Finding Typical Internet User Behaviors

Péter Megyesi, Sándor Molnár

Finding typical Internet user behaviors

Péter Megyesi, Sándor Molnár

High-Speed Network Laboratory
Department of Telecommunication and Media Informatics
Budapest University of Technology and Economics
{megyesi, molnar}@tmit.bme.hu

Abstract. In various network tests we often need to use different trace files in order to get the most comprehensive result. This procedure requires multiple input files which were generated in different ways. In this paper we suggest a method for analyzing a traffic measurement and extracting the most typical user behaviors. We introduce the Traffic Descriptive Strings (TDS) which is a projection of measurement data. We present an algorithm which is able to score the similarities between two TDSs.

Keywords: Internet, user behavior, traffic analysis

1 Introduction

Testing various network devices often leads to a point where captured trace files have to be used. The choice of the right input raises the question of using traces which are available on the Internet for everyone or generate them manually. The first method is usually useful at final stage of testing for making the results comparable. The second technique is applied at early stages during the tuning of the different parameters as the usage of manually generated trace files have the benefit that we know what the input contains, therefore we know what to expect as a result. However, in order to create the most general and realistic input files we have to know the usual way of how users use the Internet.

In this paper we present a method for extracting typical user behaviors from Internet traffic measurements. We introduce the Traffic Descriptive Strings (TDS) which are a projection of traffic measurements and can be used for representing a user's traffic stream. We have implemented a scoring scheme which is able to evaluate the similarities between two traffic streams represented by TDSs. This scoring scheme is used by an algorithm which is able to find the most similar typical user behavior for any given TDS. Our method is novel since according to our knowledge the current literature lacks methods representing user traffic by such string descriptors.

In the next section we discuss the general way of creating Traffic Description Strings from a traffic measurement report. Section 3 presents the algorithm which is able to extract typical patterns from a measurement results converted into TDSs. In Section 4 we present the TDS scoring scheme and we give some examples for its

result using an artificially created database. Finally, we summarize our work in Section 5.

2 Traffic Descriptive Strings

During the creation of the Traffic Descriptive Strings we have to run a traffic classification algorithm on the input raw measurement. The efficiency of the various traffic classification algorithms covers a wide range, but Deep Packet Inspection (DPI) solutions tend to generate the best results. DPI is a form of classification that also examines the data part of the packets not just the headers. Although the source of the most sophisticated industrial DPI tools are closed to the public, there are many open source DPI projects [3]. One of the most popular open sourced DPI is OpenDPI. OpenDPI is the open source version of Ipoque's DPI engine [4]. Ipoque is a provider of DPI solutions for internet traffic management and analysis. OpenDPI supports most of today's popular network protocols [5].

Before the processing of the report generated by the traffic classification tool, the definition of the traffic classes has to be made. This process is highly dependent on the classification tool we use since the application might not be able to distinguish certain traffic classes we have defined. In the followings we will use the traffic classes found in Ipoque's Internet study [2] as these classes are also distinguishable by OpenDPI. Thus the defined traffic classes can be found in Table 1 where we have assigned a character to each of them.

Table 1. Defined traffic classes.

Traffic Class	Assigned Character	Description
P2P	A	Peer-to-peer file sharing
Web	B	Web browsing incl. file download
Streaming	C	Audio and video streaming
VoIP	D	Voice over IP
IM	E	Instant messaging
Tunnel	F	Encrypted tunneling protocols
Standard	G	Legacy Internet protocols
Gaming	H	Multiplayer and network games

After the definition of the traffic classes the creation of the Traffic Descriptive Strings is implemented as follows. Firstly, a time resolution has to be chosen. This will serve as a time interval that the algorithm parcels the measurement. Each time slot will be handled independently, thus the choice of the resolution has to be adjusted for the required results. A too short time interval can result in an output where typical patterns are not extractable. On the other hand, using a too long time unit could hide the differences between distant inputs. In our early tests we chose this time resolution to one minute. However, this decision will be further investigated based on the range of human user reactions with controlled applications.

As a next step, we disassociate the input to individual users in case of an aggregated measurement. Then the algorithm examines for every user that which of the defined traffic classes were used in every time slot. If the transmitted data of one traffic class is larger than a given limit (e.g. 100 Kbytes) its character is written into the Traffic Descriptive String of the user. We separate the time slots with the semicolon character. Thus a suitable input format for the typical user behavior extractor algorithm is given in Figure 1.

```
1223378304 AB;AB;AB;AB;AB;AB;AB;AB;AB;AB;AB;AB;
1223378343 DH;DH;DH;DH;DH;DH;DH;DH;DH;DH;DH;DH;
1223378892 C;C;C;C;C;C;C;C;C;C;C;C;
```

Fig. 1. Measurement results converted into TDS format.

The example in Figure 1 describes three users for 12 time units (12 minutes in our example); the first user was running a peer-to-peer file sharing client while browsing the Internet; the second one was using a VoIP and a gaming application simultaneously; and the third user was streaming online media content. We also add the UNIX timestamp of the beginning of the given user's activity for possible future processing.

3 Extracting typical user behaviors

During the extract of typical behaviors we tend to find frequently occurring shorter term periods in the input. In this procedure the algorithm examines every fixed time length patterns between a lower and a higher time limit. The reason for the lower limit is that too short time periods wouldn't generate stabile traffic. For example, if a torrent client runs for only one minute the number of the peer-to-peer connections wouldn't reach its maximum value. On the other hand, the longer a pattern is the lesser it would occur in the traffic stream. In our test we have defined the lower limit in 4 minutes and the higher limit in 10 minutes.

As a first appealing solution we have inspected existing motif finding methods used in bioinformatics problems. In bioinformatics algorithms tend to find similarities in protein chains. These algorithms usually use a simple string input which contains the markings of the four proteins: A for Adenine, T for Thymine, G for Guanine and C for Cytosine. Thus the principal of the task is the same: finding typical motifs in a string array. The idea of using bioinformatics algorithms for motif finding is presented in [6]. In this paper the authors present a framework using Glam2 for signature generation. Glam2 is a software package for finding motifs in sequences, typically amino-acid or nucleotide sequences [7]. Glam2Scan is a part of the Glam2 software package which can find matches in a sequence database to a motif discovered by Glam2 [8]. Glam2Scan gives a score for each match indicating how well it fits the given motif so it also could be the base an approximately match algorithm.

We have examined the possibility of applying the same architecture presented in [6] with a different preprocessing method. However, after a long investigation process we have found that this technique is not appropriate for our work. The main reason behind this is that bioinformatics algorithm use alphabet where the role characters are the same. In a Traffic Descriptive String the time delimiter semicolon character has completely different meaning than the others. Moreover, in our case we would like to have a flexible method for defining suitable replacements for certain traffic classes (e.g. assigning different characters for applications which generates the same type of traffic, but we would like to distinguish them in the result, such as web browsers) and also complete exclusions (e.g. give low score when replacing web to P2P).

Thus we implemented a unique algorithm for the Traffic Emulation Tool which can extract typical user behaviors from the input files. Firstly, it filters out the long idle periods (multiple consecutive semicolon characters) from the input thus the results won't contain patterns in which the idle period is longer than the active period. Secondly, for symmetric TDS patterns (where the same characters occur in every time unit) the algorithm recalculates the real number occurrences by dividing the total run length with the time length of the pattern.

In the algorithm two limits have to be declared: a hard limit and soft limit. Firstly, the tool counts the occurrences for every occurring TDS substring which has the given time length. Then the algorithm discards patterns below the soft limit. After that, following iteration is repeated. The algorithm calculates the most similar pattern for the substring which occurrence is equal to the soft limit using the scoring mechanism presented in Section 4 and increases the result's occurrence by the soft limit. Then the soft limit is increased by one. This procedure is repeated until the soft limit reaches the hard limit. After that, the remaining TDS substrings occurring more than the hard limit are added into the typical behavior pool.

The choice of the limits is dependent on the result we would like to achieve. For example, setting the two limits to the same value will avoid the usage of the scoring mechanism and result in only the TDSs which occurrence is greater than then the given hard limit. On the other hand, setting the soft limit to 1 will give the most detailed result but it can significantly increase the run time of the algorithm.

4 The TDS scoring algorithm

When we were designing the TDS scoring algorithm the following rules were laid down:

1. If we compare an A string to a B string, the returned score must be less than or equal to score the algorithm returns comparing the A string with itself.
2. The equality must only stand if the same characters with the same amount are in both A and B.
3. The algorithm must inspect the time length of the TDSs and give lesser score if it differs.
4. We must have a way of setting unique values for which traffic types are suitable substitutions for each other and which are completely excluded.

Taking these considerations into account, we have defined a scoring matrix labeling its rows and columns with the defined traffic classes' characters. We also add the "X" character which will refer to no action. If we substitute an "A" character from the first string to a "B" character in the second string the score under the "A" row and "B" column will be added to the total score. Firstly, the algorithm concatenates "X" characters to the shorter string until both of them contain the same amount of characters. After this, a dynamic programming algorithm finds the best substitution solution for the characters calculating the maximal possible score [1], [9]. As the last step the algorithm modifies the given score with a divider if the time lengths of two strings differ.

In order to get the most ideal values of the scoring matrix and the length modifier we created a test database of Traffic Descriptive Strings. In contrast with the typical user behaviors this artificial database contains only TDSs in which the actions are the same in every time slot. We integrated every variation of minimum four maximum ten time unit long scenarios which contains maximum 4 classes of traffic simultaneously.

An example for the scoring matrix can be found in Table 2. The ideas behind the definition of the individual values were the followings. The values in the "X" row represent the scores given for extra traffic. These numbers have to be negative otherwise the first rule wouldn't apply. Here we give lesser values for classes with higher bandwidth as adding them into the traffic stream would suppress the others. Similar considerations were made for the values in the "X" column. These numbers represents the absence of a class without substitution. Here we also give lesser values for the classes with higher bandwidth as their absence from the traffic stream would make the dominant flow disappear. The other values in the main diagonal are set to 4 thus if a character is substituted for itself the algorithm gives 4 points. The remaining values in the matrix are close to zero. The main reason for this idea is that these traffic classes are fairly different from each other thus none is a suitable replacement for another. However, a slight preference sequence can be given by small values.

Table 2. Example for the scoring matrix.

	X	A	B	C	D	E	F	G	H
X	0	-3	-2	-3	-2	-1	-2	-1	-2
A	-3	4	0.1	0.2	0.1	0	0.1	0	0.1
B	-2	0	4	0.1	0.1	0.1	0.1	0.1	0.1
C	-3	0.1	0	4	0.2	0	0	0	0.2
D	-2	0	0.1	0.2	4	0	0	0	0.2
E	-1	0	0.1	0	0.1	4	0	0.2	0.1
F	-2	0	0.1	0	0.1	0	4	0	0.1
G	-1	0	0.1	0	0.1	0.2	0.1	4	0.1
H	-2	0	0.1	0.2	0.2	0.1	0.1	0.1	4

Two examples are given for the result in Table 3 and Table 4. In these cases we allow one minute time different between the compared strings and divide the final score the by 1.2 if the time difference consists. In both table the 0th row shows the score with the input itself. In case of Table 3 the input was a six-minute-long VoIP activity. Since this is a symmetrical TDS this string is in the test database thus the most similar pattern is itself. As the results show, the second most similar TDS is the seven-minute-long VoIP activity. After that the given scoring scheme prefers the addition of low bandwidth classes, the Instant messaging (E) and the Standard protocols (G).

In case of Table 4 the input was again a six-minute-long activity, but here the first three minutes contained VoIP while last three Streaming. In the results we can see that the six-minute-long only VoIP and only Streaming got the same amount of points being equally similar to the input. The next two results are the simultaneous VoIP and Streaming, the five-minute-long in the 3rd place while the six-minute-long in the 4th place. The reason for this succession is that even thought the five-minute-long TDS is shorter in time, but the sum of the generated traffic would be closer to the input.

Table 3. Scoring algorithm, example 1.

#	score	ratio	time length	TDS
0	24	1	6	D;D;D;D;D;D;
1	24	1	6	D;D;D;D;D;D;
2	18.33	0.76	7	D;D;D;D;D;D;D;
3	18	0.75	6	DG;DG;DG;DG;DG;DG;
4	18	0.75	6	DE;DE;DE;DE;DE;DE;
5	15	0.62	5	D;D;D;D;D;

Table 4. Scoring algorithm, example 2.

#	score	ratio	time length	TDS
0	24	1	6	D;D;D;C;C;C;
1	12.6	0.52	6	D;D;D;D;D;D;
2	12.6	0.52	6	C;C;C;C;C;C;
3	11.66	0.48	5	CD;CD;CD;CD;CD;
4	9	0.37	6	CD;CD;CD;CD;CD;CD;
5	8.83	0.36	7	D;D;D;D;D;D;D;

These examples describe one possible way of choosing the scoring values. If we would like to prefer e.g. the longer scenarios against the additions, we could easily adjust the values for our purpose. The adjustment of the scoring values from different point of views can be found in [1].

5 Conclusion

In this paper we introduced the Traffic Description Strings (TDS) which is a projection of traffic measurement result. We have presented an algorithm which is able to extract typical user behaviors from real traffic measurement. This method uses a unique scoring scheme which is able determine similarities between two arbitrary given TDSs.

The knowledge of the typical way of how users use the Internet could benefit us in several ways. For example, we can generate up to date trace for various testing purposes since we know in which combinations the different type of applications have to be launched.

Acknowledgments. The authors would like to thank Dr. Géza Szabó, TrafficLab, Ericsson Research Hungary for his contribution and constant support for this project. This research was supported by OTKA-KTIA grant CNK77082.

References

1. Megyesi P.: Matching Algorithm for Network Traffic Descriptive Strings. In: Scientific Students' Associations Conference, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Budapest (2011) (Available online: http://tdk.aut.bme.hu/Conf/TDK2011/halozattervezes/Halozati-Forgalmat-Leiro)
2. Schulz, H., Mochalski, K.: Internet Study 2008/2009. Ipoque, Leipzig, Germany (2009)
3. Open source Deep Packet Inspection projects, https://www.dpacket.org/group-posts/open-source-software-general-discussion/open-source-software-related-deep-packet-inspect
4. OpenDPI, http://www.opendpi.org/
5. OpenDPI manual, http://opendpi.googlecode.com/files/OpenDPI-Manual.pdf
6. Szabó, G., Turányi, Z., Toka, L., Molnár, S., Santos, A.: Automatic Protocol Signature Generation Framework for Deep Packet Inspection. In: VALUETOOLS 2011, ENS, Cachan, France (2011)
7. Glam2 manual, http://meme.sdsc.edu/meme/doc/glam2_man.html
8. Glam2Scan manual, http://meme.sdsc.edu/meme/doc/glam2scan_man.html
9. Dynamic Programming, http://www.cs.berkeley.edu/~vazirani/algorithms/chap6.pdf