

A Unified Format for Traces of Peer-to-Peer Systems

Boxun Zhang
TU Delft,
the Netherlands
B.Zhang@tudelft.nl

Alexandru Iosup
TU Delft,
the Netherlands
A.iosup@tudelft.nl

Pawel Garbacki
Google, Inc.,
Switzerland
PawelG@gmail.com

Johan Pouwelse
TU Delft,
the Netherlands
peer2peer@gmail.com

ABSTRACT

Peer-to-Peer (P2P) systems have recently emerged as a scalable platform for which costs are shared between the system users. Today, P2P technology is serving millions of users world-wide, with applications such as file sharing, video streaming, grid computing, and massively multiplayer online games. Such diversity and scale pose important research and technical problems, which in turn require a much better understanding of the usage patterns and of the performance bottlenecks. However, the large amounts of P2P monitoring and measurement data that already exist have not been made public, for fear of lack of anonymity and in lack of a standard format. To address this problem, in this work we propose a unified format for workloads of P2P systems. Our format stores information coming from many types of P2P applications at several levels of detail, has a structure that balances generic and application-specific data, and protects the anonymity of the peers whose personal information was captured in monitoring and measurement data. Using two large traces taken from real P2P systems we show evidence of the usefulness of the proposed format, and substantiate the hope that our unified format has the potential to become a standard for sharing P2P traces.

Categories and Subject Descriptors

E.5 [Files]: Organization/structure;
H.2.1 [Logical Design]: Schema and subschema;
D.4.7 [Organization and Design]: Distributed systems;
D.4.8 [Performance]: Measurements

General Terms

Standardization, Performance, Measurement, Documentation

Keywords

peer-to-peer, unified trace format, performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

LSAP'09, June 10, 2009, Munich, Germany.

Copyright 2009 ACM 978-1-60558-592-5/09/06 ...\$5.00.

1. INTRODUCTION

Peer-to-peer (P2P) technology provides the means for building truly scalable systems. The P2P paradigm of system users also providing the system resources voluntarily or in return for service, has distinguished itself as a new research field with the focus ranging from large-scale resource sharing to innovative applications with millions of users. Large-scale P2P systems such as the BitTorrent and the eDonkey file-sharing networks affect today the lives of millions of people, and there are many ongoing efforts to make the current use of P2P technology more efficient, and to design and deploy new P2P systems. To be successful in the future, these systems and efforts need a solid understanding of the P2P usage patterns, and raw monitoring and measurement data taken from real P2P systems to drive system tuning and research experiments. While several datasets and related analysis studies exist, the datasets are not publicly available [5,13,25] or they are presented in a format that does not allow their comparison and combined use with other datasets [17]. In this work we present the design and use of a unified format for traces of P2P systems that can be used for these purposes, and in particular as a component of a large-scale performance analysis and evaluation toolbox.

One of the areas that can benefit from the existence of a unified format for traces of P2P systems is the area of performance analysis and evaluation of P2P systems. A direct benefit would be obtained by complementing the current model-based scenarios with a trace-based approach. In this way, the hidden patterns that exist in real traces of large-scale systems will be implicitly used to improve the testing and tuning of P2P systems. We single out three indirect benefits. First, from a methodological point of view, a unified format would allow a straightforward and meaningful comparison of traces, and lead to valid models for the system workload. In fact, none of the many models used so far in reported P2P simulations has been validated against multiple traces, even when several traces of the same system exist within the community, such as in the case of BitTorrent. Second, an important part of the experimental work is currently spent in setting up the input workload. While many support tools and techniques exist [6,7,16], the sheer heterogeneity of P2P applications make the setup process difficult to repeat and unnecessarily complex. Third, a unified format that becomes a community standard fosters data exchange. In the authors' experience, it is difficult to obtain data from other researchers, as the raw data are usually

stored in an measurement-specific format that is undocumented and/or time consuming to explain. Moreover, sharing the raw data is legally difficult due to the un-anonymized personal information within [8].

Our unified format for traces of P2P systems combines three key ideas. First, to store information obtained from the multiple layers of functionality present in real P2P applications, we create a multi-layered format. For example, in the BitTorrent P2P file-sharing network there exists a layer for finding if a file exists, another for finding information about who is willing to share it, and yet another for the transfer of the file. Second, to accommodate as many P2P applications as possible, and specifically non-file-sharing P2P applications such as P2P grids [5] and P2P massively multiplayer online games [20], our format is designed as a collection of multi-purpose relational databases. Third, considering the volume of data that results from P2P monitoring or measurements, we separate in our format raw traces from derived data, and record only the former.

Today's need for a unified format for P2P trace data exchange and use is similar to that of the parallel systems community a decade ago, and of the grid computing community around five years ago. There, researchers have first proposed unified trace formats, and have only then started to gather datasets into public community archives such as the Parallel Workloads Archive [4] and the Grid Workloads Archive [18]. The parallel and grid computing communities have created unified formats only for the very specific system function of running user jobs. Similarly, the web community has developed a common logging format for web server accesses [28,29]. In contrast to these unified formats, our goal of creating for P2P traces a unified format that reflects the multi-layered structure and the heterogeneity of P2P applications is much more difficult to achieve. There are many P2P applications, and thus a much larger number of system functions and events that need to be covered. Moreover, the applications themselves are very different in scope, and a unified format needs to find the right balance between generic and application-specific information. Our main contribution towards the design of a unified format threefold:

- We formulate the requirements of a unified format for traces of P2P systems (in Section 2);
- We design a unified format for traces of P2P systems (in Section 3) and demonstrate its use with two large P2P traces (in Section 4);
- We lay a path towards a community P2P trace archive (in Section 5).

2. THE PROBLEM

In this section we formulate the problem addressed in this work. We first formulate a motivating scenario, then describe the research problem.

2.1 Motivating Scenario: BitTorrent

A P2P *system* is a system that uses P2P technology to provide a set of services; this group of services forms together an application. There are many applications of P2P systems; of these, P2P applications such as file-sharing, video streaming, grid computing, and massively multiplayer online games have already gained a large-scale audience. We

call *peers* the participants in a P2P system that contribute to or use the system's resources and services; a real user may run several peers simultaneously. A peer is completely disconnected until it *joins* the system, and is *active* until *leaving* the system. During the period between a join and its consecutive leave a peer is *active* in the system. We call a *swarm* the group of peers, from all the peers in a P2P system, that interact with each other for a specific goal, such as transferring a file. While active in the system, peers may be part of one or several swarms; the concept of system activity extends directly to per-swarm activity.

We choose as our motivating scenario BitTorrent, a P2P application that focuses on file sharing. The reason is three-fold. First, BitTorrent is currently the largest P2P file-sharing, with an Internet traffic share that grew from 30% to over 50% in the past five years [3,24]. Second, BitTorrent has excellent scalability, especially for flashcrowds, that is, sudden spikes in the number of users and in the resource demand. Third, we have conducted several large-scale measurements of BitTorrent over the past five years, including two that remain the largest reported BitTorrent measurements to date [17,25].

BitTorrent is a complex, multi-level P2P application, which includes at least the community web site level, the trackers level, and the file transfer (peer) level. The files (*torrents*) transferred in BitTorrent contain two parts: the raw file data and a metadata (directory and information) part. Peers interested in a file obtain the file's metadata at the web site level and use the peer location services present at the tracker level to find other peers interested in sharing the file. The raw file data is then exchanged between peers at the peer level. To facilitate this exchange, the raw data are split in smaller parts, called *chunks*. Thus, to obtain a complete file a user has to obtain all the chunks comprised by a file through the use of three application levels. One of the most delicate parts of this process is the setup of a connection between peers, which employs a proprietary handshake protocol; due to the growing presence of firewalls [23], this protocol fails often, limiting the performance of BitTorrent. In BitTorrent, a *leecher* is a peer who still needs chunks, and a *seeder* is a peer who has the complete torrent and shares it with other peers.

2.2 Problem Statement

The problem addressed in this work is the design of a unified format for traces of P2P systems. To address this problem, we have to answer the following two research questions:

Q1: *How to build a format that addresses the requirements for a unified format for traces of P2P systems?*

Q2: *How to build a format that can be used for common P2P applications such as file-sharing, video streaming, grid computing, and massively multiplayer online gaming?*

We identify five main requirements for a unified format for traces of P2P systems; a format that addresses these five requirements answers to the two research questions.

Requirement 1: Multiple levels of information To make it more accessible, a unified format must be designed to reflect as much as possible the structure of P2P applications. Thus, a common set of levels must be found across P2P applications.

Requirement 2: Comprehensiveness A unified format must be designed so that it can be used for the most

common P2P applications such as file-sharing, video streaming, grid computing, and massively multiplayer online gaming. However, the format must not only comprehensively cover current trace features, but also be extensible to accommodate future characteristics.

Requirement 3: Anonymity A unified format must be designed such that all personal information concerning peers can be presented in an anonymous way, while preserving the features that would be useful to the trace users. For example, from a user profile the name may be anonymized (or even omitted unless it has some social meaning), but the user location may be important for latency-aware P2P applications. However, anonymized data should not be lost for its rightful owner; instead, the format must provide tables for mapping between original and anonymized data and vice-versa. The mapping should be sensible enough to prevent the possibility that the data will be de-anonymized by correlating multiple (anonymized) traces.

Requirement 4: Balance between generic and specific The design of a unified format must find a good balance between the need for generic information that allows comparing traces taken from different systems, and application- or even trace-specific information that is needed for reproducing accurately the original trace.

Requirement 5: Minimality A unified format must be designed to include the minimal amount of information needed to reproduce accurately the trace to which it has been applied. In this sense, the format should make a clear distinction between raw (primary) and derived (secondary) data, and be designed to store only the former.

3. A UNIFIED WORKLOAD FORMAT

In this section we present our unified format for traces of P2P systems. We begin by showing how we address the requirements formulated in Section 2.2. We give throughout this section examples based on BitTorrent, our motivating scenario.

To address Requirement 1, in our design the trace data are stored at four different levels, three corresponding each to one of the system, the swarm, and the peer P2P application levels introduced in Section 2.1, and a fourth level to store data that characterizes the interaction between the P2P application and the resources it uses.

In dealing with Requirement 2, we store at each level both static and dynamic data, separately. Whenever dynamic data are stored, each record in the dataset also includes a time stamp, which represents the latest moment in time at which the measured data value was still valid; usually this is the time of the measurement sample.

For Requirement 3, we use mapping tables to store relationships between data fields in the traces and identifiers (IDs) stored as integral values. When the mapping tables for a converted dataset are not made public, this approach can anonymize the traces, with the notable drawback of losing information; the extent of information loss and the quality of the anonymization form a difficult to analyze trade-off. Due to the use of integral identifiers, the mapping tables greatly reduce the storage requirements and significantly increase the processing speed of the stored data, especially for dynamic data tables.

Next, to address Requirement 4, at each level we distinguish between generic and application-specific data. We design a flexible data format for storing these data (*properties*),

Table 1: Format for application-specific data.

ID	column name	description
1	Time Stamp	Only in dynamic data tables.
2	Composite ID	a tuple value of (<i>Trace ID</i> [, <i>Swarm ID</i> [, <i>Peer ID</i>]]). Trace level data has only Trace ID; swarm level data has both Trace ID and Swarm ID; peer level data has Trace ID, Swarm ID and Peer ID.
3	Category ID	Category of the trace (relates to the Categories mapping table).
4	Property ID	ID of application-specific property (relates to the Properties mapping table).
5	iVal	An integer value of the property.
6	fVal	A float value of the property.
7	sVal	A string value of the property.

which is depicted in Table 1. Application-specific properties in our format can be either fixed or flexible. The fixed properties are defined for each application through a long-term community effort, and represent the expert knowledge of the community. The flexible properties can be defined by anyone who has specific data need not shared with the community; flexible properties may not be comparable between traces. The property definitions define the semantics of the application-specific data, and are stored and exchanged using mapping tables for which the *Property ID* column in Table 1 acts as an index. Each property value is stored using one or more of three data fields, of integer (*iVal*), float (*fVal*), and string (*sVal*) type, respectively. Also in Table 1, the *Time Stamp* column only exists in data tables that store dynamic data, while tables storing static data do not have this field. The introduction of a *Category ID* reflects the presence in P2P applications of different content types, such as movies/games/music for file-sharing; for a row, leaving this field blank means that the values apply for a general, all-encompassing category.

In the case of Requirement 5, we note that the presence of fixed application-specific properties means that our format currently specifies a reduced set of commons among all P2P applications, while in the future the expertise of the community should add to this set a minimal selection of relevant fixed properties.

In the remainder of this section, we introduce in turn the data formats for each of the four levels considered in our format.

3.1 Trace Level

The trace level stores information about the system from which traces were taken, and about the measurement process. Here, the static, generic data contains metadata about the trace, which gives users essential information of the trace and can be used as metrics for the selection of data trace in their work. Other information such as the measurement period, the application from which the trace was taken, the application category, and the authors and copyright information, are also stored in these data.

Dynamic trace-level data are already provided by many P2P systems. In particular, BitTorrent communities such as Pirate Bay publish online statistical information about their

network traffic, dynamically. Thus, these data can be obtained at a trace level, without actually obtaining and then aggregating data from the swarm and peer levels. These data, though lacking detail, allow answering questions about population evolution and about global content popularity.

Table 2: Map between the static, generic properties and their ID, at trace level.

Property ID	Property	Description
100	Type of community	A string value indicating the type of the BitTorrent community, for example, private tracker or public tracker.
200	Measurement Approach	A string value indicating how the trace is collected, e.g. querying trackers, scraping webpages and etc.
300	Data Source	A string value indicating where the trace is collected, e.g. URL of a website.

Table 3: Example table with static, generic properties, at trace level. T_ID, C_ID, and P_ID stand for trace, category, and property ID, respectively.

T_ID	C_ID	P_ID	iVal	fVal	sVal
0	-	100	-	-	Public tracker
0	-	200	-	-	Scraping webpages
0	-	300	-	-	ThePirateBay

Table 4: Map between dynamic, generic properties and their ID, at trace level.

Property ID	Property	Description
500	Number of Peers	A integer value indicating the total number of peers in the trace at certain time.
600	Total Throughput	A float value indicating the total throughput of the system.

Table 5: Example table with dynamic, generic properties, at trace level. T_ID, C_ID, and P_ID stand for trace, category, and property ID, respectively.

TS	T_ID	C_ID	P_ID	iVal	fVal	sVal
1072003000	0	-	500	2000	-	-
1072003000	0	-	600	-	6000.50	-
1072008000	0	-	500	1690	-	-
1072008888	0	-	600	-	7929.80	-

We first present the generic static and dynamic properties at trace level. We depict three fixed properties of static data in Table 2; a sample of their use is depicted in Table 3. The trace has been collected from Pirate Bay, as indicated by the field *sVal* of the row with *PropertyID* = 300. We further show in Table 4 two fixed properties for dynamic data; the number of peers and the total throughput are common to all P2P applications. Table 5 presents an example table storing fixed properties for dynamic data; the *P_ID* field acts as an index in Table 4. The number of peers (*PropertyID* = 500) is stored as an integer value, while the total throughput (*PropertyID* = 600) is stored as a float value.

Table 6: Map between dynamic, application-specific properties and their ID, at trace level.

Property ID	Property	Description
800	Number of Seeders	A integer value indicating total number of seeders in the network.
900	Number of Leechers	A integer value indicating total number of leechers in the network.

Table 7: Example table with dynamic, application-specific properties, at trace level. T_ID, C_ID, and P_ID stand for trace, category, and property ID, respectively.

TS	T_ID	C_ID	P_ID	iVal	fVal	sVal
1072003000	0	-	800	1000	-	-
1072003000	0	-	900	200	-	-
1072006600	0	-	800	750	-	-

Second, we discuss for this level the application-specific properties, both static and dynamic. Table 6 depicts two properties that are common to all P2P applications, and Table 7 depicts the related usage example. The number of seeders (*PropertyID* = 800) decreases from 1000 to 750 in the hour that has elapsed between the consecutive measurements.

3.2 Swarm Level and Peer Level

The swarm-level static properties are intended to provide descriptive information about the swarms present in the trace. For BitTorrent traces, we store as static generic data torrent metadata such as the archive name, the torrent identification number (hash), and the torrent creation time.

Similarly to the trace level, data at swarm level are published online by many P2P systems. For example, many BitTorrent trackers and even web sites publish swarm-level data such as the number of seeders and leechers, and the number of complete file downloads. These data can be used to evaluate the swarm and, aggregated, even the system performance. We store such data in a dynamic generic data format similar with the format used for trace-level dynamic data (see Tables 4 and 5); the only difference is that here the composite ID also includes the the swarm ID.

Table 8: Example maps between peer events and their IDs. (top) Map that is common to all P2P applications. (middle) Map defined by the BitTorrent community. (bottom) Map defined by the information retrieval community.

Property ID	Event
0	Join Swarm
1	Leave Swarm

Property ID	Event
10000	Get Chunk
10001	Connection Refused
10002	Header Incorrect
10003	Connection Timedout

Property ID	Event
20000	Search
20001	Select

Table 9: Example of a peer event table that combines generic and application-specific properties. T_ID, S_ID, and P_ID stand for trace, swarm, and property ID, respectively. The Category ID and the fVal columns do not include any data and are omitted.

TS	T_ID	S_ID	Peer_ID	P_ID	iVal	sVal
972023332	0	1	4415	10001	-	-
972023401	0	1	200	10000	129	-
972023403	0	1	300	10002	-	error reason
972023451	0	2	1100	20000	-	'Britney Spears'
972023571	0	2	1200	20001	-	'Birtney Speersz'

At peer level, we store static generic data such as the port number, and the upload and download bandwidths are stored. We further store as static application-specific data the peer configuration parameters and the software ID. The latter can be used to distinguish between different implementations of the same P2P application, from a community-generated mapping table. Also at peer level, we treat as dynamic properties, either generic or application-specific, the events that involve peers, such as joining or leaving a swarm. Table 8 shows three example mapping tables between peer events and their IDs. The mapping table depicted in the top table is generic, and contains data that can be compared across all P2P applications. The middle and bottom rows of the figures show examples of application-specific mapping tables defined by distinct communities. Table 9 shows an example of storing peer events in a table that combines generic and application-specific peer events.

3.3 Resource Level

The resource level stores information about the resource demands and consumption of the P2P application. For example, in a P2P file-sharing system the network resources receive data packets of specified size, and the packet submission and delivery times define the resource consumption. Similarly, in a P2P grid computing system the computational resources receive jobs with specific requirements about the memory size and CPU speed, and the job start and finish time define the resource consumption. Such data are useful not only for the P2P scientist, who can validate design choices against real constraints, but also for the community interested in the resource used by the P2P application (the resource community), for our example, the networking community and the grid computing community, respectively.

Data for this level already have already been collected, either by the P2P community [27] or by the resource community [5, 19]. To promote cooperation between communities, we use at this level the data exchange formats that have become the official or the de-facto standards for the resource community, such as the GWA format [18] for the grid community.

3.4 Data Storage

The data converted in the unified format can be stored in plain text and XML files, in a relational database, or in another format. In our design, the same data should be stored in two formats, plain text and an SQL-compatible relational database, which are logically identical. The plain text format follows the "one information unit per line" phi-

Table 10: Characteristics of the experimental traces.

Trace ID	BT-TUD-1	BT-TUD-2
Measure	Value	Value
Tracing Period	Dec 2003-Jan 2004	May 2005
Tracing Duration	1 month	1 week
Recorded events	34.83 M	36.02 M
No. of files	12	2000
No. of unique users	159,851	229,410
Observed traffic	177.98 TB	193.78 TB

losophy that is common in web [28, 29] and cluster/grid logging, and which greatly simplifies parsing. The SQL format greatly facilitates extracting basic statistics from the traces. It is straightforward to convert the plain text format into an XML representation of the same data. We see a format with a fixed number of fields as a feature rather than a limitation: a community using this format will have to select the most important features into a fixed set, which will enable comparable studies for that community.

Having two data storage formats follows our previous experience with the Grid Workloads Archive, where the plain text files were preferred as input sources for trace-based simulation, and the SQL format was used to select traces with interesting characteristics for a specific experiment or scenario. A complete description of either format is beyond the space-availability of this paper.

4. SAMPLE RESULTS

In this section we present sample results to show evidence that large traces taken from P2P systems can be converted into our proposed unified format, and to demonstrate the format's usefulness for performance studies.

4.1 Converting and Comparing P2P Traces

To demonstrate the ability of our unified format to accommodate traces taken from real P2P systems we have used two measurement traces taken from the BitTorrent P2P file-sharing network: BT-TUD-1 from a measurement done in 2003-2004, and BT-TUD-2 from a measurement done in 2005. Both traces are the result of large-scale measurements and have been analyzed in detail elsewhere [17, 25]. While the measurements were performed by very similar teams, they used different toolsets and produced different data formats. We have successfully converted both traces to our unified format.

The conversion allows us to give an example of comparing converted traces, which is one of the main benefits of using a unified format. Table 10 summarizes a comparison of the trace-level characteristics for the two converted traces. Both traces include millions of performance-related events and hundreds of thousands of recorded users; the main traced activity is the transfer of close to 200 TB! of data for each trace. However, the two traces are very different from a performance study perspective. The BT-TUD-1 trace is four times longer, which makes it useful for extracting long-term behavior such as weekly usage patterns and flashcrowds. The BT-TUD-2 trace studies an order of magnitude more files, which makes it useful for performance studies in which the taste of users plays a role, and in particular for scenarios where imbalance in interest leads to server overload. Even this simple head-to-head comparison would not have been

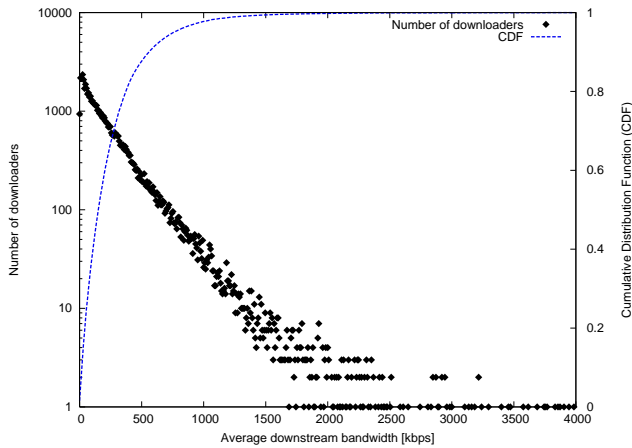


Figure 1: The average download speed of SuprNova users downloading content through BitTorrent. (Data from trace BT-TUD-1.)

possible without having access to data presented in a similar format. While most of these characteristics could have been obtained from published analysis reports, this comparison would have been difficult, since some of the information was not included in one or both of the related reports. A similar situation occurs for many other reports, since most research on P2P trace characteristics targets previously unreported aspects.

4.2 Application-Level Performance

It is convenient to extract complex information from datasets stored in the unified format. To show evidence of this, we present now the results of the analysis of application-level performance for BT-TUD-1 and BT-TUD-2. We follow two performance metrics, the average and the cumulative application-level throughput. We define the *average application-level throughput* for a peer the average download bandwidth observed for that peer; we have computed this metric for both individual peers and entire swarms (as an average of averages). We define the *cumulative application-level throughput* for a peer as the total amount of useful traffic (only file chunks, no overhead data) generated by that peer.

The average application-level throughput for the two measurements is shown in Figures 1 and 2. The most important observation is that the application-level throughput more than doubled between the two measurements, from 240 KBps to 500 KBps. Figure 2 also indicates that for swarms with size from 10 to around 170 peers the application-level throughput does not depend on the size of the swarm, which confirms the findings of the “fluid model” for BitTorrent [26]. However, for larger swarms this seems to no longer be the case. We attribute this situation to a lack of statistically meaningful data (few of the swarms included in this measurement were this big), and to the dynamic peer arrival and departure that was not accounted for in the “fluid model”.

The properties of the cumulative application-level traffic are summarized in Figure 3. Over 50% of the users generate useful download traffic of over 1 GB. Over 95% of the users generate useful download traffic of up to 2-2.3 GB. The transition to and smaller mode that occurs around 3 GB are responsible for another 4% of the population. We conclude that the average BitTorrent peer has important network requirements.

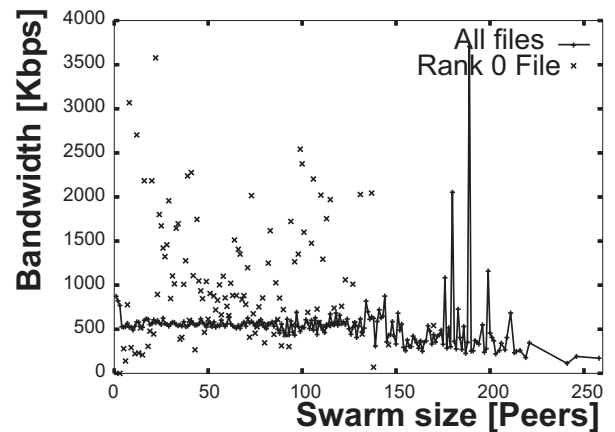


Figure 2: The average download speed of Pirate Bay peers downloading content through BitTorrent vs. swarm size. The average download speed of individual peers downloading the same file, “Rank 0”, is also depicted. (Data from trace BT-TUD-2.)

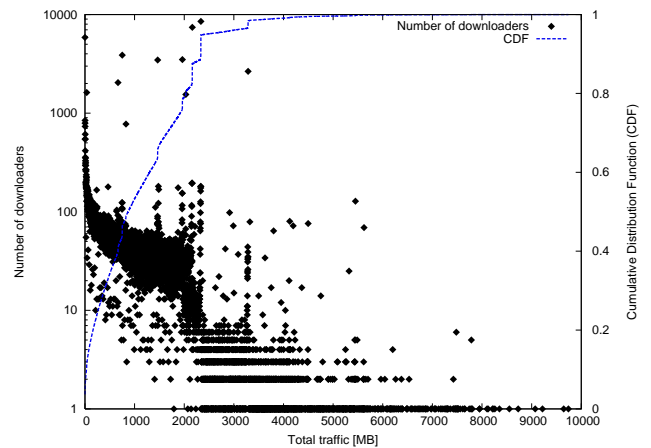


Figure 3: The total useful traffic of Pirate Bay users downloading content through BitTorrent. (Data from trace BT-TUD-2.)

5. TOWARDS A P2P WORKLOADS ARCHIVE

In this section we describe a path towards a P2P Workloads Archive, that is, a community public-access archive with traces of different P2P systems presented in a unified format. We have already taken a similar path for Grid Workloads Archive, which stores traces of grid computing systems. However, a P2P Workloads Archive presents numerous new challenges. In the following we raise three research questions that need to be answered before creating it.

Research Question 1: How to store multiple converted traces that belong to the same logical dataset? Measurements usually result in the acquisition of several traces that belong to the same logical dataset. For example, we have collected traces corresponding to many BitTorrent swarms that belong to the same community [17, 25]. One way to convert these raw traces into the unified format is to convert all the obtained traces into one, albeit large, unified format trace. For plain text storage, the size of this file is for our case (2,000 swarms followed for less than a week

generated around 80 GB of uncompressed raw data) is 1.5 GB. Another way is to convert each obtained trace into one unified format trace, thus having a much smaller size per converted trace. Metadata, that is, data that is not stored directly in the converted traces, can be used to gather these smaller traces into datasets; this is the approach taken by CRAWDAD [1] for wireless network traces.

Research Question 2: How to derive automatically statistics from stored traces? Statistics are commonly derived from individual or groups of traces, to create and validate empirical data models or as direct input for experimental work. A community trace archive should extract and present online the common statistics, and also make available the tools needed to extract them elsewhere [18]. However, the problem of defining a common set of derived statistics for P2P systems remains open. A similar, application-specific sub-problem is raised for file sharing; finding the appropriate tools would simplify a comparison between P2P file-sharing systems based on practical results.

Research Question 3: How to anonymize peer-to-peer traces without losing too much information? (suggested by the LSAP reviewer Jon Howell) The trade-off between the quality of anonymization (e.g., resistance to correlations between several traces) and the usefulness of the anonymized data must be investigated. Our proposed first step, to only strip away the most sensitive part of the information, may be subject to correlation attacks [8], and may also result in making the traces unusable (when the stripped information was in fact needed).

Research Question 4: How to analyze the large volume of data present in stored traces? Even when common statistics can be obtained from the archive, much further or different analysis is possible using trace data. For instance, evaluating the dynamics of millions of peers can give insights into the demand and performance of the network. However, given the sheer scale, suitable algorithms must be found or designed to process these data. The related research question How much data are enough for a statistically meaningful result?, which would help reducing the size of the problem, has been raised and addressed by the Internet community [21], but not for P2P systems.

Research Question 5: How to find the best trace for a specific scenario? We have already found in our previous work with the Grid Workloads Archive that trace characterization using few or even a single number, and tools to automatically select trace parts that have specific properties are necessary to address this research question. However, it is not clear how to extend our previous answers for the heterogeneous and large-scale P2P applications.

6. RELATED WORK

In this section we survey several approaches to define unified formats in computer science areas, e.g., Internet, clusters, grids. All these approaches use formats that may be suitable for their domain, but cannot be used for storing traces of general P2P systems. Thus, our work complements these approaches for the broad area of P2P systems.

In 1995, the Internet community assembled the first publicly and freely available workload archive: the Internet Traffic Archive (ITA) [11]. ITA stores packet-level information using a format derived from the output of tcpdump, a common tool for packet-level trace capture. The Internet community has since created several other archives, such as the

CAIDA archives [9, 12, 22]. The CAIDA archives are combined the largest source of Internet traces. CAIDA uses a large number of formats, usually related to the measurement tool's output; no unified format for the CAIDA data exists.

The computer architectures community started a first and so far the most successful database (BYU) at the end of the 1990s. The BYU stores traces of instructions for several machines in a fixed format. For the cluster-based communities, the Parallel Workloads Archive (PWA) has become the de-facto standard for the parallel production environments community. Similarly, the Grid Workloads Archives (GWA) is now the largest archive of grid computing traces. The PWA and the GWA define each their own formats [10, 18]; these formats could correspond to the resource level in our proposed format. Even by using the extensibility hooks provided by the GWA format, it would be difficult to extend the archive to include P2P system traces, with the notable exception of traces from P2P systems with grid computing application, such as traces taken from the grid system with a peer-to-peer architecture OurGrid [5]. The CRAWDAD [1] archive of wireless data offers traces for a wide range of wireless protocols. The archive imposes a structure on the metadata (i.e., the description of the data) but there is no common template shared by the raw traces.

The UMass Trace Repository [2] provides a collection of network, storage, and resource usage traces. The repository includes data collected from a campus network for P2P file sharing based on the OpenNap server and logs extracted from two large BitTorrent trackers. Traces in the UMass repository have been collected by multiple research groups and are stored in raw formats which have not been unified in any way.

We have made available online since 2006 one of the datasets that we have collected and analyzed prior to this work [17]. However, the presentation format suffers from the same problem encountered for public datasets of other systems: it is closely tied to the application (BitTorrent) and to our proprietary measurement tools, and therefore difficult to generalize and even to parse.

7. CONCLUSION AND ONGOING WORK

Peer-to-Peer systems have recently emerged as novel Internet applications that serve millions of users world-wide. Despite their popularity, monitoring and measurement data taken from P2P systems are rare. As a result, scientists and system administrators find it difficult to conduct meaningful experiments, and system design and tuning are severely hampered. To address this situation, in this work we introduce a unified format for trace of peer-to-peer systems that gives special attention to the complex structure and heterogeneity of P2P applications, and to their sensitivity to making public personal information. We show that the format is useful by converting to it two large traces taken from real P2P systems, and by subsequently presenting analysis results that benefit from a unified data format.

We are currently converting more traces taken from P2P file-sharing systems to our format. For the future we intend to convert traces taken from other P2P application areas, such as video streaming, grid computing, and massively multiplayer online games; we invite the community to contribute with raw data to this effort.

8. DATA AVAILABILITY

Technical documentation on the format and the data used in this work are available at:

<http://www.p2ptraces.org>

Acknowledgements

We would like to thank the LSAP reviewers, both John Howell and the anonymous reviewers, for their very useful comments.

The research leading to this contribution has received funding from the European Community's Seventh Framework Programme in the P2P-Next project under grant agreement no 216217, and from the Delft ICT Talent Grant 2008.

9. REFERENCES

- [1] CRAWDAD data set. [Online] Available: crawdad.cs.dartmouth.edu.
- [2] UMass trace repository. [Online] Available: traces.cs.umass.edu/index.php/Network/Network.
- [3] ipoque internet studies, 2006-2009. [Online] Available: www.ipoque.com/resources/internet-studies/.
- [4] The Parallel Workloads Archive. [Online] <http://www.cs.huji.ac.il/labs/parallel/workload/>, Jul 2007.
- [5] N. Andrade, W. Cirne, F. V. Brasileiro, and P. Roisenberg. OurGrid: An approach to easily assemble grids with equitable resource sharing. In *JSSPP*, volume 2862 of *LNCS*, pages 61–86. Springer, 2003.
- [6] M. Andreolini, V. Cardellini, and M. Colajanni. Benchmarking models and tools for distributed web-server systems. In *Performance*, volume 2459 of *LNCS*, pages 208–235. Springer, 2002.
- [7] S. Avallone, D. Emma, A. Pescapè, and G. Ventre. Performance evaluation of an open distributed platform for realistic traffic generation. *Perform. Eval.*, 60(1-4):359–392, 2005.
- [8] M. Barbaro and T. Zeller. A face is exposed for AOL searcher no. 4417749. New York Times article, Aug 9 2006. [Online] Available: <http://www.nytimes.com/2006/08/09/technology/09aol.html>.
- [9] CAIDA Team. The Cooperative Association for Internet Data Analysis, Mar 2009.
- [10] S. J. Chapin, W. Cirne, D. G. Feitelson, J. P. Jones, S. T. Leutenegger, U. Schwiegelshohn, W. Smith, and D. Talby. Benchmarks and standards for the evaluation of parallel job schedulers. In D. G. Feitelson and L. Rudolph, editors, *JSSPP*, volume 1659 of *LNCS*, pages 67–90. Springer, 1999.
- [11] P. Danzig, J. Mogul, V. Paxson, and M. Schwartz. The Internet Traffic Archive, Mar 2009.
- [12] DatCat Team. The DatCat Internet Measurement Data Catalog, Mar 2009.
- [13] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *ACM Symp. on Operating Systems Principles (SOSP)*, 2003.
- [14] A. Iammitchi, S. Doraimani, and G. Garzoglio. Filecules in high-energy physics: Characteristics and impact on resource management. pages 69–80, 2006.
- [15] A. Iosup, C. Dumitrescu, D. H. Epema, H. Li, and L. Wolters. How are real grids used? The analysis of four grid traces and its implications. In *IEEE Int'l. Conf. on Grid Computing (GRID)*, pages 262–269, 2006.
- [16] A. Iosup and D. H. J. Epema. GRENCHMARK: A framework for analyzing, testing, and comparing grids. In *IEEE/ACM Int'l. Symp. on Cluster Computing and the Grid (CCGrid)*, pages 313–320, 2006.
- [17] A. Iosup, P. Garbacki, J. A. Pouwelse, and D. H. J. Epema. Correlating topology and path characteristics of overlay networks and the internet. In *IEEE/ACM Int'l. Symp. on Cluster Computing and the Grid (CCGrid) Workshops, GP2PC*, page 10, 2006.
- [18] A. Iosup, H. Li, M. Jan, S. Anoop, C. Dumitrescu, L. Wolters, and D. H. J. Epema. The Grid Workloads Archive. *Future Generation Comp. Syst.*, 24(7):672–686, 2008.
- [19] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is P2P dying or just hiding? In *Globecom*, Dallas, Texas, USA, Dec 2004.
- [20] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Infocom*, 2004.
- [21] S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer. Building a time machine for efficient recording and retrieval of high-volume network traffic. In *Internet Measurement Conference (IMC)*, pages 267–272. USENIX Association, 2005.
- [22] MOAT Team. The NLANR Measurement and Network Analysis Group, Mar 2009.
- [23] J. Mol, J. Pouwelse, D. Epema, and H. Sips. Free-riding, fairness, and firewalls in p2p file-sharing. In *IEEE Int'l. Conf. on Peer-to-Peer Computing (P2P)*, pages 301–310, 2008.
- [24] A. Parker. The True Picture of Peer-To-Peer File-Sharing, 2005. Panel Presentation, IEEE Int'l. Workshop on Web Content Caching and Distribution.
- [25] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *IPTPS*, volume 3640 of *LNCS*, pages 205–216. Springer, 2005.
- [26] D. Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM*, pages 367–378, 2004.
- [27] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. of ACM SIGCOMM IMW*, pages 137–150, 2002.
- [28] W3C. Extended Log File Format. W3C Working Draft, 1996. [Online] Available: w3.org/pub/WWW/TR/WD-logfile.html.
- [29] Apache. Log Files, Common Log Format. Apache HTTP Server Version 1.3 Documentation. [Online] Available: <http://httpd.apache.org/docs/1.3/logs.html>.