Trace Analysis Exploration Using Semantic Web Tools Use Case: You Tube Network Traffic

Oscar Alberto Santana-Alvarez
University of Guadalajara
Zapopan Jalisco México
oscar_santana@cucea.udg.mx

Liliana Ibeth Barbosa-Santillán
University of Guadalajara
Zapopan Jalisco México
ibarbosa@cucea.udg.mx

Gerardo Padilla-Zárate
Intel Corporation. Guadalajara Design Center.
Tlaquepaque, Jalisco, México
gerardo.padilla.zarate@intel.com

***Abstract*—Analysis and exploration of information gathered through local networks are tasks that must constantly be done. Such information is useful for the local network administrators because it gives them a good input about the most effective maintenance procedure for the local network. Maintenance may include activities such as watching over preferences among users, keeping track of the size of the files users are accessing, most viewed videos, etc. It is especially important to watch over Video On Demand (VoD) traffic, such as YouTube-like services providers because of the size of the files they handle and their popularity among users, especially students. One approach to address monitoring activities is network traffic traces, which are sequences of events that recorded specific aspects of a web site. Such traces are usually stored as plain text files (i.e., logs). This paper presents the trace analysis exploration using semantic Web tools, with focus on the You Tube Network Traffic approach, which is based on semantic methods in facilitating network trace analysis by populating two Network Traffic Trace Turtle Files (NTTTF) with network traces obtained by monitoring means. The queries used over the NTTTFs allow us to identify key information presented in the network traffic trace associated with different aspects of our case study. The results showed the feasibility of this approach, where NTTFs improved the way valuable information is being found. Analysis of network traces showed information such as the most viewed videos, the slowest YouTube servers, etc. With this information, more accurate maintenance procedures can be followed. The analysis and exploration of approximately 1,100,000 (one million one hundred thousand) YouTube network traffic traces was performed by means of semantic queries.**

Keywords*—Network Traffic Traces; Semantic Analysis; Local Area Networks.

I. INTRODUCTION

The analysis and exploration of network traces obtained by means of network monitoring is a time consuming task because of the enormous quantity of data and the structure of the files gathered. Besides, human intervention is necessary in order to interpret the information stored in traces. Therefore, it is necessary to implement a framework that support network administrators with the task of finding key information about the use of the local network. This paper presents the exploration and analysis of information stored in network traffic traces. Network traffic traces are data in text format that stores specific aspects of a network. It proposes an approach that facilitates the way network administrators analyze and extract valuable information from network traces by means of semantic web tools. We have the hypothesis that the use of semantic web tools in the analysis and exploration of network traces will allow us to find valuable information in a more manageable and friendly way in order to support network managers.

A. Contributions

The main contributions of the present paper are as follows:

- It presents the steps involved in our approach, which allows the analysis of YouTube Traffic Network traces. The steps include the conceptualization of flat flow of data into NTTTFs. It also includes the queries that were performed in order in order to extract key information about users' behavior about YouTube traffic at local network. The contribution of this paper falls into the category of querying for mining.
- It presents the schema of the NTTTFs, which allows us to perform semantic queries with sparl. NTTTFs help us identify key information and relate important information inside network traces.

B. Structure of the paper

The structure of the paper is organized as follows: In Section 2, we present the related work, while in Section 3, our approach is fully explained. In Section 4, we explain the experiments we performed with our approach. Section 5 explains the results we obtained and, finally, in Section 6, we conclude and mention future work on the subject.

II. RELATED WORK

1) Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications: Zink et al. [1] present a full investigation on how they found out that certain videos are far more popular than others by performing a statistical analysis on the networking traces. They took into account users local preferences instead of international preferences. Therefore, they demonstrated by means of simulations that by performing some actions in the network infrastructure such as the implementation of proxy catching, they could reduce network traffic significantly. We took the same traces Zink et al. used, in order to perform a semantic analysis.

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3

2) Execution Trace Exploration and Analysis Using Ontologies: Al Haider et al. [2] obtained execution traces from a software of a basic calculator and they performed a simple semantic analysis by using ontologies. Although they utilized a reasoner and sparql instructions, they did it with barely 100,000 traces. One of the contributions of the present paper is the implementation of a semantic analysis on a large amount of network traces.

3) Understanding Execution Traces Using Massive Sequence and Circular Bundle Views: Cornelissen et al. [3] proposed a a visual approach to understand the system at hand in order to maintain it. They do so by using dynamic information, e.g., execution traces. They developed a tool that shows the relationships between the method calls through curves in a circular view. They propose a visual approach to gather information from traces while we propose the execution of sparql queries in order to achieve the same goal.

4) Execution Trace Visualization in a 3D Space: Just as Cornelissen et al. [3] proposed a visual approach to understand the system to be analyzed by means of execution traces, Dugerdil et al. [4] proposed a visual approach to analyze execution traces based on 3D space.

5) Exploiting text mining techniques in the analysis of execution traces: Pirzadeh et al. [5] proposed an approach to analyze execution traces by taking into account the traces execution phases. They applied their approach to large traces of two different systems. We did not take into account traces execution phases because such analysis could not be applied to the network traces we choose due to their schema and the content as well.

6) Evaluating distributed real-time and embedded system test correctness using system execution traces: Hill et al. [6] proposed the evaluation of distributed real-time and embedded system test correctness by means of execution traces. Although the authors evaluated a distributed real-time and embedded system by means of execution traces, they did not do it by using semantic tools, which is an important difference from our work.

7) A survey of trace exploration tools and techniques: A survey of trace exploration tools and techniques was done by A. Hamou-Lhadj et al [7]. A review of the different tools and techniques for trace exploration has been made. None of them makes use of TTL files to perform a semantic analysis.

8) Relational Database Approach for Execution Trace Analysis: S. Alouneh et al. [8] proposed and approach to implement relational databases with execution traces. They visualize their proposed architecture and give advantages of their approach even though they did not implement it.

9) A Distributed Architecture for Dynamic Analyses on User-Profile Data: Antoniol et al. [9] proposed a model to represent dynamic information (traces). They collected and comprised traces from a distributed system and and left the door open to manipulate the resultant traces.

10) A Systematic Survey of Program Comprehension through Dynamic Analysis: Cornelissen et al. [10] made a systematic review on program comprehension through dynamic analysis (analysis of execution traces). This study was categorized into four facets: activity, target, method and evaluation.

11) SEAT-A usable trace analysis tool: Hamou-Lhadj et al. [11] proposed a software for a exploration analysis of execution traces. The software supports several features such as filtering techniques or working on several traces while we developed a tool that takes all the traces included in a folder and puts them together in a turtle format.

III. Our Approach

In the present section, an overview of our approach is being given. Figure 1 clarifies our approach.

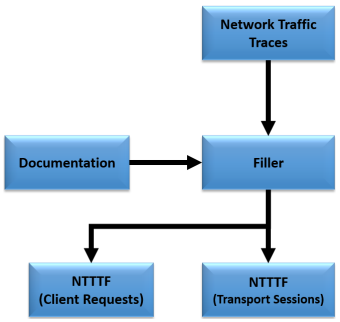


Fig. 1. Steps followed to obtain the NTTTFs.

First, documentation was obtained from UMass [12] traces repository. The downloaded documentation file has the schema of the network traces. Such a schema served us to code the algorithm of The Filler.

After that, both NTTTFs were populated by means of a software developed in .NET platform. This software is called The Filler. The Filler was developed in C# programming language. It generates two turtle files with .ttl extensions (NTTTF Client Requests and NTTTF Transport Session) and inserts into them the network traces also downloaded from the UMass [12] repository. Once we had the NTTTFs populated, semantic queries were performed in order to find key information.

Al Haider et al. [2] also proposed the use of semantic tools for traces. However, they used an ontology on execution traces they obtained from a software of a basic calculator. This leads to several differences; for example, the use of a reasoner to check consistency of the ontology. Also, they did its semantic analysis with barely 100,000 traces. One of the contributions of the present paper is the implementation of a semantic analysis on a large amount of network traces

Alouneh et al. [8] proposed an approach to implement relational databases with execution traces. They visualize their proposed architecture and give advantages of their approach even though the authors did not implement it. The main reason why we did not utilize relational databases is because we performed several tests before the experiments shown in the present paper. We had the serious restriction of using only 400,000 traces on the Jena Framework [13], so, we adjusted

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3

the resultant schema of the NTTF files and we were able to perform sparql queries on 1,100,000 traces in Jena. That is to say, we wanted to put to the test Jena Framework on the quantity of traces it can handle. Doing it with relational databases has the major drawback that every corporation that has a software for relational databases has his own implementation. We wanted to make our research as portable as possible by using turtle text files. It is not the scope of this paper to compare the efficiency of tools that make use of text files against tools that make use of relational databases. The scope of the present paper is to demonstrate that with network traces, keeping traces as text files and the use of semantic tools, we can find information for the decision taking. Even though efficiency is an important variable to take into account, it is the advantages of using ontologies the point we want to show.

A. Documentation

As a first step, we obtain the documentation from the UMass [12] repository. The document identifies two types of network traces:

- Trace files that contain information about client requests for YouTube video clips.
- Trace files that contain information about the transport session for video clips requested by clients from the UMass campus network.

Table 1 shows in the left column the fields found in the transport session trace files and an example in the right column. This is for clarification purposes.

TABLE I
SCHEMA AND EXAMPLE OF A TRANSPORT SESSION NETWORK TRACE

Field	Example
id	# 5
source_ip (anonymized)	64.15.112.107
sport	80
dest_ip (anonymized)	148.85.44.11
dport	2365
_pro	6
dir	1
start_time	0.464492
finishtime	74.901594
duration (in seconds)	74.437101
datapkts	3182
size_in_bytes	4644692
rate	499.180
flags	16

All parameters are self-explanatory except, for the flags parameter. This attribute is being used for control purposes, in this special case, network administrators. One thing to note about this attribute is that its value was always 16 in all the traces. So, this attribute, nor by itself, neither in conjunction with another attribute, gave us any information at all about the trace. Table 2 shows the fields found in the documentation for the client request trace files on the left column, while an example network trace is being shown on the right column.

TABLE II
SCHEMA AND EXAMPLE OF A CLIENT REQUEST NETWORK TRACE

Field	Example
timestamp	1189828805.208862
YouTube Server IP (anonymized)	63.22.65.73
sport	80
Client IP (anonymized)	140.8.48.66
Request	GETVIDEO
Video Id	lML9dik8QNw
Content Server IP	158.102.125.12

B. Network Traffic Traces

Network traffic trace files contain the data collected by monitoring the local network. There are 21 files that contain two types of network traffic traces, as was said before, YouTube Client Requests and YouTube Transport Sessions. Figure 2 shows a fragment of the content of a network traffic trace file. Each file has .dat extension and the information stored in every one of them is plain text.

```
1189828805.208862 63.22.65.73 140.8.48.66 GETVIDEO lML9dik8QNw 158.102.125.12
1189828810.212831 63.22.65.73 35.139.191.73 GETVIDEO bBXyB7niEc0&origin 105.136.66.5
1189828829.197218 63.22.65.77 205.181.191.47 GETVIDEO qVEcloLm4I4 63.22.67.110
1189828830.228538 63.22.65.73 102.15.239.161 GETVIDEO FKOn8OMN7F8 63.22.64.40
1189828832.795029 63.22.65.77 102.15.228.156 GETVIDEO WPcL8swzt8k 254.212.22.126
1189828833.031500 63.22.65.77 140.8.55.234 GETVIDEO OBSwS-QAv68&origin 105.136.66.5
1189828833.901196 63.22.65.73 140.8.49.142 GETVIDEO B361KLb_PlI&origin 105.136.66.5
1189828838.399229 63.22.65.73 140.8.48.66 GETVIDEO I0AKol7FIAA 254.212.22.75
1189828841.713910 63.22.65.77 102.15.255.4 GETVIDEO dciqhrfFXFw&origin 105.136.66.5
1189828841.792919 63.22.65.77 102.15.226.143 GETVIDEO 0bpgvFTF37c 254.212.22.187
1189828842.720476 63.22.65.73 140.8.54.40 GETVIDEO jXq3hO82cTY&origin 105.136.66.5
1189828846.842593 63.22.65.77 205.181.182.53 GETVIDEO XXG1-hlMX1o 254.212.18.215
1189828852.098143 63.22.65.73 102.15.234.3 GETVIDEO Oeq3XblvufE 254.212.17.132
1189828852.779726 63.22.65.73 205.181.167.225 GETVIDEO nFkW-bReI0c&origin 105.136.66.5
1189828853.623512 63.22.65.73 102.15.239.161 GETVIDEO QA4aJnYOChE 63.22.64.65
1189828859.296750 63.22.65.73 140.8.48.66 GETVIDEO Ksxhx9Nh6lQ 254.212.18.195
1189828860.421157 63.22.65.73 205.181.190.224 GETVIDEO aNrY9OcS-y8 254.212.19.76
1189828863.792110 63.22.65.77 102.15.237.62 GETVIDEO tl1IgCfZ7PA 254.212.23.146
1189828870.055699 63.22.65.73 140.8.54.219 GETVIDEO _s1YngtN3y4 63.22.64.32
1189828880.704996 63.22.65.73 205.181.190.224 GETVIDEO M5lnwJL5Mew 254.212.29.90
1189828885.808688 63.22.65.77 102.15.229.19 GETVIDEO CH1VqsQPTrE 254.212.19.76
1189828887.811597 63.22.65.73 205.181.188.157 GETVIDEO 82w3FkY5kPE&origin 105.136.66.5
1189828895.114468 63.22.65.73 35.139.180.61 GETVIDEO 53d7ikNox3c 254.212.29.239
1189828896.696499 63.22.65.77 35.139.31.8 GETVIDEO _-sNIWi2fLs 254.212.23.155
```

Fig. 2. Network Traces located in the .dat files.

As can be seen, each line represents a network traffic trace and the order of the fields is the same as shown in Section 3.

C. The Filler

The Filler automatically creates two empty files, ClientRequests.ttl and TransportSession.ttl, the NTTTFs, and automatically populates them with the information stored inside the network traffic traces files. It was because we were dealing with a great number of elements in the trace files, approximately 1,100,000 (one million one hundred thousand) traces, and doing so manually would have been a very hard and time consuming task. The Filler was created in order to cope with this task. Figure 3 shows the main screen of the software with all the controls visible and enabled in order for the reader to look at all the controls that are present in the Filler.

Figure 4 shows the software running with the first step already being taken. It is a software that consists of three simple steps:

- Step 1 - Load Network Traces (Client Requests). By pressing this button, a window will appear in which the

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3

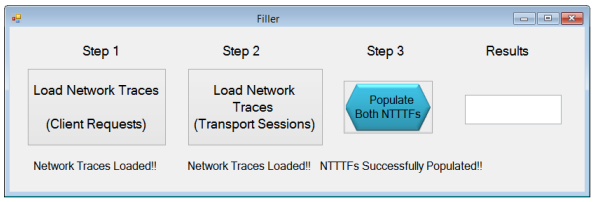


Fig. 3. Main screen of Filler. All controls enabled for clarification.

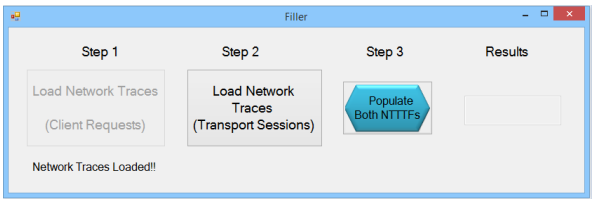


Fig. 4. The Filler running.

user can select the folder of the computer that contains the network traffic traces that correspond to the client requests. It is important to note that all the files that are inside the folder will be read in order to extract the traces from them. The latter was done in order to ensure minimum intervention from the user with the aim of making the software more easy to use. The folder needs to contain only the files that have traces within them. In our approach, the NTTTF Client Request was populated using two YouTube network traffic traces files. Their name start with the prefix flows.xxxx and have the .dat extension.

- Step 2 - Load Network Traces (Transport Sessions). By pressing this button a window will appear in which the user can select the folder of the computer that contains the network traffic traces that correspond to the transport session. The files that are inside the folder will be read in order to extract the traces from them. The folder needs to contain only the files that have traces within them. In our approach, the NTTTF Transport Session was populated using 19 YouTube network traffic traces files. Their name start with the prefix youtube.parsed.xxxxxx and have .dat extension.
- Step 3 - Populate Both NTTTFs. This button triggers the method that takes the selected folders and reads all the files included in them one by one. Each line of every file that is being read, is being parsed to the corresponding NTTTF with text marks that adjust the Turtle standard in order to obtain a Resource Description Format (RDF) [14] compliant file.

IV. Experimentation and Results

Once both NTTTF files (client request and transport session) were generated by The Filler, Jena Framework [13] was installed on the computer in order to perform the experiments. The Jena Framework has within the sparql [15] tool, which was used for performing semantic queries over the NTTTFs. Sparql is a RDF [14] query language that allows us to perform semantic queries, updating, copying and creation of data, etc. For our approach only the execution of semantic queries was needed.

The aim of the tests was to find valuable information that support the decision making of the local network managers. In order to perform the tests, the following steps were followed:

- Step 1 - Every sparql query was written down inside a text file with a .rq extension.
- Step 2 - The sparql application was executed with every query following the syntax:

sparql –query queryfile.rq –data NTTTFxx.ttl –time

where queryfile.rq was substituted by 1-Top100v1.rq, 2-TopDuration.rq and 3-TopSlowServers.rq

where NTTTFxx.ttl was substituted by NTTTFCR.ttl (Client Request) and NTTTFTS.ttl (Transport Session).

where the -time clause measures the time the query needed to execute. Finally, we interpret the results and demonstrate the valuable information that can be found by means of semantic tools.

A. First Test

The aim of the first test was to locate the top 30 most viewed videos. The purpose of this query is to provide local network managers with a list of the videos that can be stored in the cache local server in order to reduce the traffic outside of the local network, increasing the bandwidth left for other users. The first test made use of the NTTTF Client Request. Figure 5 shows the sparql query:

```
prefix
ntttfcr: <http://ntttfcr.com/ns/NTTTFCR#>

SELECT (COUNT(?ntttfcr) as ?Record_Quantity)
?VideoId

WHERE
{
  ?ntttfcr ntttfcr:VideoId ?VideoId.
}

GROUP BY ?VideoId
ORDER BY DESC(?Record_Quantity)
LIMIT 30
```

Fig. 5. First test sparql query.

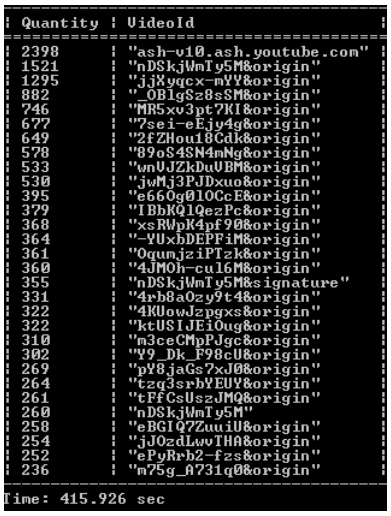
Here is the shell command that was necessary in order to execute the query:

sparql –query 1-Top100v1.rq –data NTTTFCR.ttl –time

Figure 6 shows the result thrown by the sparql tool.

The left column in Figure 6 shows the number of times a certain YouTube was visited. In the right column, there are the video identifiers. The VideoId that is on top is the video that has been accessed the most number of times (2398 times). This means that, if we backup this video in a local cache server, 2398 accesses outside of the local network will be saved. Now, by adding the top 100 hits of the same query, we get 28793 accesses. If we follow the same procedure, we would be saving

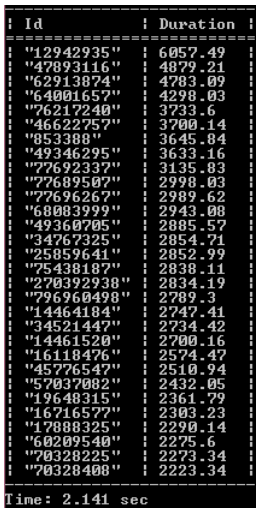
Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3



Quantity	VideoId
2398	"ash-v10.ash.youtube.com"
1521	"nDSkjWmIy5M&origin"
1295	"jjXyqcx-mYY&origin"
882	"_OBlgSz8sSM&origin"
746	"MR5xv3pt7KI&origin"
677	"7sei-eEjy4g&origin"
649	"2fZHou18Cdk&origin"
578	"89oS4SN4mNg&origin"
533	"wnUJZkDuVBM&origin"
530	"jwMj3PJDxuo&origin"
395	"e66Qg0lOCcE&origin"
379	"IBbKQlQezPc&origin"
368	"xsRWpK4pf90&origin"
364	"-YUxbDEPFiM&origin"
361	"OqumjziPIzk&origin"
360	"4JMOh-cul6M&origin"
355	"nDSkjWmIy5M&signature"
331	"4rb8aOzy9t4&origin"
322	"4KUowJzpgxs&origin"
322	"ktUSIJEiOug&origin"
310	"m3ceCMpPJgc&origin"
302	"Y9_Dk_P98cU&origin"
269	"pY8jaGs7xJ0&origin"
264	"tzq3srbYEUY&origin"
261	"tFfCsUszJMQ&origin"
260	"nDSkjWmIy5M"
258	"eBGIQ7ZuuiU&origin"
254	"jJOzdLwvTHA&origin"
252	"ePyRrb2-fzs&origin"
236	"m75g_A73iq0&origin"

Time: 415.926 sec

Fig. 6. Results of the first test.



Id	Duration
"12942935"	6057.49
"47893116"	4879.21
"62913874"	4783.09
"64001657"	4298.03
"76217240"	3733.6
"46622757"	3700.14
"853388"	3645.84
"49346295"	3633.16
"77692337"	3135.83
"77689502"	2998.03
"77696267"	2989.62
"68083999"	2943.08
"49360705"	2885.57
"34767325"	2854.71
"25859641"	2852.99
"75438187"	2838.11
"270392938"	2834.19
"796960498"	2789.3
"14464184"	2747.41
"34521447"	2734.42
"14461520"	2700.16
"16118476"	2574.47
"45776547"	2510.94
"57037082"	2432.05
"19648315"	2361.79
"16716577"	2303.23
"17888325"	2290.14
"60209540"	2275.6
"70328225"	2273.34
"70328408"	2223.34

Time: 2.141 sec

Fig. 8. Results of the second test.

28693 accesses outside of the local network. We think this would represent a huge saving in bandwidth if measures for saving would be applied. Another piece of information that can be noted from Figure 6 is the time it took for the sparql tool to perform the query, namely 415.926 seconds. This represents a small amount of time taking into account the quantity of traces analyzed.

B. Second Test

The second test we performed was to locate the id of the longest transport sessions. We suggest that a local backup of the longest transport sessions will increase overall bandwidth available for the users. This test made use of the NTTTF Transport Session. Figure 7 shows the sparql query:

```
prefix
ntttfts: <http://ntttfts.com/ns/NTTTFTS#>

SELECT ?Id ?Duration ?Size ?Rate
(((?Size/?Duration)/1024) AS ?Velocidad)
WHERE
{
  ?ntttfts ntttfts:Id ?Id.
  ?ntttfts ntttfts:Duration ?Duration.
  ?ntttfts ntttfts:Size ?Size.
  ?ntttfts ntttfts:Rate ?Rate.
}

ORDER BY DESC(?Percentage_Accuracy)
LIMIT 30
```

Fig. 7. Second test sparql query.

Here is the ms-dos command that was necessary in order to execute the query:

sparql –query 2-TopDuration.rq –data NTTTFTS.ttl

Figure 8 shows the result thrown by the sparql tool.

Figure 8 shows on the left column the identifiers of the transport sessions with the highest duration in seconds (right column). As can be seen, we have very long sessions (6057.49 seconds the longest of them). Presumably, the longer the transport sessions, the longer videos last. We are sure that by storing not only most viewed videos but also the longest videos in a local cache server will have a direct impact in the overall available bandwidth for the rest of the users of local network. Now, by summing the top 100 hits of the same query we get 202,608.46 seconds in total. This means that by backing up these videos, we will be unloading local network from this amount of time. Another piece of information that can be noted from Figure 8 is the time it took for the sparql tool to perform the query, namely 2.141 seconds.

C. Third Test

The third test we performed was to locate the slowest YouTube servers. We suggest that a local backup of the slowest servers' videos is necessary in order to increase overall bandwidth available for the users. This test made use of the NTTTF Transport Session. Here is the sparql query:

```
prefix
ntttfcr: <http://ntttfcr.com/ns/NTTTFCR#>

SELECT (COUNT(?ntttfcr) as ?Record_Quantity)
?VideoId

WHERE
{
  ?ntttfcr ntttfcr:VideoId ?VideoId.
}

GROUP BY ?VideoId
ORDER BY DESC(?Record_Quantity)
LIMIT 30
```

Fig. 9. Third test sparql query.

Here is the ms-dos command that was necessary in order to execute the query:

sparql –query 3-TopSlowServers.rq –data NTTTFTS.ttl

Figure 10 shows the result thrown by the sparql tool.

Figure 10 shows on the left column the IP addresses with the lowest rates (right column). This means that these are the slowest YouTube servers. The identification of slow servers

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3

SourceIP	Rate
"254.212.22.13"	1.623
"254.212.22.177"	1.624
"254.212.31.99"	1.764
"63.22.65.242"	2.284
"63.22.67.81"	2.375
"63.22.65.238"	2.463
"254.212.31.125"	2.652
"254.212.19.193"	2.956
"254.212.23.152"	3.075
"63.22.67.182"	3.469
"63.22.65.202"	3.527
"254.212.22.188"	3.528
"254.212.25.141"	3.542
"254.212.30.233"	3.623
"254.212.23.241"	3.715
"254.212.30.108"	3.749
"254.212.31.26"	4.525
"254.212.22.216"	5.257
"254.212.30.68"	5.598
"254.212.25.230"	5.840
"254.212.23.252"	6.144
"254.212.31.60"	6.168
"63.22.67.48"	6.299
"254.212.31.100"	6.380
"254.212.25.174"	7.167
"254.212.25.197"	7.681
"254.212.19.245"	8.158
"254.212.22.105"	8.245
"254.212.31.78"	8.353
"254.212.25.226"	8.545

Time: 2.156 sec

Fig. 10. Results of the second test.

allows for actions to be taken to improve the situation, such as the search for alternate servers. These are actions taken by some network devices, such as routers, in order to enhance the overall performance of the network. Sometimes there is no solution, no alternate servers for example, to this problem. We are sure that by identifying slow servers and reducing access to them will not only diminish the network usage but also reduce the work load of network devices that perform actions to bypass them. Another piece of information that can be noted from Figure 10 is the time it took for the sparql tool to perform the query, namely 2.156 seconds, which is very little time for the information we gather.

D. Fourth Test

This test represents the last bastion we need to ensure we can have a direct impact on the enhancement of the local network by using semantic web tools. The fourth test we performed was to search for atypical values on the duration on the transpot sessions traces. An example of an atypical value may be a session that lasted too long for a small file with a good rate. This test made use of the NTTTF Transport Session. Figure 11 shows the sparql query.

```
prefix
ntttfcr: <http://ntttfcr.com/ns/NTTTFCR#>

SELECT (COUNT(?ntttfcr) as ?Record_Quantity)
?VideoId

WHERE
{
   ?ntttfcr ntttfcr:VideoId ?VideoId.
}

GROUP BY ?VideoId
ORDER BY DESC(?Record_Quantity)
LIMIT 30
```

Fig. 11. Fourth test sparql query.

Here is the ms-dos command that was necessary in order to execute the query:

sparql –query 4-Anomalies.rq –data NTTTFTS.ttl

Figures 12 and 13 show the result given by the sparql tool.

Id	Adj_Rate	Expec_Dur	Perc_Acc
"179052939"	203.000	85.596059	100.0125
"41499985"	433.625	73.464398	100.0102
"464011919"	384.375	64.260162	100.0100
"10222582"	296.875	107.30442	100.0097
"7963810"	657.125	74.920296	100.0081
"19529938"	771.000	174.21530	100.0064
"71014214"	1019.750	143.17234	100.0058
"10028883"	895.875	84.047439	100.0045
"208667451"	1286.750	11.346415	100.0045
"19613883"	797.500	82.382445	100.0044
"176524904"	1390.750	86.416681	100.0043
"256516134"	1314.375	18.728292	100.0042
"77674366"	1422.000	192.45569	100.0039
"286471345"	1068.125	75.916208	100.0038
"840335500"	1315.000	11.863117	100.0035
"14364216"	1609.375	8.9972815	100.0034
"804698197"	1554.000	8.3861003	100.0031
"205809556"	1493.375	100.69808	100.0030
"197505870"	1885.875	6.9675879	100.0028
"193404776"	1626.250	193.21506	100.0026
"56919421"	2565.625	298.18855	100.0025
"835212687"	1538.625	92.095214	100.0025
"14232750"	2015.125	21.735624	100.0024
"478734005"	3073.250	111.06157	100.0023
"703803459"	1334.375	8.6812177	100.0022
"19648315"	1175.125	2361.8423	100.0022
"112693962"	768.000	76.041666	100.0021
"726062018"	565.625	74.24	100.0021
"68228256"	2641.000	9.3207118	100.0020
"62913874"	2575.875	4783.1870	100.0020

Fig. 12. Results of the fourth test with DESC clause.

Id	Adj_Rate	Expec_Dur	Perc_Acc
"50651712"	285.500	77.408056	99.97889
"318835566"	202.875	93.555144	99.98006
"801594294"	220.500	79.455782	99.98072
"179079780"	442.750	68.679841	99.98841
"233780649"	441.000	119.18367	99.99049
"802128800"	331.500	70.467571	99.99102
"416516478"	440.875	46.362347	99.99147
"283244266"	464.375	106.01776	99.99317
"176318105"	730.000	75.375342	99.99501
"41247664"	1408.125	81.237106	99.99557
"114868111"	1571.125	93.856313	99.99628
"112173507"	1744.500	77.935224	99.99682
"14461520"	1913.625	2700.0796	99.99702
"33861210"	2000.500	25.823544	99.99707
"309949231"	1720.500	11.880267	99.99720
"500814662"	1236.375	133.43847	99.99736
"31568239"	1985.250	161.05780	99.99739
"64841211"	2361.625	1218.5084	99.99741
"230706109"	1425.500	114.90705	99.99743
"10959683"	1359.500	114.74806	99.99744
"9922088"	2114.875	2059.3084	99.99749
"44171893"	468.625	86.516937	99.99750
"797628313"	2284.750	77.319619	99.99756
"177997837"	2486.750	68.127475	99.99761
"487659026"	1928.750	59.800388	99.99764
"737119301"	1969.875	11.858620	99.99764
"56873678"	787.375	31.522463	99.99766
"191225182"	2771.625	124.86249	99.99799
"164938438"	1030.625	120.82765	99.99805
"56330775"	2287.875	461.38010	99.99807

Fig. 13. Results of the fourth test with ASC clause.

Figures 12 and 13 show, from left to right, the Id of the transport session, adjusted rate, expected duration, percentage of accuracy. The main difference is that Figure 12 shows the top most different transport sessions Ids and Figure 13 shows the lowest different transport sessions. This query works as follows: first, we needed to adjust the rate to obtain 100% accuracy. That is the reason why we multiply the rate by 125. Later on, we divided the size and adjusted the rate. By doing this we obtained the expected duration of a transport session. Finally, we compared the duration of the transport session with the expected duration. This comparison gave us a percentage value of accuracy. Every value that has a bias of more than 1% would be considered as atypical. As it can be seen from both figures, there is no atypical value. This means that all values from this trace are perfectly explainable.

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3

V. Conclusion and Future Work

Semantic web tools, such as sparql from Jena Framework, allow us to perform semantic queries in order to extract valuable information from raw data. As shown in Section 4, the information gathered by means of semantic queries could help people (local network administrators in this case) to better understand the system at hand in order to take precise decision over its maintenance. It is important to note that the total amount of time for the three queries to execute in a laptop with standard specifications (Core i5, 8Gb RAM, 1TB HDD) was of 420.223 seconds (7 minutes). It is a very small amount of time taking into account the network traces quantity, approximately 1100000 (1 million one hundred thousands). That is to say, approximately 10 minutes are necessary in order to follow our approach. It is our belief that more experimentation must be done in order to take advantage of more semantic tools that are at our disposal, for example, ontologies. Further investigation will be done with network traces but next time, we will take advantage of the characteristics of ontologies in order to perform more sophisticated and complex processes. Ontologies are formal representations of information and have characteristics such as datatype properties, object properties, inference of information, etc., that makes them attractive to be applied to traces. One way of applying ontologies to traces is by defining a model of an ontology that matches the schema of the trace at hand. After that, we need to look for the technology that best suits the expecting results, a software for example. Next steps are uncertain and will depend on what authors will seek to solve with the ontology. This will allow us to gather more accurate and concluding information for the decision making.

VI. Acknowledgments

Oscar Alberto Santana Alvarez will like to thank the Consejo Nacional de Ciencia y Tecnología (CONACyT) for their support for his PhD program. Also, we will like to thank to Michael Zink, Kyoungwon Suh and Jim Kurose for giving us access to the network traffic traces they gathered. These files were the principal input for the present paper.

References

[1] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network - measurements, models, and implications," Comput. Netw., vol. 53, no. 4, pp. 501–514, Mar. 2009, [retrieved: 05, 2014]. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2008.09.022

[2] N. Al Haider, B. Gaudin, and J. Murphy, "Execution trace exploration and analysis using ontologies," in Proceedings of the Second International Conference on Runtime Verification, ser. RV'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 412–426, [retrieved: 05, 2014]. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29860-8_33

[3] B. Cornelissen et al., "Understanding execution traces using massive sequence and circular bundle views," in Proceedings of the 15th IEEE International Conference on Program Comprehension, ser. ICPC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 49–58, [retrieved: 05, 2014]. [Online]. Available: http://dx.doi.org/10.1109/ICPC.2007.39

[4] P. Dugerdil and S. Alam, "Execution trace visualization in a 3d space," in Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on, April 2008, pp. 38–43.

[5] H. Pirzadeh, A. Hamou-Lhadj, and M. Shah, "Exploiting text mining techniques in the analysis of execution traces." in ICSM. IEEE, 2011, pp. 223–232, [retrieved: 05, 2014]. [Online]. Available: http://dblp.uni-trier.de/db/conf/icsm/icsm2011.html#PirzadehHS11

[6] J. H. Hill, P. Varshneya, and D. C. Schmidt, "Evaluating distributed real-time and embedded system test correctness using system execution traces." Central Europ. J. Computer Science, vol. 1, no. 2, pp. 167–184, 2011, [retrieved: 05, 2014]. [Online]. Available: http://dblp.uni-trier.de/db/journals/cejcs/cejcs1.html#HillVS11

[7] A. Hamou-Lhadj and T. C. Lethbridge, "A survey of trace exploration tools and techniques," in Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research, ser. CASCON '04. IBM Press, 2004, pp. 42–55, [retrieved: 05, 2014]. [Online]. Available: http://dl.acm.org/citation.cfm?id=1034914.1034918

[8] S. Alouneh, S. Abed, B. Mohd, and A. Al-Khasawneh, "Relational database approach for execution trace analysis," in Computer, Information and Telecommunication Systems (CITS), 2012 International Conference on, May 2012, pp. 1–4.

[9] G. Antoniol and M. D. Penta, "A distributed architecture for dynamic analyses on user-profile data." in CSMR. IEEE Computer Society, 2004, pp. 319–328, [retrieved: 05, 2014]. [Online]. Available: http://dblp.uni-trier.de/db/conf/csmr/csmr2004.html#AntoniolP04

[10] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis." IEEE Trans. Software Eng., vol. 35, no. 5, pp. 684–702, 2009, [retrieved: 05, 2014]. [Online]. Available: http://dblp.uni-trier.de/db/journals/tse/tse35.html#CornelissenZDMK09

[11] A. Hamou-Lhadj, T. Lethbridge, and L. Fu, "Seat: a usable trace analysis tool," in Program Comprehension, 2005. IWPC 2005. Proceedings. 13th International Workshop on, May 2005, pp. 157–160.

[12] T. Weibel. Network - umass trace repository. [accessed: 05, 2014]. [Online]. Available: http://traces.cs.umass.edu/index.php/Network/Network

[13] Apache.org. Jena framework. [accessed: 05, 2014]. [Online]. Available: http://www.apache.org/

[14] W3.org. Rdf - semantic web standards. [accessed: 05, 2014]. [Online]. Available: http://www.w3.org/RDF/

[15] W3.org. Sparql query language for rdf. [accessed: 05, 2014]. [Online]. Available: http://www.w3.org/RDF/

Copyright (c) IARIA, 2014. ISBN: 978-1-61208-364-3