InTraBase: Integrated Traffic Analysis Based on a Database Management System

M. Siekkinen, E.W. Biersack, G. Urvoy-Keller
Institut Eurécom
BP 193
06904 Sophia-Antipolis Cedex
France
{siekkine,erbi,urvoy}@eurecom.fr

V. Goebel, T. Plagemann
Department of Informatics
University of Oslo
P.O. Box 1080 Blindern
NO-0316 Oslo, Norway
{goebel,plageman}@ifi.uio.no

Abstract

Internet traffic analysis as a research area has attracted lots of interest over the last decade. The traffic data collected for analysis are usually stored in plain files and the analysis tools consist of customized scripts each tailored for a specific task. As data are often collected over a longer period of time or from different vantage points, it is important to keep metadata that describe the data collected. The use of separate files to store the data, the metadata, and the analysis scripts provides an abstraction that is much too primitive: The information that "glues" these different files together is not made explicit but is solely in the heads of the people involved in the activity. As a consequence, manipulating the data is very cumbersome, does not scale, and severely limits the way these data can be analyzed.

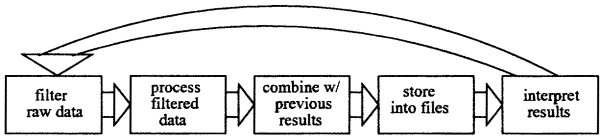
We propose to use a database management system (DBMS) that provides the infrastructure for the analysis and management of data from measurements, related metadata, and obtained results. We discuss the problems and limitations with today's approaches, describe our ideas, and demonstrate how our DBMS-based solution, called InTraBase, addresses these problems and limitations. We present the first version of our prototype and preliminary performance analysis results.

1. Introduction and Motivation

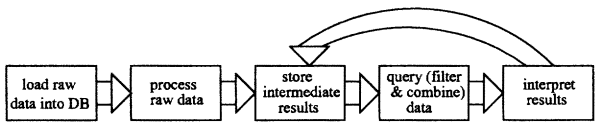
Internet traffic analysis as a research area has experienced rapid growth over the last decade due to the increasing needs caused by the massive growth of traffic in the Internet together with new types of traffic generated by novel applications, such as peer-to-peer. Today, the state of the art in traffic analysis is handcrafted scripts and a large number of software tools specialized for a single task. The amount of data used in traffic analysis is typically very large. Also, traffic analysis normally is an iterative process: A first analysis is performed and based on the results obtained, new analysis goals are defined for the next iteration step. These facts lead to the following three problems:

Management: Many tasks are solved in an ad-hoc way using scripts that are developed from scratch, instead of developing tools that are easy to reuse and understand. Another problem with today's traffic analysis is the huge amount of data generated. Trace files with "raw" data, files containing intermediate and final results

0-7803-9249-3/05/$20.00 ©2005 IEEE



(a) Typical Case



(b) Using DBMS-based Approach

Figure 1: Cycles of Tasks for the Iterative Process for Off-line Traffic Analysis.

need to be properly annotated and archived in order to be able to use them at a later point in time. However, the tools used generally do not provide any support for managing these large amounts of data. Therefore, trace files are typically archived as plain files in a file system. Depending on the number of files and the skills of the researcher to properly organize them, the later retrieval of a particular trace or data item may be a non-trivial problem. As Paxson [18] has pointed out the researchers themselves often cannot reproduce their own results.

Analysis cycle: A common workflow to analyze network traffic proceeds in cycles (see Figure 1(a)). Let us take as example our own analysis of BitTorrent [13], a peer-to-peer system for file distribution. When following the analysis steps, one can identify three iterations of the analysis. In a first iteration, we studied the global performance of BitTorrent in terms of how many peers succeeded downloading the complete file. From the results, we noticed a large flash-crowd of peers arriving at the beginning. In a second iteration, the performance of individual sessions was studied. First, the raw data was analyzed on the basis of individual sessions that either had successfully completed the file download or aborted. In the next step, the performance of the individual sessions in both categories was computed. The information from the previous cycle was combined to obtain average performance of a session during the flash-crowd period and after. In a last iteration, we considered the geographical location of the clients that successfully completed their download to study download performance per geographic region. Since the semantics of the data were not stored during the analysis process, reusing intermediate results (e.g. to integrate geographic information) turned out to be cumbersome and most of the time the data extraction had to be done again starting from the raw data after modifying the scripts.

Scalability: Scalability is an important issue in traffic analysis and poses a problem as the amount of data is typically very large. Often tools are first applied to

process small amounts of data. If then applied on large data sets, it often turns out that the run-time or memory requirement of the tool grows *more than linearly* with the amount of data, in which case modifications and heuristics are introduced that often sacrifice quality of the analysis for performance. An example is the well-known tool `tcptrace` [6] that can, among others, be used to extract an individual flow from a large amount of trace data collected via `tcpdump`*.

The above mentioned issues are also in some extent discussed in [14] and [11]. These problems lead us to investigate whether database management systems (DBMSs) might ease the process of analyzing traffic. Traditional database systems (DBSs) have been used for more than 40 years for applications requiring persistent storage, complex querying, and transaction processing of data. Furthermore, DBMSs are designed to separate data and their management from application semantics to facilitate independent application development. Internet traffic consists of well-structured data, due to the standardized packet structures, and can therefore easily be handled with a DBMS.

We present later the first prototype of the InTraBase, an implementation of our DBMS-based approach. The goal is to devise an open platform for traffic analysis that would facilitate the researchers' task. Our solution should:

(i) conserve the semantics of data during the analysis process;
(ii) enable the user to easily manage his own set of analysis tools and methods and;
(iii) share them with colleagues;
(iv) allow the user to quickly retrieve pieces of information from analysis data and simultaneously develop tools for more advanced processing.

The remainder of this paper is structured as follows: In the next section, we review some related work. In Section 3, we present our InTraBase approach. In Section 4, we describe the first prototype of InTraBase and in Section 5, we show some measures of its performance and make a comparison to another tool called `tcptrace`. Finally, we draw some conclusions and discuss open issues in the last section.

2. Related Work

Table 1 summarizes the differences between the various existing approaches and our InTraBase approach.

Data, metadata, and software management are related to the problems of management and analysis process cycle (see Section 1). Because publicly available solutions are generally more interesting for the research community, we include public availability as a metric. Integrated approach means that in addition to data, metadata and software are also managed in an integrated way. It is a feature of our approach only, which will be described in Section 3. Finally, the capability to analyze traffic on-line is the last feature that we consider. By on-line analysis we mean the ability to per-

*Due to the focus of our research and for the sake of illustration, the examples given usually refer to the collection and analysis of TCP related data. However, we do not want to imply that our approach is limited to this kind of study.

Table 1 Characteristics of Different Approaches for Traffic Analysis.

Approach		Data mgt	Metadata mgt	SW mgt	Scalable	Publicly available	Integrated approach	On-line
Ad-hoc scripts								
Specialized tools (tcptrace [6])						X		(X)
Toolkits (CoralReef [15])				X		X		(X)
ISP database projects	Sprint IPMon [9]		X		X			
	Gigascope [8] (AT&T)			X	X			X
	Internet Traffic Warehouse [7] (Telcordia)	X	X	X	X			
InTraBase		X	X	X	X	X	X	

X = feature is supported in the approach
blank = feature is not at all supported or is implemented in an ad-hoc manner
(X) = feature is supported in some members of the category

form the analysis tasks on a continuous stream of traffic, i.e. to process the input data at a rate equal to its arrival rate.

The first approach is ad-hoc scripts. However, this approach does not have any of the characteristics we look for. The next step forward are the specialized tools such as `tcptrace` [6] which allows to analyze a `tcpdump` trace and produce statistics or graphs to be visualized using the xplot software. Still none of the important management issues are considered by these tools.

There have been some efforts toward complete analysis toolkits that are flexible enough to be used in customized ways. One example is the Coralreef software suite [15] developed by CAIDA, which is a package of device drivers, libraries, classes and applications. The programming library provides a flexible way to develop analysis software. The package already contains many ready-made solutions. The drivers support all the major traffic capturing formats. This approach concentrates on the software management aspect but addresses neither the problem of handling nor managing the data, i.e. source data and results, nor managing related metadata. Also scalability is an issue.

There exist some larger projects than those mentioned so far for traffic analysis. They usually involve huge amounts of traffic data, and therefore, require a lot of attention to the organization and handling of the data, i.e. raw traffic data, associated metadata, and derived analysis data. We classify these approaches as "ISP database projects" since they are tailored to the needs of large ISPs and, as it turns out, two of the three approaches we consider are developed by a large ISP. Unfortunately, none of them is publicly available.

Sprint labs initiated a project called IP Monitoring Project (IPMON) [10], [9] to develop an infrastructure for collecting time-synchronized packet traces and routing information, and to analyze terabytes of data. In their architecture, a DBS is used for metadata management only and metadata is stored about raw input data sets,

analysis programs, result data sets, and analysis operations. Details about metadata management can be found in [17]. IPMON has adopted CVS for managing software.

Gigascope [8] is a fast packet monitoring platform developed at AT&T Labs-Research. It is not a traditional DBMS but a Data Stream Management System (DSMS) that allows on-line analysis of traffic arriving at high rates. As a powerful DSMS, Gigascope can handle a high rate traffic stream in real-time. However, the real-time requirements imply that the input data are processed in one pass, what evidently imposes limits on the operations that can be performed. We refer the reader to [19] for a detailed assessment of the suitability of DSMSs for traffic analysis. Gigascope is specialized for network monitoring applications such as health and status analysis of a network or intrusion detection. Gigascope does not manage data nor metadata, which are typically managed via a separate data warehouse. Gigascope has a registry for query processes that are providing output streams according to the associated query. The user can also define his own functions and data types for the queries. In this way, Gigascope addresses the software management problem.

We wish to perform complex traffic analysis tasks, such that cannot be performed with a single pass over the input data. For this, we need to be able to make multiple iterations over the analysis process cycle, like in Figure 1(a), which is impossible with a DSMS. The work related closest to our approach is the Internet Traffic Warehouse [7], which is a data warehouse for managing network traffic data built by Telcordia. Analysis results on application level are provided by storing application information about traffic in addition to IP packet headers. Using a suite of programs, input traffic traces are filtered and parsed into temporary files, which are subsequently loaded into the data warehouse. This system is proprietary and mainly aimed for ISPs and especially suitable for accounting. We, on the other hand, target researchers on the field of traffic analysis with a publicly available and extendible solution. In addition, we would like to do filtering and parsing operations *within* the DBS and preserve the raw data stored in the database as unchanged as possible.

3. IntraBase Approach

3.1 Fully Integrated DBMS-Based Solution

We advocate a DBMS-based approach for traffic analysis. While some research groups have already proposed the use of DBMS techniques to process Internet traffic (see previous section), our approach is different. First of all, we **completely** manage the collected data within the DBS. In other words, we process the "raw" data as little as possible prior to loading it into the database.

A high-level architectural view of our solution is shown in Figure 2. We store data from different sources into the DBS. The data uploaded into the database is referred to as *base data*. Examples of base data are packet traces collected via `tcpdump` or a similar tool but also data obtained by instrumenting an application or time series created with the help of Web100 [16] that allows to track precisely the state of a TCP connection.

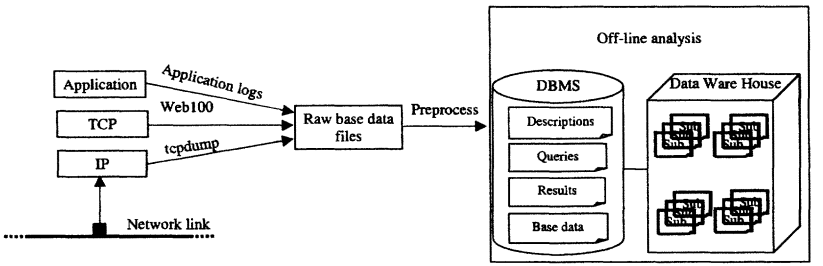


Figure 2: High-level Architecture of the DBS.

Once the base data is uploaded into the DBS, we process it to derive new data that is also stored in the database. For instance, we demultiplex the `tcpdump` packet traces into connections by assigning a connection identifier to each packet.

All the processing is done **within** the DBS. Details are given in Section 4. We want a DBS that not only contains all the data but also contains reusable *elementary functions* and more complex tools built on top of the elementary functions, as illustrated in Figure 3. The boxes on the lowest layer represent the base data uploaded into the database. The middle layer contains the elementary functions that primarily process the base data and create new data. Finally, the highest layer represents tools for more complex analysis tasks. (We refer the reader to [21] for details about T-RAT.) A given component commonly utilizes some of the components at the lower layers. These relationships are described with arrows.

3.2 Benefits From Our Approach

An obvious advantage of using a DBS is the support provided to organize and manage data and related metadata. As far as the analysis cycle is concerned, once the base data is in the DBS, it becomes structured data and, therefore, processing and updating becomes easier. Also this makes searching easy and allows for complex queries. Modern DBSs also support the use of indexes to speed up lookup operations.

From Figure 3, one can identify two additional strong points of a DBMS-based approach. First, it is possible to easily combine different data sources provided that time synchronization issues between different recorded base data are solved. For example, it is realistic to assume that application layer events explain some of the phenomena in the traffic observed at TCP layer. One can issue joint queries on the tables holding application layer events and TCP traffic to extract the necessary information. Second, since the InTraBase consists of reusable components, implementing new tools will be less laborious and error prone.

Figure 1(b) describes the common analysis cycle with our DBMS-based approach. If we compare to Figure 1(a), we can see that the initial processing steps performed by the DBS to store data derived from the base data shorten the analysis cycle. Filtering, combining, and aggregating data are operations performed automatically by

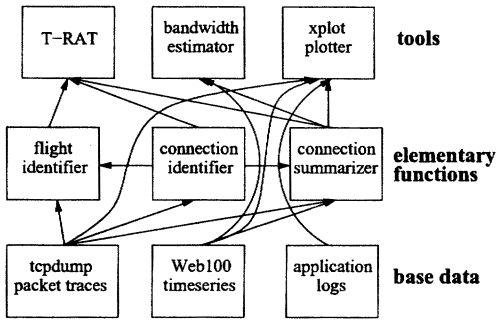


Figure 3: Integrated Data and Tool Management.

the query processor of a DBS, which makes this cycle shorter than the one in Figure 1(a).

DBSs are designed to handle very large amounts of data and there exists a large body of knowledge and literature about performance tuning, which is essential for good performance [20].

4. The First Prototype of InTraBase

We built a first prototype for analyzing TCP traffic from `tcpdump` packet traces with PostgreSQL, which is an open source DBMS that has a widespread user community.* The main reason for choosing PostgreSQL [3] is its object-relational nature that allows to extend the functionality by adding new programming language bindings, called loadable procedural languages (PL). After adding a binding, one can implement external functions in well-known programming languages, such as Perl or Python, which is impossible with standard relational DBS like MySQL. We are using currently PL/pgSQL [2] and PL/R [1]. PgSQL is specifically designed for PostgreSQL and R [4] is a language and environment for statistical data analysis and visualization.

The core tables used in InTraBase are described in Figure 4. The table *traces* contains annotations about all the packet traces that are uploaded in the database. The *packets* table holds all packets for a single trace. The two tables *connections* and *retransmissions* hold connection level summary data for all traces. The *cnxid* attribute identifies a single connection in a packet trace, *reverse* differentiates between the two directions of traffic within a connection, and *tid* identifies a single trace. *Cid2tuple* is a table to store a mapping between unique *cnxid*s and 4-tuples formed by source and destination IP addresses and TCP ports. The attributes of the *packets* table are directly from the standard output of `tcpdump` for TCP packets. The attributes of the

*Note that this prototype focuses only on the analysis of TCP traffic. However, we do not want to imply that our approach is limited to this kind of study but the prototype can be extended to support UDP traffic, for instance.

other tables were chosen so that the connection level information roughly covers that given by `tcptrace`.

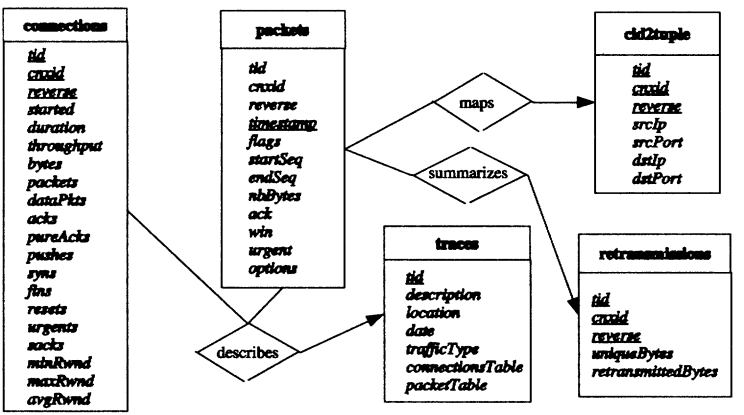


Figure 4: The Layouts of Core Tables in InTraBase After the 5 Processing Steps. Underlined Attributes Form a Key That is Unique for Each Row.

Processing a `tcpdump` packet trace with InTraBase includes five major steps:

1. Copy packets into the *packets* table in the database;
2. Build an index for the *packets* table based on the connection identifier;
3. Create connection level statistics from the *packets* table into the *connections* table;
4. Insert unique 4-tuple to *cnxid* mapping data from *packets* table into the *cid2tuple* table;
5. Count the amount of retransmitted bytes per connection from the *packets* table and insert the result into the *retransmission* table;

Step1, copying packets into the *packets* table is done as follows: `tcpdump` is used to read the packet trace file and the output is piped through a filter program to the database. The filter program's primary task is to make sure that each line of text, i.e. each packet, is well-structured before uploading it into the database. More specifically, each line of text representing a TCP packet contain all the attributes defined in the packet table. If an attribute is missing, the filter program adds a special character signifying that its value is null. For example, pure acknowledgments do not have starting and ending sequence numbers and the filter program inserts null values for them. The second task of the filter program is to add a unique connection identifier and a reverse attribute for each packet. A connection identifier is unique for each 4-tuple. (More information about using the 4-tuple as unique identifiers for connections is given in Section 5.2.)

The remaining four processing steps are performed with SQL queries. It would be logical to have the retransmission data created in step 5 in the same table with the

other connection level statistics created in step 3, but the need to use separate SQL queries to create these two sets of data forces us to use separate tables. In Figure 4, the table *packets* does not contain the 4-tuple attributes and, in fact, the reason for performing the processing step 4 is that we can drop the 4-tuple attributes data from the *packets* table, which saves disk space because we only store the 4-tuple twice per connection (both directions) instead of once for each packet.

After the five processing steps the tables in Figure 4 are populated with data from the packet trace and the user can either issue standard SQL queries or use a set of functions provided for more advanced querying on the uploaded data. Alternatively, the user may develop his own functions. The schema shown in Figure 4 enables the user to limit himself on connection level analysis but also drill down to per-packet analysis.

We have implemented functions in procedural languages to perform operations that cannot be done with plain SQL queries. Currently, we have used the PL/pgSQL language to write functions for operations such as creating graphs in xplot format and producing timeseries of throughput, packet inter-arrival times, jitter, retransmitted packets etc. We have also used it to write functions that perform analysis tasks on a packet trace. For example, using specific timeseries functions, we separate for all connections in a given packet trace all bulk transfer periods from inactive periods where the application operating on top of TCP is silent. PL/R has been used to write functions that produce graphs and do statistical calculations.

We released the first public prototype of InTraBase March 1, 2005 (accessible at http://metrojeu2.eurecom.fr:8080/intrabase.html). It enables the user to install the functionality of InTraBase into a PostgreSQL DBMS and process `tcpdump` packet traces as described earlier in this section. Note that, compared to Figure 2, we only support processing of `tcpdump` base data in this prototype.

5. Evaluation of Our Prototype

We evaluate two aspects of our prototype: (i) feasibility in terms of processing time and disk space consumption and (ii) qualitative and quantitative comparison of our approach with the `tcptrace` tool.

5.1 Analysis of Processing Time and Disk Space Consumption

We conducted measurements in order to evaluate whether a DBMS-based solution scales in an acceptable way, i.e. has a reasonable processing time for large files and acceptable disk space overhead. We processed the five different steps described in Section 4 using different size `tcpdump` packet trace files of two different types of traffic: BitTorrent traffic and mixed Internet traffic. BitTorrent, as a peer-to-peer file distribution system, tends to produce long-lived and large connections in terms of transferred bytes. A typical mixture of Internet traffic, on the other hand, contains many short connections, the so-called mice, and few long-lived connections, the so-called elephants [12]. Consequently, a BitTorrent trace file contains fewer connections than a mixed Internet trace file of same size. For example, the 10 Gigabyte files of BitTorrent and mixed Internet traffic used contain 53,366 and 1,709,993 con-

nections, respectively. As most of the processing is done on a connection basis, we expect the number of connections to have an impact on the performance. Our BitTorrent traffic trace file contains also fewer packets than the mixed Internet traffic trace file of the same size. The reason is that the BitTorrent traffic was captured on a Sun machine where the minimum capture length is 96 bytes whereas the mixed Internet traffic was captured using the default length of 68 bytes. We did our tests using Linux 2.6.3 running on an Intel Xeon Biprocessor 2.2GHz with SCSI RAID disks and 6GB RAM*.

Processing Time

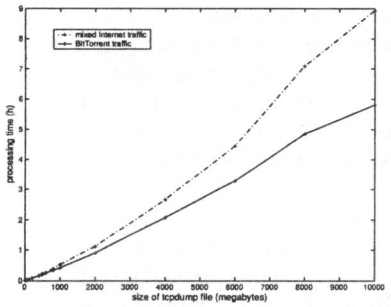
Figure 5(a) shows the evolution of the processing time as a function of the trace file size for BitTorrent traffic and mixed Internet traffic. We can see that for both traces the growth is approximately linear. Processing BitTorrent traffic takes approximately 0.6 hours per Gigabyte and mixed Internet traffic approximately 0.9 hours per Gigabyte. In fact, since eventually each packet in the packet trace needs to be processed, it is impossible to find a solution that scales better than $O(n)$, where n is the number of packets. We can only try to minimize the processing time per packet. In Figure 5(b), the processing times are plotted against the number of packets. As expected, the curves are closer to each other than in Figure 5(a) because the mixed Internet traffic traces contain more packets per Gigabyte than BitTorrent traffic files. Still, clearly the processing time depends on the combination of number of packets and connections. These performance results are good enough in the current stage of our research as we rarely process traffic traces larger than 10 Gigabytes.

We have also analyzed how much processing time each of the five steps requires with respect to different trace file sizes. Since the results from BitTorrent and mixed Internet traffic are very similar, we present in Table 2 only the numbers for BitTorrent. The main observation is that when the file size grows, the fraction of time spent for copying packets becomes smaller at the expense of the other operations, which are performed within the DBS. The reason could be overhead caused by atomicity of transactions enforced by the DBMS. As the operations are performed with a single function call each, the DBMS ensures that each of these operations is an atomic transaction. To achieve this, the DBMS writes all relevant information of the tasks performed within a single transaction to log files. Consequently, the heavier the transaction, the heavier also the task for the DBMS to ensure atomicity. For example, we are performing most of our analysis tasks as large batch jobs on whole traces executed with a single function call. If at any point the execution of the function is interrupted, the DBMS must be able to go back to the state before starting the execution. Therefore, a solution would be to break these functions into several lighter suboperations. This should ease the burden for the DBMSs caused by the atomicity property of transactions. We will further investigate this issue in the future.

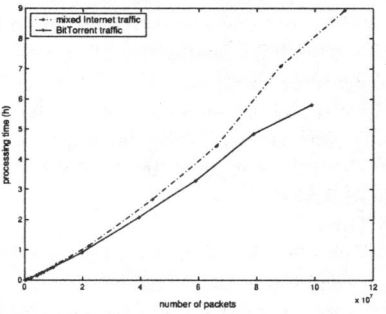
Disk Space Consumption

Modern DBMSs use indexes to speed up operations to access information stored on disk. This introduces an overhead in disk space consumption. Also, data stored

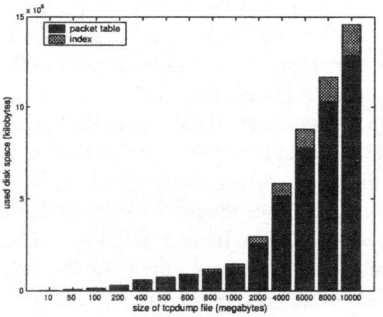
*Unfortunately, PostgreSQL is unable to take advantage of more than 2GB of this memory in certain critical operations such as sorting.



(a) Total Processing Time vs. `tcpdump` File Size.



(b) Total Processing Time vs. Number of Packets.



(c) Disk Space Usage for Different `tcpdump` File Sizes Containing BitTorrent Traffic

Figure 5: Processing Time and Disk Space Usage Measurements for BitTorrent and Mixed Internet Traffic Traces.

in a database takes up more disk space than in a flat file because of different data structures used in the DBSs, e.g. a four byte integer number in a database might not require all the four bytes when stored in a flat file. Moreover, we need to store 2 and 4 bytes fields from the TCP/IP packet headers using 4 and 8 byte data types because PostgreSQL does not support unsigned data types. Finally, we add some data such as connection and table identifiers for each packet but also remove some redundant data by storing only single instances of IP addresses and TCP port numbers. Figure 5(c) visualizes the disk space consumption for the BitTorrent trace. It is the *packets* table that consumes most of the disk space. The sizes of other tables are negligible and are therefore excluded from the figure. The total disk space consumption of

Table 2 Processing Times of Different Steps With Respect to Trace File Size in Percents of the Total Processing Time.

File size	Copying (%)	Indexing (%)	Connections (%)	Cid2tuple (%)	Retransmissions (%)
10 Megabytes	54	<1	23	8	15
100 Megabytes	48	4	19	13	16
1 Gigabytes	44	4	23	13	16
10 Gigabytes	33	9	24	15	18

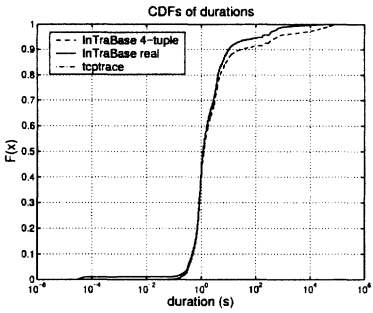
the data in the database is 1.4 to 1.5 times larger than in a flat file. The disk space overhead due to indexes is around 15% from the plain data stored in the database. The results are similar for the mixed Internet trace. A total disk space overhead of 50% is acceptable since nowadays disk space is cheap and rarely an issue. Note that the disks space overhead is the price to pay for having structured data.

5.2 Comparison of InTraBase and Tcptrace

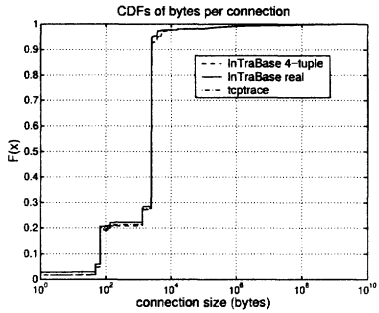
Let us emphasize here that comparing the processing times of `tcptrace` and InTraBase as such is meaningless. For simple tasks with relatively small files `tcptrace` is clearly a better choice. As stated in Section 3.2, the benefits of InTraBase are in the depth of analysis that can be done and its scalability. However, we tried to measure the processing time of `tcptrace` for the mixed Internet trace to have an idea of its behavior compared to InTraBase. As expected, with file sizes up to 4 Gigabytes the processing times were in the order of minutes. However, with file sizes of 6 Gigabytes and larger, `tcptrace` was unable to finish the analysis because after using 3 Gigabytes of memory, it could no longer allocate more.

Because `tcptrace` is among those tools that often need to trade-off accuracy for performance, as discussed in Section 1, it is interesting to compare the results obtained from InTraBase and `tcptrace`. We compare the per connection statistics produced by `tcptrace` and InTraBase. In the case of InTraBase, we consider two different definitions of a connection: the 4-tuple formed by source and destination IP addresses and TCP ports, from now on called 4-tuple connection, and an accurate one, from now on called real connection, where we further separate connections within distinct 4-tuples due to TCP port number reuse. We accomplish this by searching multiple TCP three-way hand shakes, i.e. SYN packet exchanges, among packets sharing a common 4-tuple. The definition of a real connection agrees perfectly with the definition from the specification of TCP [5], and therefore, can be considered as the "correct" one. These two different cases for InTraBase are included to get an idea of how often they differ and in this way to assess the need for separating the connections within a distinct 4-tuple*.

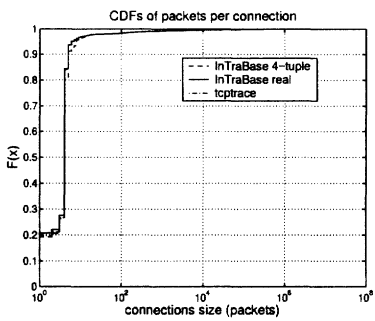
*In the early stage of InTraBase development we considered only the 4-tuple as the identifier of a connection. As it is not entirely accurate, we improved the accuracy. However, this improvement came with a price to pay in performance, and therefore, we wish to assess the need for it.



(a) Durations of connections



(b) Sizes of connections in bytes



(c) Sizes of connections in data packets.

Figure 6: Comparison of Per-Connection Statistics from `tcptrace` and InTraBase.

`Tcptrace` reported statistics from a 10 Gigabyte BitTorrent trace on 55482 connections while InTraBase found 53366 4-tuple connections and 56162 real connections. This shows that `tcptrace` and InTraBase's 4-tuple definition indeed miss some real connections. Cumulative distribution functions (cdfs) of connection durations are plotted for each case in Figure 6(a). The cdfs for `tcptrace` and InTraBase's real connections agree with each other almost perfectly and cannot be distinguished from each other. Between 10 and 10^5 seconds the cdf of durations of the 4-tuple connections deviates from the two other curves. This suggests that the 4-tuple definition captures fewer short connections which is an expected result since some of the 4-tuple connections are in reality several connections due to the reuse of TCP port numbers. However, a look at Figures 6(b) and 6(c), which show the cdfs of connection sizes in bytes and packets, respectively, reveals that the connections missed by

the 4-tuple definition carry negligible amounts of bytes and packets. All in all, the differences between these three sets of connection statistics for the 10 Gigabyte BitTorrent trace seem to be marginal. Therefore, it is justified to simply use the 4-tuple as connection identifier in most of the cases and avoid heavy operations to futher improve the accuracy. We did not conduct the same comparison with the mixed Internet traffic trace since `tcptrace` was unable to process the 10 Gigabyte trace due to memory limitations.

6. Conclusions

Our previous experience of Internet traffic analysis with a plain file-based approach has lead us to the conclusion that for improved manageability and scalability, new approaches are needed. We have advocated to use a DBMS for network data analysis. While this requires an upfront investment to master a DBMS, our initial experience is clearly encouraging in terms of reuse of intermediate data and elementary functions and the type of analysis that becomes possible. We have built a first prototype and conducted measurements of processing time and disk space usage. The results show that our DBMS-based solution has acceptable performance and disk space overhead compared to a plain file-based approach. Furthermore, we showed that in certain cases our solution works where tools such as `tcptrace` reach their limits and are no longer able to operate. There is still a lot of work left such as developing more complex tools, applying the approach to other types of traffic analysis data (e.g. BGP data) and to much larger data sets in the order of Terabytes.

Acknowledgments

The authors would like to thank Karl-André Skevik and Ragnar Nicolaysen for their help in maintaining the database server used to develop and test the InTraBase prototype.

References

[1] "PL/R - R Procedural Language for PostgreSQL: http://www.joeconway.com/plr/".

[2] "PostgreSQL 7.4 Documentation (PL/pgSQL - SQL Procedural Language): http://www.postgresql.org/docs/current/static/plpgsql.html".

[3] "PostgreSQL: http://www.postgresql.org/".

[4] "The R Project for Statistical Computing: http://www.r-project.org/".

[5] "TCP RFC(793): ftp://ftp.rfc-editor.org/in-notes/rfc793.txt".

[6] "Tcptrace: http://www.tcptrace.org/".

[7] C.-M. Chen, M. Cochinwala, C. Petrone, M. Pucci, S. Samtani, P. Santa, and M. Mesiti, "Internet Traffic Warehouse", In *Proceedings of ACM SIGMOD*, pp. 550–558, 2000.

[8] C. D. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk, "The Gigascope Stream Database", *IEEE Data Eng. Bull., 26(1)*, 2003.

[9] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi, "Design and Deployment of a Passive Monitoring Infrastructure", In *Proceedings of Passive and Active Measurements(PAM)*, April 2001.

[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level Traffic Measurement from the Sprint IP Backbone", *IEEE Network Magazine*, November 2003.

[11] J. Gray, D. T. Liu, M. A. Nieto-Santisteban, A. S. Szalay, G. Heber, and D. DeWitt, "Scientific Data Management in the Coming Decade", , Microsoft Research, 2005.

[12] L. Guo and I. Matta, "The War between Mice and Elephants", In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, Riverside, CA, November 2001.

[13] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime", In *Proceedings of Passive and Active Measurements (PAM)*, April 2004.

[14] J. Jacobs and C. Humphrey, "Preserving Research Data", *Communications of the ACM*, 47(9):27–29, September 2004.

[15] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, , and K. Claffy, "The architecture of the CoralReef Internet Traffic monitoring software suite", In *Proceedings of Passive and Active Measurements(PAM)*, 2001.

[16] M. Mathis, J. Heffner, and R. Reddy, "Web100: extended TCP instrumentation for research, education and diagnosis", *SIGCOMM Comput. Commun. Rev.*, 33(3):69–79, 2003.

[17] S. B. Moon and T. Roscoe, "Metadata Management of Terabyte Datasets from an IP Backbone Network: Experience and Challenges", In *Proceedings of Workshop on Network-Related Data Management (NRDM)*, 2001.

[18] V. Paxson, "Experiences with Internet Traffic Measurement and Analysis", Lecture at NTT Research, February 2004.

[19] T. Plagemann, V. Goebel, A. Bergamini, G. Tolu, G. Urvoy-Keller, and E. W. Biersack, "Using Data Stream Management Systems for Traffic Analysis - A Case Study", In *Proceedings of Passive and Active Measurements*, April 2004.

[20] D. Shasha and P. Bonnet, *Database Tuning: Principles, Experiments, and Troubleshooting Techniques*, Morgan Kaufmann Publishers, 2003, ISBN 1-55860-753-6.

[21] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates", In *Proceedings of ACM Sigcomm*, Pittsburgh, PA, USA, August 2002.