 2: Splitting stereo audio and subtracting the left from right channel

As mentioned previously, Triana also contains hundreds of statistical, mathematical, and signal processing units, which can be used in conjunction with all of the MultipleAudio compatible units, opening up an vast range of units

to facilitate and aid MIR and the creation of useful MIR algorithms. Triana includes Fast Fourier Transform units, a range of filters, graph and histogram viewers, spectrum analysers and more, meaning that scientists can take advantage of pre-written software modules within Triana to aid in the development of new algorithms. This allows the user to bypass the conventional approach to programming; creating various methods and coding a core program to connect the set of related method procedures together.

Triana is capable of working within P2P environments through the use of P2PS and JXTA and it can work within Grid environments through the use of the Globus toolkit accessed via the GAT/GAP interface. Further, it has the capability of fusing these environments through the use of WSPeer, which can host Web Services and OGSA implementations, such as WS-RF, within P2P environments like P2PS, see [16] and [10] for a full description.

In DART, we are interested in forming unstructured P2P networks and therefore need to employ technologies that can adapt and scale within such an environment. For distribution across dynamic networks, we use the P2PS binding for Triana. Peer-to-Peer Simplified (P2PS) was a response to the complexity and overhead associated with JXTA. As its name suggests, it is a simple yet generic API for developing P2P systems. P2PS encompasses intelligent discovery mechanisms, pipe based communication and makes it possible to easily create desirable network topologies for searching, such as decentralised ad-hoc networks with super peers or rendezvous nodes. P2PS is designed to cope with rapidly changing environments, where peers may come and go at frequent intervals.

At the core of P2PS is the notion of a pipe: a virtual communication channel that is only bound to specific endpoints at connection time. When a peer publishes a pipe advertisement it only identifies the pipe by its name, ID, and the ID of its host peer. A remote peer wishing to connect to a pipe must query an endpoint resolver for the host peer in order to determine an actual

endpoint address that it can contact. In P2PS a peer can have multiple endpoint resolvers (e.g. TCP, UDP etc), with each resolving endpoints in different transport protocols or returning relay endpoints that bridge between protocols (e.g. to traverse a firewall). Also, the P2PS infrastructure employs XML in its discovery and communication protocols, which allows it to be independent of any implementation language and computing hardware. Assuming that suitable P2PS implementations exist, it should be possible to form a P2PS network that includes everything from super-computer peers to PDA peers.

In the DART framework the distribution policy for Triana is loosely coupled. Although Triana acts as a manager and processor in the system, the distributed functionality is provided by the DART framework, which implements a decentralised discovery and communication system based on P2PS. This allows Triana workflows to be uploaded to peers for execution and enables users to query the network to locate results and to perform custom searches. The DART system therefore manages the specifics of the network and Triana acts as a client (i.e. the DART manager or user) that both accesses DART and also acts as an end processor to execute workflows that have been previously uploaded for the analysis of the audio. Therefore, Triana does not tie the network together, rather it accesses a loosely coupled framework that allows wide-range distributed Triana entities to communicate via the Internet. Section 3 of this paper discusses the DART framework in more detail and details the mechanisms to facilitate so-called work package assignment and the retrieval of the results.

3 Dart: A Framework For Distributed Information Retrieval

3.1 Work Package Assignment and Results Retrieval

Many DART applications are process-intensive and can require the distributed analysis of a large number of audio files in order to provide some useful, non-trivial feedback to the user. One of the main aims of the DART system is the development of a music recommendation system, whereby music on a users hard drive (usually in MP3 format) is analysed, and recommendations are made to the user based on the results of the analysis. Both index based analysis, including statistical correlations between song names to extract commonalities in order to make a recommendation, and audio content based analysis such searching the track for tempos, timbre, mood, pitch, and frequency range etc., can be used to analyse the audio. Algorithms for both types of analysis are composed in Triana (as discussed earlier), from workflows that are created by the DART Manager. These workflows are bundled into a workflow 'package' (which can also be called a 'DART package', and contains the Java code and their required resources) that is distributed onto the network in order to upload the new analysis to the users (or workers) machines. As the algorithms are updated and refined by the DART Manager, the updated Triana workflow and any new tools are bundled into a new package, and propagated onto the network.

The scenario discussed in this section describes the decentralised DART network, in which nodes are organized in a super peer topology using the P2PS middleware. In P2PS producers (e.g. providers of packages containing workflows or results) create adverts to advertise that they have something that is of interest to other participants in the network. Consumers (i.e. the peers that wish to use available packages, result sets and so on) issue queries in order to search for relevant adverts. P2PS rendezvous nodes are then responsible for matching

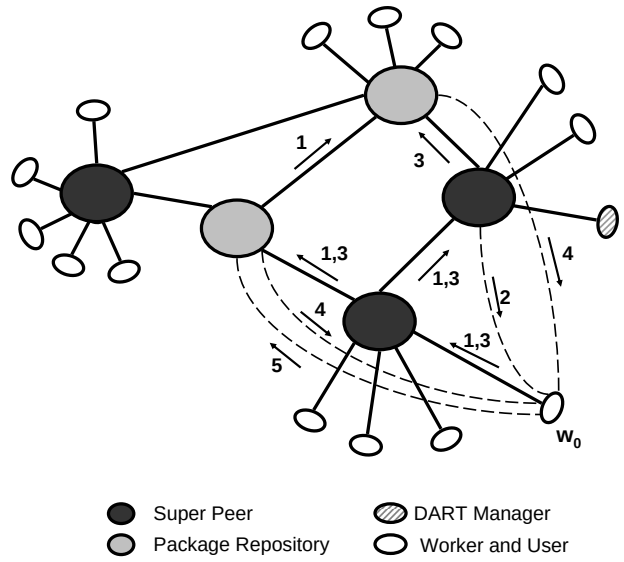
queries with adverts within their local cache, in order to search for matches and respond appropriately. Consumers receive adverts when their query matches, and these adverts can be used to retrieve the relevant information they require i.e. download the new workflow package to perform the analysis. The DART Manager node produces and advertises the workflow package representing the new DART bundle (called DART Package Adverts) containing algorithms that the worker nodes need to run (new Triana units and workflows). With the DART system, the data files that undergo analysis are on the users local systems hard drive, and therefore all data processing is local so network bandwidth is not consumed transferring large data files over the DART network. Although local, the processing is massively parallel, as participants analyse their own audio files in parallel.

Simple peers (or 'workers') are available to execute the algorithms and workflows, and therefore issue a package query to download a package in order to start the analysis of their music collections. The entire workflow is executed by each worker which downloads the package; the workflow is not then broken down and farmed out to separate nodes to complete different tasks.

Super peer interconnections are used to make package queries travel across the network rapidly; super peers play the role of rendezvous nodes, since they can store package adverts and compare these files with queries issued to discover them; thereby acting as a meeting place for both package providers and consumers. Since packages could require a reasonable amount of storage space, it is assumed that only some of the peers in the network will cache these files. These peers are called Package Repositories (PR) and can also be super peers or worker peers. Each node in the system decides if they want to be a super peer and/or package repository, as well as a worker.

Figure 3 shows a sample topology with 5 super peers (2 of which are also package repositories), and the sequence of messages exchanged among different nodes in order to perform the package submission protocol. These messages are

related to the execution of a workflow by a single worker, labelled as W_0. Note that in this figure, normal peers are not considered as package repositories.



Messages

1. *PackageQuery*
2. *PackageAdvert*
3. *DataQuery*
4. *DataAdvert*
5. *Download Request*

Figure 3: Super peer protocol for the dissemination of workflow packages: sample network topology and sequence of exchanged messages to execute one package cycle.

When a new DART workflow package is available, the DART manager puts this package on one or more package repositories and propagates an associated metadata file, or *package advert*, on the super peer network. This advert is an XML file describing the properties of the algorithms to be executed (i.e.

workflow parameters containing the units/tools, platform requirements if any, information about required input audio data files, etc.).

When available to offer some of its CPU time, a worker searches the network to verify if a new version of the package is available. More specifically, the worker sends a *package query* that travels the network through the super peer interconnections (message 1 in Figure 3). A package query is expressed by an XML document that contains hardware and software features of the worker node, if this is necessary e.g. available RAM, disk space or JDK version. The query succeeds whenever it matches an advert of a package that can be actually executed by the requesting worker. This *package advert* is then sent directly to the worker.

Thereafter, the worker must search for a package repository that stores the updated workflow package, and sends a *data query* to the network. As more than one package repository can match the query, a matching repository does not send the package directly to the worker, in order to avoid multiple transmissions of the same file. Conversely, the repository returns only a *data advert* to the worker.

A worker can choose a repository according to policies that can rely on the distance of repositories, their available bandwidth etc. Then the worker initiates the *download* operation from the selected repository.

The DART protocol allows for the progressive dissemination of workflow packages on different repositories. Initially these packages are stored on one or few repositories. However, when a worker downloads a package, if the local super peer plays also the role of a repository, the package is first downloaded and cached by this super peer, then forwarded to the worker. In the future another package query can be matched directly by this package repository. Replication of the workflow package on multiple package repositories allows for a significant saving of time in the querying phase and enables the simultaneous retrieval of packages from different repositories.

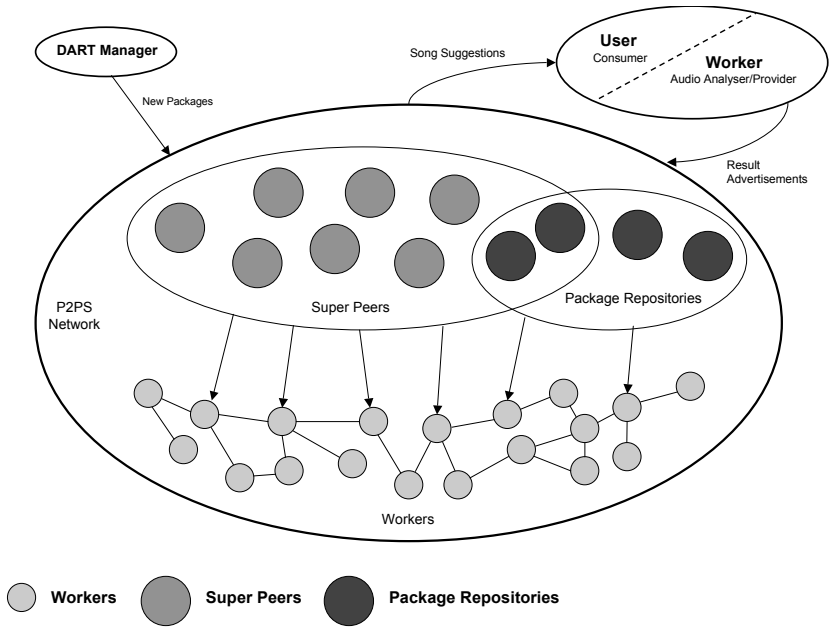


Figure 4: High-level overview of the DART system, showing the various peers and their connectivity.

Once the worker has received the updated package, the workflow is executed by the workers, and they begin to analyse the audio files on the workers system during the systems idle time. Once a package cycle is complete and the worker has results to present, the worker then creates an XML advert containing the results and metadata generated by the algorithm specified in the package (a results advertisement). As the actual results generated would be extremely small in size in this DART system, the functionality of the super peer has been extended in order to also cache and make them available.

Each worker on the network can also be thought of a results provider on the DART system, as well as act as a user, as it can query for results (in this case a suitable music/song suggestion as generated on the super peer). There

is no central results collector, but rather DART utilises a fully decentralised model and allows the results to propagate through the network hop by hop, to be stored on the super peers. The super peers can process the metadata and issue an XML results advertisement on receipt of a results query from the user. Once the query is received, the results may be sent to the user.

3.2 Peer Overview

This section gives a brief overview of what role each node on the network plays, and the jobs associated with that role:

DART Manager

- Creates new workflows and Triana units
- Advertises new DART Packages

User

- Queries Results
- Downloads/Receives Results

Worker Nodes

- Queries New Packages
- Downloads Package
- Works/processes Workflows
- Advertises Results

Package Repositories

- Query and Download New Packages (Stored locally)

- Advertise Packages

Super Peers

- Caches Advertisements
- Performs simple analysis of results received from workers

3.3 Workflow Design

The workflow created by the DART manager and distributed to the other peers on the network, will always be evolving. Initially, it is easier to concentrate on statistical methods for recommending music to the users of the DART system, however this will soon evolve to become more involved as the Triana workflow is refined and more MIR algorithms are incorporated into the package sent to the worker nodes. The algorithms used for the analysis and recommendation of music will be made available to the MIR community for refinement, suggestions, and also to allow for the advancement of this field of research as MIR-algorithm specialists provide input ideas and offer improvements to both refine and maximise the benefit of the use of the DART system.

3.4 DART Flexibility

So far this paper has focussed on the topic of distributing Triana workflows over a ubiquitous P2P network in order to facilitate the recommendation of music to its users, and the DART framework created in order to achieve this. However the DART infrastructure can be used to distribute workflows for a variety of applications, musical and otherwise. One application scenario is a DART system installed on a local/closed network; scanning sound effect audio files on separate systems in networked commercial studio facilities, to search for sound effect files (or for example, drum sounds) that match a specific criteria. For example, if a DART user requires a snare drum sample, usual search mechanisms cannot

search for suitable sounds unless the audio files filename contains the word snare or other suitable identifier – the search mechanism would do no more than string matching.

However, given the appropriate workflow created by the DART manager, when propagated onto the network the DART system would allow the user to search all the audio content in the distributed sound library, and suggest files with suitable characteristics that match the specific search criteria set by the user. This would also be able to return further results, which would not be returned via conventional methods – any samples that match the users search criteria, will be returned. This means that when ignoring criteria such as pitch and tempo and investigating the timbre of the sound file, new sounds that could be adapted to work for the user, could be suggested. For example, a sped up sample of a car collision may easily work well in place of a snare sample!

Another example consists of a scenario that requires speech recognition algorithms to detect a certain caller from a large amount of recorded telephone audio data that was distributed over several machines. Given a sophisticated enough algorithm (created in Triana as a workflow), the DART framework could be used to scan terabytes of recorded audio data to help trace calls from a specific caller. The flexibility of the DART system stems from the ability to easily refine and change the Triana workflow that dictates what the worker nodes will be processing in their screensaver/idle/nice time.

4 Distributed Simulations

In order to demonstrate the DART system's potential to scale and distribute workflows over a ubiquitous P2P network as the number of participants increase, ICAR-CNR (The Institute of High Performance Computing and Networking) in Italy, and Cardiff University, Wales, conducted a study in which a simulation of the DART scenario (as discussed in Section 3) was run. A simulation analysis

was performed by means of an ad hoc event-based simulator, written in C++, to evaluate the performance of the package dissemination protocol described in Section 3.1.

The simulation scenario is described in Table 1. In this case, a workflow package of around 10.4mb in size (the current size of the Triana audio toolkit) is to be distributed to the worker nodes on the network. It is here assumed that the workflow can be executed by any worker and that only super peers can be package repositories.

Table 1: Simulation scenario.

Scenario feature	*Value*
Number of workers, or simple peers, N_{peer}	1,000 to 20,000
Number of super peers, N_{speer}	100 to 2,000
Average number of workers connected to a super peer	10
Maximum number of neighbors for a super peer	4
Average connection time of workers	4 hours
Average disconnection time of workers	1 hour
Number of package repositories	1 to 50% of N_{speer}
Size of input data files	10.4 Mbytes
Latency between two adjacent super peers	100 ms
Latency between a super peer and a local worker	10 ms
Bandwidth between two adjacent super peers	2 Mbps
Bandwidth between a super peer and a local worker	1 Mbps
Mean workflow execution time	10 hours

This scenario simulates the performance and behaviour of a distributed P2P network with 1,000 to 20,000 workers, with a maximum value of 2,000 super peers as the number of super peers is assumed to be 10% of the number of workers. Workers can disconnect and reconnect to the network at any time. This implies that the download or execution of a workflow fails upon the dis-

connection of the corresponding worker. It is assumed that connections between two adjacent super peers have a larger bandwidth and a longer latency than local connections (i.e. between a super peer and a local simple node) .

In the simulation scenario, each worker has to download and execute a workflow package; to this aim it issues a package query and follows the protocol described in Section 3.1. If the download operation fails due to a worker disconnection, a new package query is forwarded and the procedure is repeated.

The experiments aimed at evaluating the effectiveness of the dynamic caching mechanism described in Section 3.1. Therefore, the number of available package repositories was varied from 1 to half the number of super peers: one of these repositories provides the workflow package from the beginning, while the others act as *cachers*, as they can download, store and provide data on the fly.

Simulations have been performed to analyze the overall *dissemination time*, T_{diss}, defined as the time needed to propagate the workflow package at least at the 95% of the workers. This time is crucial to determine the rate at which workflow packages can be retrieved from the package repositories in order to guarantee that the workers are able to keep the pace with the availability of new versions of the package. The average time needed to perform a single download operation, T_{dl}, is also calculated.

The average utilization index of package repositories, P_{act}, is defined as the fraction of time that a package repository is actually utilized, i.e., the fraction of time in which at least one download connection, from a worker (or another repository), is active with this repository. The value of P_{act} is averaged on all the repositories and can be seen as an efficiency index.

Figure 5 shows that the dissemination time decreases as the number of package repositories increases, as worker nodes can exploit higher parallelism and download workflow packages from multiple repositories (possibly closer) and also because the repositories themselves are under less stress and therefore data

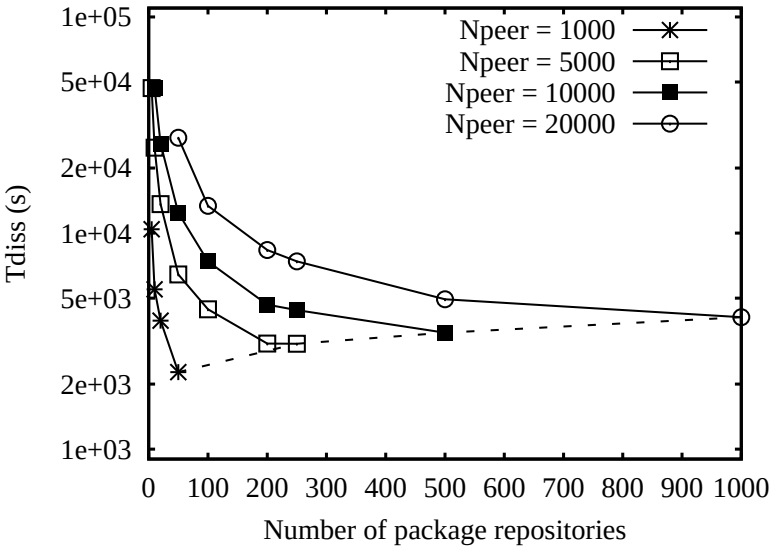


Figure 5: Time at which 95% of workers have downloaded a new version of the workflow package from a package repository.

download time decreases. Conversely, if the number of repositories is constant, the dissemination time increases with the number of workers as more download operations must be performed and therefore a single repository has to serve more workers on average.

In these simulations, the protocol is shown as scalable when one observes the results obtained with a fixed percentage of repositories. As an example, observe results obtained when the number of repositories is set to 5% of peers (i.e., 50% of super peers, see dashed line in Figure 5). It is interesting to note that as the number of peers increases, the dissemination time increases very slightly, much less than the number of peers. For example, with 1,000 peers and 50 repositories, dissemination time is approximately 2,200 seconds; however with 20,000 peers and 1,000 repositories, dissemination time is approximately 4,100 seconds.

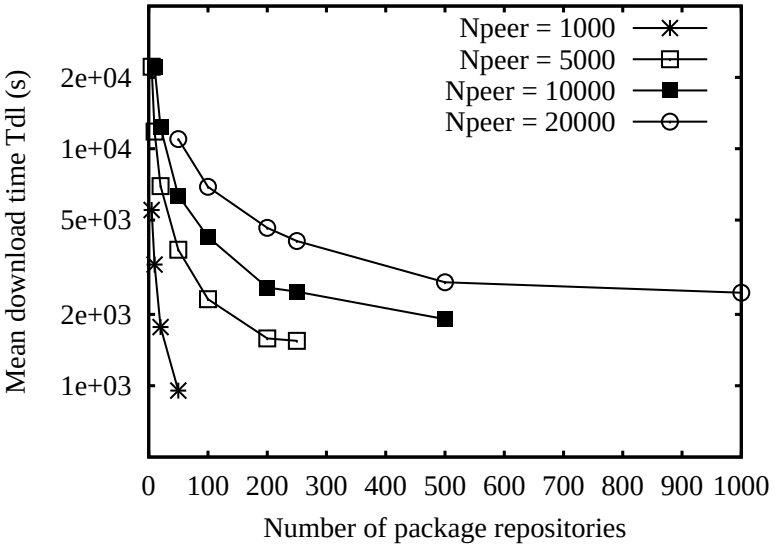


Figure 6: Average download time of single worker from package repository when worker disconnection does not interrupt the download.

Figure 6 shows the average download time of a single worker from a package repository when a worker disconnection does not interrupt the download operation. The results here are analogous to the behaviour previously observed when considering Figure 5; the qualitative behaviour is the same, but the values are lower. The download time decreases as the number of repositories increases, and the download time will increase as the number of workers increases.

Figure 7 shows the percentage of time in which a repository is actually exploited (i.e. there is at least one download in progress). We can observe that as more repositories become available, the percentage of time decreases; therefore, one should be hesitant in setting up a high number of repositories, because while this can slightly decrease dissemination time, they can also be under-exploited. Another interesting result is that the utilisation of a given number of repositories increases as the network becomes bigger and more workers need to download

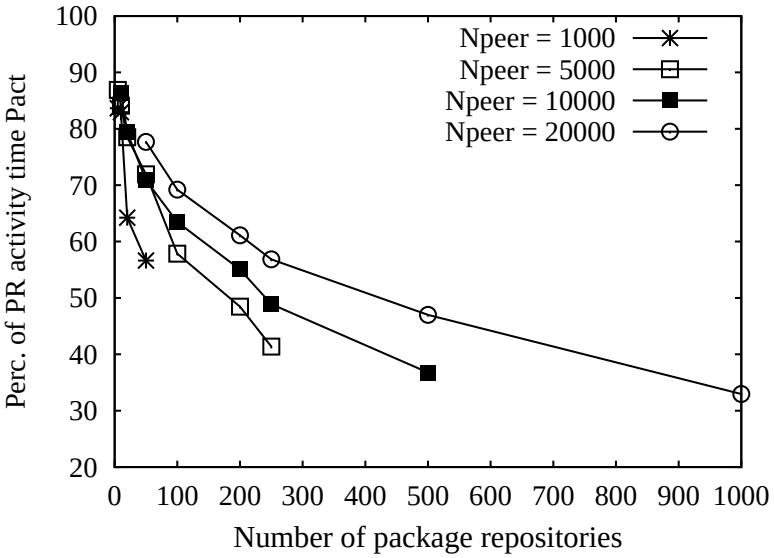


Figure 7: Percentage of time in which a package repository is actually exploited (at least one download in progress).

the workflow package. This is another verification of the scalability behaviour discussed earlier. It should be noticed that this percentage never reaches 100% because a data cacher (a package repository that has no data at the beginning) must download data from another repository before it can serve a worker.

Figure 8 displays the percentage of download operations that are interrupted due to the disconnections of corresponding downloading workers. In this simulation, only results for which this percentage is lower than 30% are displayed. It was observed that the percentage of interrupted downloads decreases as the number of repositories increases. In fact, the download time decreases if more repositories are available (see Figure 6), then a worker has more chances to conclude its download operation. Finally, if the percentage of repositories with respect to the number of peers is set to a given value (for example 5%), the percentage of interrupted downloads is almost constant (see dashed line), which

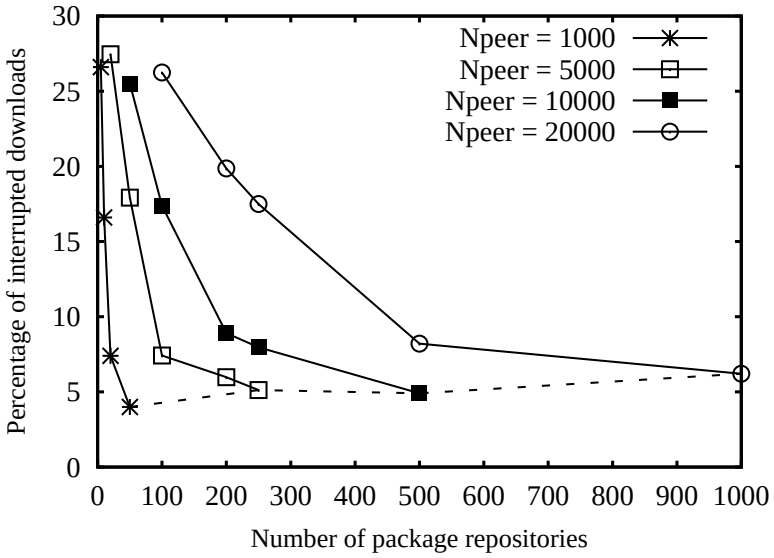


Figure 8: Percentage of interrupted downloads.

is a further confirmation about the scalability of the dissemination protocol.

5 Conclusions

The DART Music Recommendation System (MRS) project makes novel use of dynamic workflows in a massively distributed P2P environment. Remote processing on peers in the DART network is performed through the execution of workflows designed for Music Information Retrieval (MIR), and dynamically propagated through the network and discovered using a super peer mechanism. The scientist or analyst (DART manager) that creates the workflow is able to modify it and retransmit it to the peers, fine tuning the application based on results.

We have performed extensive simulations using an ad-hoc event simulator to explore how the transmitted workflows will propagate throughout the network

as the number of peers and super peers increases. The results show that the network will scale as the number of members increases as long as the number of super peers that act as data providers increases by the same ratio. This shows that the real application should be capable of operating at an Internet scale with acceptable performance. Given the simulation results, a real network and application is the next goal of the project.

References

[1] J. J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93, 2003.

[2] S. Baumann. Music Similarity Analysis in a P2P Environment. *Proceedings of the 4th European Workshop on Image Analysis for Multimedia Interactive Services, London, UK, April*, 2003.

[3] P. Cozza, C. Mastroianni, D. Talia, and I. Taylor. A super-peer protocol for multiple job submission on a grid. In W. Lehner, N. Meyer, A. Streit, and C. Stewart, editors, *Euro-Par 2006 Workshops*, volume 4375 of *LNCS*, pages 116–125, Dresden, Germany, June 2007. Springer Berlin/Heidelberg. ISBN: 978-3-540-72226-7.

[4] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, and S. Koranda. GriPhyN and LIGO, building a virtual data grid for gravitational wave scientists. In *HPDC*, pages 225–, 2002.

[5] J. Foote and M. Cooper. Audio retrieval by rhythmic similarity. In *Proceedings of the International Conference on Music Information Retrieval*, volume Paris: IRCAM, pages 265–266, 2002.

[6] I. Foster et al. Modeling Stateful Resource with Web Services. `http://www.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf`.

[7] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.

[8] C. Goble and D. De Roure. myExperiment: social networking for workflow-using e-scientists. *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, pages 1–2, 2007.

[9] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Proceedings of the Third International Conference on Music Information Retrieval*, volume Paris: IRCAM, pages 107–115, 2002.

[10] A. Harrison, I. Wang, I. Taylor, and M. Shields. WS-RF Workflow in Triana. *International Journal of High Performance Computing Applications (IJHPCA) Special Issue on Workflow Systems in Grid Environments*, to be published 2007.

[11] B. Logan and A. Salomon. A Content-Based Music Similarity Function. *Cambridge Research Laboratory. Technical Report Series. Jun*, 2001.

[12] B. Ludäscher, I. Altintas, C. Berkley, D. G. Higgins, E. Jaeger, M. Jones, E. A. Lee, and Y. Zhao. Scientific workflow management and the kepler system. concurrency and computation: Practice and experience. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, 2006. to appear.

[13] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience, special issue on Grid Workflow*, accepted for publication, 2006.

[14] M. Shields and I. Taylor. Alchemist: user driven searching in ubiquitous networks. *Proceedings of the 1st international workshop on Advanced data processing in ubiquitous computing (ADPUC 2006)*, 2006.

[15] I. Taylor, E. Deelman, D. Gannon, and M. Shields (Eds.). *Workflows for e-Science*. Springer, New York, Secaucus, NJ, USA, 2007.

[16] I. Taylor, M. Shields, I. Wang, and A. Harrison. Visual Grid Workflow in Triana. *Journal of Grid Computing, Special Edition on Workflow*, To be published, 2005.

[17] G. Tummarello, C. Morbidoni, P. Puliti, and F. Piazza. Semantic audio hyperlinking: a multimedia-semantic web scenario. In *Proceedings of the 1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution*, pages 111–115, 2005.

[18] G. Tzanetakis and P. Cook. Music genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[19] G. Tzanetakis, J. Gao, and P. Steenkiste. A scalable peer-to-peer system for music information retrieval. *Computer Music Journal*, 28(2):24–33, Summer 2004 2004.

[20] C. Wang, J. Li, and S. Shi. An Approach to Content-Based Approximate Query Processing in Peer-to-Peer Data Systems. *Grid and Coopera-*

tive Computing, Second International Workshopi (GCC 2003), Shanghai, China, December, 2003.

[21] C. Yang. Music Database Retrieval Based on Spectral Similarity. *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval*, pages 37–38, 2001.