Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2012, Vol. 4, No. 2

Inferring User-Perceived Performance of Network by Monitoring TCP Interruptions

Junaid Shaikh, Markus Fiedler, Tahir Nawaz Minhas, Patrik Arlos

School of Computing, Blekinge Institute of Technology

Karlskrona 371 79, Sweden

Tel: +46 455 38 50 00

E-mail: {junaid.junaid, markus.fiedler,tahir.nawaz.minhas,patrik.arlos}@bth.se

Denis Collange

R&D Orange Labs

Sophia Antipolis, France

E-mail: denis.collange@orange.com

Received: February 7, 2012 Accepted: May 19, 2012 Published: June 19, 2012

DOI: 10.5296/npa.v4i2.1357 URL: http://dx.doi.org/10.5296/ npa.v4i2.1357

Abstract

The fluctuating performance of wireless and mobile networks has triggered the need for smart algorithms to assess the user perception, resulting from the quality of network services. While efforts have been done to model the user experience resulting from the network performance, there is still the need for practical methods to assess the user-perceived performance, in the real environment. In this work, we present a set of criteria to observe the user behavior on the Web, passively from the network-level. The criteria are based on the monitoring of TCP control flags and HTTP requests. Thus, information about user actions performed in the web browser can be inferred by monitoring the TCP termination flags and by keeping track of the HTTP requests. Along the way, we also present some anomalies observed in the TCP connection termination process, which may result in performance degradation of Web transfers.

Keywords: TCP, Web, Quality of Experience, User-perceived performance

1. Introduction

In the recent years, the Internet has witnessed the mushrooming of networks and applications. Among all the applications, the Web application is the most dominant one. The popularity of Web is further fueled by the migration of video on the Web. According to study [1], Web traffic accounts for the major part of the traffic volume on the Internet.

With the increasing usage of the Internet, the expectations of users are also evolving. While the computational power of the devices, intelligence of applications and speed of networks are increasing with time, following Moore's law, expectations of users are following "the More's law" [2]: Users want more in less time. They are becoming increasingly strict and intolerant about the quality of network and application services. This is because the users now have to rely on the Internet for their everyday tasks. Due to these growing expectations, the margin of error is getting smaller and the network protocols and algorithms need to perform smartly and accurately.

For an Internet Service Provider (ISP), it is extremely important to monitor and keep track of the service quality as perceived by the user. There are several competitors in the market and a user may easily switch to another service provider as a result of dissatisfaction, taking several other users with her. Hence, there is need of a mechanism in order to learn about the user experience over time, in order to provide better services. However, being certain about user experience is a complex task for the following reasons.

The quality perceived by the users is mainly affected by the several network-dependent and application-specific factors. However, there are many other factors that may influence the quality as perceived by the users such as, the prior experiences, expectations, the context of use, etc. Users from different geographical backgrounds may have different expectations regarding the service quality, based on previous experiences. A user surfing at work could probably be more intolerant about the bad quality of service, as compared to a user using the service on leisure.

Monitoring of the user-perceived performance has two main requirements. First, a model is required, that takes into account all the parameters that influence user-perceived performance of a service; second, a method, which estimates the performance by measuring the above parameters in a fast and scalable fashion [3]. Unfortunately, it is not so easy to measure all these parameters online in real-time, from the network-level.

For ISPs, it is easier to measure the network-dependent factors than asking the each user about her experience subjectively. However, retrieving the application-specific and user-related factors is not simple. It is because ISPs do not have control on the user-end devices. Therefore, conducting subjective experiments in a lab environment, with real users, has been a common practice to model the user-perceived performance. Although, they provide control at the user-end, subjective lab experiments have proven to be away from reality [4].

Another method of estimating user-perceived performance is to investigate those parameters that represent user actions on the application level and provide indications about

user behavior. Utility of a service could be one of the ways to have indications about the interest of users in a service [5]. In [6], user session volumes were shown as the indicator of user behavior and were compared against the service performance. While session volumes may provide an overall picture of the interest of all the users in a network service, it is not easy to infer the interest of a single user. Second, it is also difficult to deduce thresholds on the performance, beyond which a user stops using the service.

Monitoring of the Transmission Control Protocol (TCP) connection terminations on the Web is one of the ways to monitor user-perceived performance degradation of a service [7]. Normally, users press the Stop or Reload button in the web browser to abort an on-going transfer, when it is much slower than their expectations. These interruptions result in early termination of the TCP connections with a Reset (RST) flag from the client side. These RST flags can be monitored passively on the network-level to observe the user behavior.

Before considering the TCP RST flags as being the indication of user behavior, it is important to make sure that a TCP RST flag is generated only when the user interrupts a TCP connection. Therefore, it calls for a detailed classification of TCP end flags, which allows identification of those TCP RST flags that are generated as a result of user interruptions.

In [8], TCP connections are, based on the type of termination, classified as *normal* connections, *abnormal* connections, *unfinished* connections and *interrupted* connections.

In this work, we have performed a systematic study to show in detail the sequence of termination flags, in order to identify the transfers aborted by the users. The sequence of these termination flags occurred as a result of different actions as listed in Table 1, performed in the web browser. Hence, monitoring and classification of the termination type in a real scenario may provide indications about a user's behavior. To ensure a fair and representative comparison, we have conducted a number of controlled experiments with various web browsers. These experiments are done on both smart phones and laptops. The results of experiments on smart phones were presented in [9]. A more detailed study along with a set of termination criteria is presented in this paper.

Network operators and Web service providers can use this knowledge to passively monitor the behavior of users over time, and manage their resources accordingly to guarantee high-quality user experience. The research community working on network Quality of Experience can use this study to validate it against the subjective experiments with real users. Finally, Web users can also use results to choose web browsers that are operating according to the rules defined by the standards.

Hence, our contribution in this paper is two-fold. First, we present the different sequences of the TCP termination flags observed with a number of web browsers. With the help of these results, cooperation between web browsers and Web servers could be improved to raise the performance of Web transfers. Second, we develop a set of criteria, which could be used to identify the user action performed in the web browser. It may help service providers to monitor the user-perceived performance.

The remainder of this paper is organized as follows. Section 2 provides some background information on the Web transfers and the TCP protocol. Section 3 describes the related work. Section 4 gives the methodology used. Section 5 shows the results and discussion on the sequences of the TCP termination flags. Section 6 presents results from passive

measurements. Section 7 proposes a set of user-interruption criteria. Section 8 concludes the paper.

Table 1. User actions in the web browser

Type	Description
Uninterrupted	The user allows the web page to load completely
Kill-browser	The user kills the web browser before the page has been loaded completely
Stop/Reload	The user presses the stop or reload button before the completion of transfer
Link-follow	The user clicks another link or follows a bookmark before the completion of previous transfer

2. TCP and Web transfer

A Web transfer consists of the request of one or more objects from the client to the server and the transfer of requested objects from the server to the client. Figure 1 depicts a Web session between a pair of the client and the server machines.

A Web session is a combination of one or more Web transfers. A Web transfer starts when a user requests for Webpage or a file. A Web page may consist of multiple embedded objects. Each embedded object is retrieved after the client-side web browser requests for the respective object, automatically. The transfer of each object is shown as the ON time in the Figure 1. Two objects in a Web transfer are separated by the active OFF time; the time taken by the client-side web browser to launch the next request automatically, after the transfer of the previous object.

Two subsequent Web transfers in a session are separated by the inactive OFF time, which is also called the user think time. This is the time taken by the user before launching the request for the next page or a file from the same server.

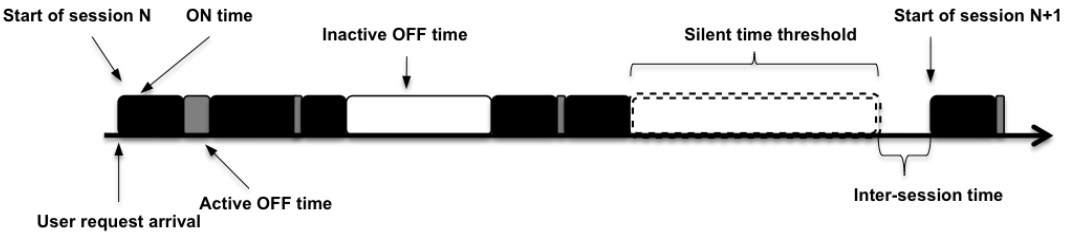

Figure 1. ON-OFF model for Web transfer

The Web traffic is carried by the TCP on the transport layer, hence making TCP the most widely used protocol on the Internet for almost two decades. A TCP connection goes through several states from the connection establishment to the data transfer to the connection termination.

TCP is a reliable stream-oriented protocol [10]. Before the transfer of the stream of bytes, a virtual connection is established. After the connection establishment between two TCPs, a request is made by the client to the server, followed by data transfer from the server. After the end of the data transfer, the connection is closed by going through a proper termination handshake.

TCP connection termination handshake employs the control field in the TCP header to flag the end of a connection. To signal the end of the connection, a segment is sent from either side, with the finish (FIN) flag set in the control field of the TCP header. The other side then responds with a FIN segment to confirm the receipt of the FIN segment. This handshake confirms that the data transfer is completed, and the connection could be closed. Sometimes, the FIN segments also contains last chunk of data in it.

To signal the error conditions, a segment with reset (RST) flag is sent. The RST segment can be sent from one of the sides to deny a connection, if a connection was requested to a nonexistent TCP port. It is also sent when one of the sides aborts an existing TCP to signal an abnormal situation.

Figure 2 illustrates a TCP flow carrying a Web transfer. It starts with a SYN handshake between the client and the server. The client then requests for the file with a HTTP GET request. If the requested file is available on the server, then the server responds with the file in the form of stream of data bytes on the TCP level. It is shown by the TCP flow carrying DATA for a Web transfer in Figure 2. The client acknowledges one or more data segments from the server with an ACK segment. Once the transfer of requested file is completed, the client may request for another file with another HTTP GET request in the same TCP connection, if the client and the server are both supporting persistent TCP connections. Once all the requests from the client are served, the client or server then starts the connection termination handshake procedure, which is shown by FIN segment. The other end then responds with a FIN to terminate the connection.

3. Related work

Evaluation of the user-perceived performance of network services is a complicated task. User perception is very much subjective, which vary heavily from one person to the another. There have been efforts made in order to evaluate and model the user perception. Many of these efforts are limited to the user perception of voice and video applications [3][11][12]. Literature on QoE shows that it has either been drawn as qualitative nature subjective evaluation or quantified as a function of the QoS parameters.

The most common way to understand the user perception of a service performance is to

conduct large number of expensive experiments in a lab environment. Many works reported these subjective experiments. The problem with these experiments is that they do not represent user perception that arises in the real environment. Users are too conscious about the performance in the lab environment than the real environment. They do not have freedom to do their tasks in natural way [4]. Secondly, the numerical scales on which the ratings are taken may not map well with the real emotions of the users.

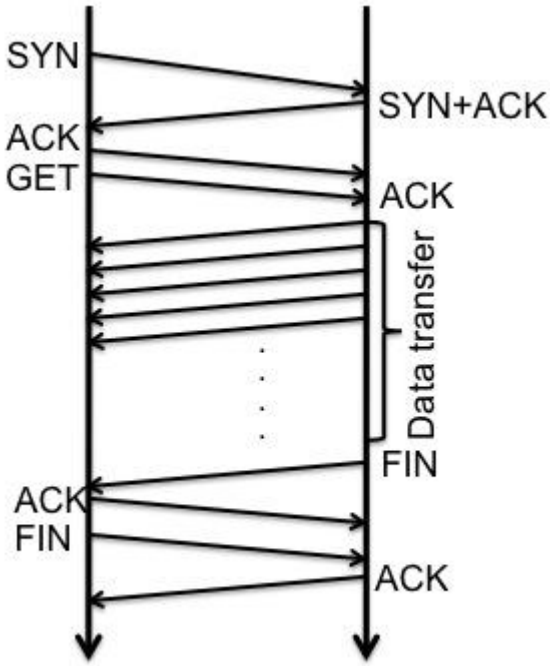

Figure 2. TCP flow carrying data for a Web transfer

In [13], a framework is proposed to capture users' perception while they are using network applications. It requires a subject to click a key whenever she feels dissatisfied with the service. While their framework could be useful for the test environment, it is not easy to have their tool installed on user machines in the operational environment. In [14], authors presented their results on the data, which was collected by an end-host data collection tool. It was concluded that it is very much challenging to get the user feedback over time, particularly when the performance of the network degrades. Authors of [15] collected the user-centric data by a similar tool. The tool allows users to report their irritation with the help of the tool.

Most of the above work required the use of additional tools for the self-reporting of opinions given by the users. We do not find many works where existing infrastructure is used to infer the user perception. Another major point we observe in the literature is that the most of work on user experience has been directed towards the improvement of a particular application rather than the network services. In [16], a wavelet-based criterion is presented for identifying the user-perceived problems passively, based on the link-level measurements.

This work presents a set of criteria to infer the user perception from the client-side TCP RST flags. In [17][18][19], it is shown that the TCP RSTs are generated due to several causes. Therefore, it may not be appropriate to consider TCP RST flags as the straightforward indication of user perception. In [20], authors presented a heuristic to identify the TCP RSTs generated as a result of the interruption from the client. We will present these criteria and its shortcomings in the Section 7.

4. Methodology

We conducted a set of active tests to observe the sequence of TCP termination flags exchanged in both directions. To execute the tests, we established an isolated environment. These tests were conducted by accessing a Webpage on a smart phone or a laptop. The Webpage was located on a local Web server.

Two types of tests were performed: Uninterrupted and interrupted. In the uninterrupted tests, the user issues a Webpage request and then allows the transfer of the Webpage to finish completely. In the interrupted tests, the user aborts an ongoing transfer of Webpage by performing some action in the web browser. The user action could be either pressing the Stop or the Reload button, exiting the web browser or clicking a hyperlink on the Webpage. These actions are further mentioned in Table 1.

In order to study the impact of the content type, three Web pages were developed. One Webpage had simple text, the second one had an image and the third one had a flash video, played in a shockwave player on the Webpage. Since the results of the tests with text and image Web pages were almost similar to each other, we only present the results related to text and to video in the remainder of this paper.

Moreover, tests were first performed on three popular mobile platforms: Windows 6.5 (HTC HD2), Android 2.2 (HTC Desire HD) and Symbian 3.0 (Nokia N8). Built-in web browsers were used on each of these platforms, as external browsers were not supporting the video content. The web browser used by Windows 6.5 is Microsoft Internet Explorer 6.0. The user agent string in the HTML header reports Android's web browser as Mobile Safari and Symbian's web browser as Browser NG, which is used on the Nokia mobile phones. On Android and Symbian platforms, the built-in web browsers use Webkit as the HTML rendering engine, which is an open-source web browser engine [21]. Each of the tests on mobile platforms was conducted with 40 repetitions.

Subsequently, other popular web browsers were also tested. Tests were performed on a laptop equipped with Windows XP operating system. Four Web browsers were tested: Internet Explorer 8, Firefox 3.6, Google Chrome 4.1 and Opera 10.51. All of these web browsers support persistent TCP connections.

Finally, passive measurements were performed on an operational network of an ISP with real users. We identified different browsers from the traffic traces with the help of their user agent strings. After identifying the Web browsers, we carried out further analysis to extract

their TCP terminations connection sequences.

5. Active tests

Table 2 summarizes the different termination types seen from all the experiments. The type of termination here refers to the sequence of terminating flags that were seen at the end of a TCP connection. Five different types of terminations are observed. Sequences of these termination flags occurred as a result of different actions as listed in Table 1, performed in the web browser.

Table 2. TCP connection termination flag sequences

Type of Termination	Description
$F_sF_cR_c$	A FIN from the server followed by a FIN and then one or more RSTs from the client
$R_cF_cR_s$	One or more RSTs from the client followed by a FIN from the client and a RST from the server
$F_s F_c$	A FIN from the server followed by a FIN the client
$F_c R_c$	A FIN from the client followed by one or more RSTs from the client
R_c	One or more RSTs from the client

5.1 Uninterrupted transfers

The bar charts in Figures 3–5 highlight the number of each terminating sequence observed as a consequence of different user actions performed in different web browsers. On Symbian and Android platforms, all the TCP connections ended with a proper FIN handshake. After the data transfer, a FIN from the server is sent which is followed by a FIN from the client-side to end the connection. This type of termination follows the rules as described by the standards [10].

Client-side RST flag: On the Windows platform, however, the text-based Webpage transfer is finished with a FIN from server, followed by a FIN and then a RST flag from the client. This RST flag appears to be a reaction of the client to the ACK received from the server, which triggers the client to immediately shutdown the connection by sending a RST flag. This behavior is found to be consistent in all the transfers.

Multiple TCP connections per transfer: Subsequently, when the video-based Webpage is downloaded from the Windows platform, there is another interesting pattern seen in the connection termination process. After receiving the base file, the client makes a GET request for the video player. It then immediately terminates the connection with a RST flag and initiates the new connection with a SYN handshake. The GET request for the previous file is thus repeated once again and then the video is played in the web browser. The second connection is terminated similarly as was observed in the case of text-based Webpage. Hence, two connections are opened for playing video in the web browser. The connection establishment procedure creates extra overhead, which affects badly the overall speed of the

transfer. The TCP connection also goes into the slow start phase once again. The client-side software should avoid this kind of behavior as the opening of multiple TCP connections per transfer may degrade the performance of the transfer.

5.2 Interrupted transfers

Interrupted transfers are those in which a user aborts an on-going transfer by manually performing any of the three actions (before the end of the download) in web browser: Pressing the stop or reload button, exiting the browser or clicking a hyperlink on the Webpage. The results in Figures 3–5 illustrate that the connection termination pattern is similar when the interruptions are made from Android and Windows platform, while it is slightly different in the case of the Symbian platform.

Server-side RST flag: While using the Symbian platform, a large ratio of TCP connections were terminated with one or more RST flags from the client, followed by a FIN flag from the client and then a RST flag from server. The reason why the server responded with a RST flag is that when it received a RST flag from client, it assumed the connection was already closed and therefore, when it received an additional packet from client (containing a FIN flag) on the same port, it responded with a RST flag to once again signal the end of the connection.

Retransmissions: In a few interrupted transfers on Symbian and Windows, and in the majority of transfers on Android, connections were terminated with a FIN flag followed by one or more RST flags from the client. By looking at the interrupted traffic traces, we found out that, when the client starts the termination process with a FIN flag, then the server responds with the retransmission of previous unacknowledged segments. Upon receiving the retransmitted segments the client tears down the connection by sending one or more RST flags. This kind of anomaly may result in the wrong estimation of loss rates on the network. On the Windows platform, the majority of the connections were terminated with one or more RST flags from the client without any FIN flag.

Indication of user-interrupted transfers: Generally, from all the above observations of interrupted connections, one thing is common that at least one RST flag from the client-side is seen regardless of the platform. The other important evidence about the user-generated interruption is that more than one consecutive RST flags were seen in most of the cases as soon as the user performs an interruption in the web browser.

Indication of transfers not interrupted by the user: On the other hand, a single RST flag is seen if the transfer is not interrupted by user, on which the RST flag is sent automatically by the client-side software.

5.3 Active tests with other popular web browsers

To test other popular web browsers, we further continued our tests on a laptop equipped with the Windows XP operating system. Tests were performed with four web browsers: Internet Explorer, Firefox, Opera and Google Chrome. Figure 6 presents the total number of RST flags observed from all our tests. This figure gives an overall picture of the web browser

generating the highest number of RST flags.

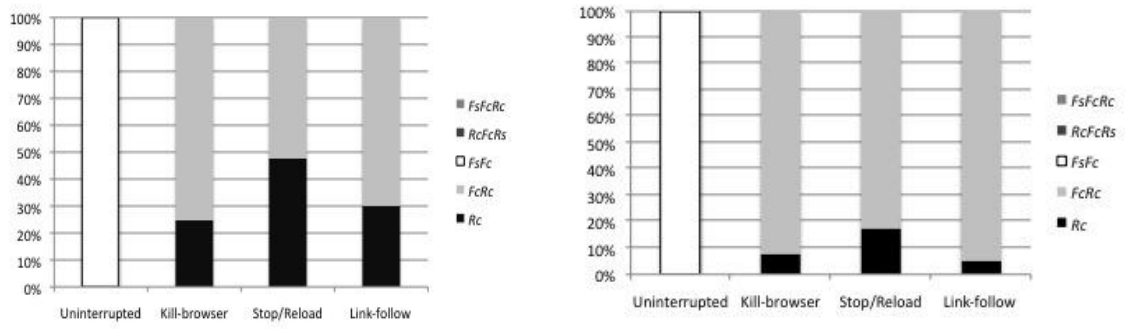

Figure 3. Termination flags on Android. Left: Video page, Right: Text page

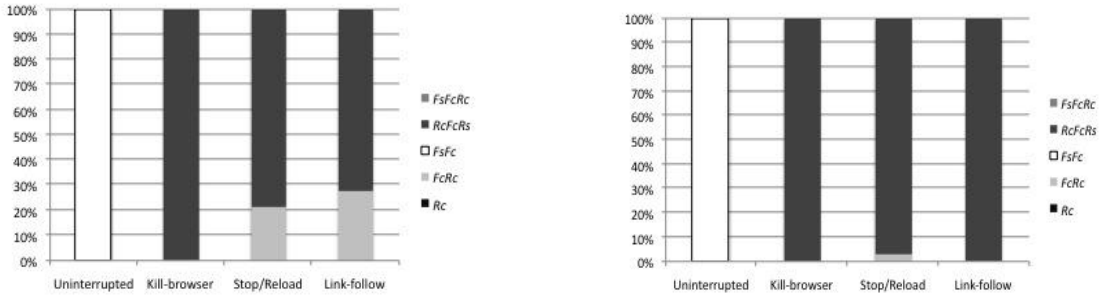

Figure 4. Termination flags on Symbian. Left: Video page, Right: Text page

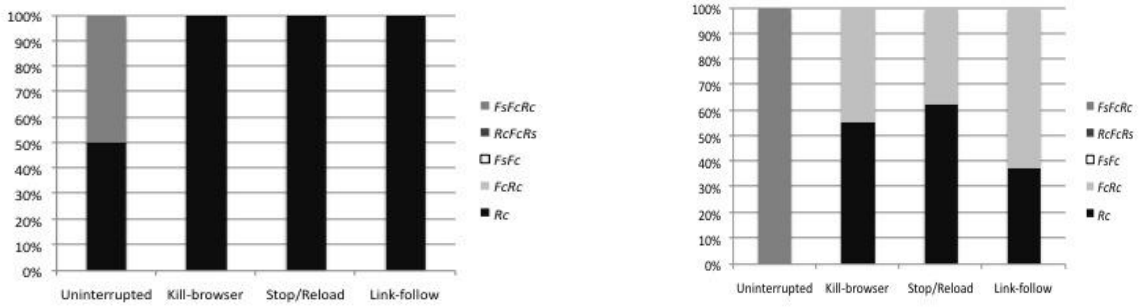

Figure 5. Termination flags on Windows. Left: Video page, Right: Text page

Figure 6 confirms the results we retrieved from mobile web browsers. The Internet Explorer web browser generates more TCP RST flags than any other web browser. The RST flags on Internet Explorer are not only generated when the user aborts a transfer, but also in the cases when a transfer is not aborted by the user. Another observation we got from the experiments is that, the generation of TCP flags is not dependent on the operating system, but on the web browser. We performed these tests on the same operating system and we observed different behavior in terms of TCP termination flags.

Proper FIN handshake: For uninterrupted tests, the connections were terminated with proper FIN handshake in the case of Firefox and Opera web browsers. The server initiates the

FIN handshake by sending a segment with FIN flag after the completion of data transfer. The client then responds with a FIN/ACK segment. It tears down the connection along with the acknowledgement of the last segment of data from the server.

Termination initiation from the client side: In the case of Google Chrome web browser, connections termination starts with a FIN from the client instead of a FIN from the server for uninterrupted tests. After the data transfer, the client initiates termination handshake, by sending a FIN segment. The server then responds with a FIN flag. Hence, the client does not wait for the server time-out but starts tearing down the connection proactively.

RST flag from the client side: The Internet Explorer replies with a RST flag instead of a FIN flag, after receiving a FIN flag from the server. After the data transfer, the server sends a FIN flag, and the client then responds with a RST flag. This behavior indicates the abnormal condition according to the TCP standards. These results indicate that the Internet Explorer does not seem to follow the standards. These tests with the Internet Explorer also confirm our observations we discussed in the case of mobile web browsers. The Internet Explorer in the case of video transfer opens two TCP connections. The first connection is terminated as soon as the video player is requested. The request for the video player is then repeated in the second connection. Due to such behavior of Internet Explorer, we observe a larger number of RST flags in the case of Internet Explorer as compared to other web browsers as displayed in Figure 6.

Interrupted tests: In the case of interrupted tests, we observe a similar behavior in the case of all web browsers. The client terminates a connection with a RST flag as soon as the user aborts a connection in the middle of the transfer. If the client still receives a data segment from the server, after sending a RST flag, then it repeatedly sends RST flags to enforce the termination of the connection.

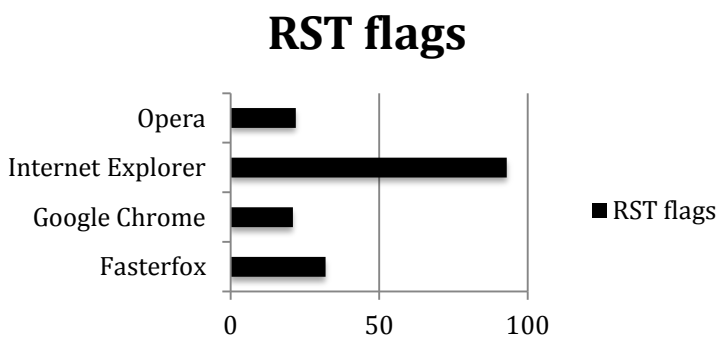

Figure 6. Total number of RST flags

6. Passive measurements on operational network

In this section we describe the connection terminations that we observed on the operational network of an ISP. We only present the web browsers that we considered in our active tests, i.e. Internet Explorer, Firefox, Google Chrome and Opera. We observed a large

variety of connection terminations, i.e. the connections with different sequences of the termination flags. However, we have listed the most common termination flag sequences with each of the web browsers, which account for more than 75% of the connections.

The most common termination type: The server sends a FIN flag after transferring the data, which is followed by a FIN flag from the client side. Generally, this is the most common termination flag sequence observed on the TCP connections launched on the network.

Transfers on Internet Explorer: The most common termination flag sequence observed with the Internet Explorer is the following: The server sends the data, which is immediately followed by one or more FINs from the server. The client then responds the server's FIN with one or more RST flags to tear down the connection. This is quite similar to what we observed in our active tests. There are also significant numbers of connections that are terminated with one or more RSTs from the client side. We infer that these connections are the mix of both terminated by the client-side web browser (in the case of video transfers), as well the user to abort a transfer.

Connection terminations with Firefox and Google Chrome: We see similar termination flag sequences observed with Firefox and Google Chrome. The majority of the connections are terminated with a FIN from the server, followed by a FIN from the client. However, in many cases, the server sends a RST flag after sending a FIN to flag the end of the connection. In these cases, we do not observe any termination flag from the client side.

Connection termination initiation by the client: There is also large number of cases in which the connection termination handshake starts with a FIN flag from the client side followed by a FIN flag from the server side. This is the most common in the case of Opera and second most common in the cases of Firefox and Google Chrome. The client in such cases detects the end of the data transfer and hence, it sends a FIN immediately along with an ACK of last data segments from the server.

Table 3. Connection terminations on operational network

Web browser	**Most Common terminations**	**Description**
Internet Explorer	F_sR_c	One or more FINs from the server followed by one or more RSTs from the client
	F_sF_c	One or more FINs from the server followed by one or more FINs from the client
	R_c	One or more RSTs from the client
Firefox	F_sF_c	One or more FINs from the server followed by one or more FINs from the client
	F_cF_s	One or more FINs from the client followed by one or more FINs from the server

	F_sR_s	One or more FINs from the server followed by one or more RSTs from the client
Google Chrome	F_sF_c	One or more FINs from the server followed by one or more FINs from the client
	F_cF_s	One or more FINs from the client followed by one or more FINs from the server
	F_sR_s	One or more FINs from the server followed by one or more RSTs from the server
Opera	F_cF_s	One or more FINs from the client followed by one or more FINs from the server
	F_sF_c	One or more FINs from the server followed by one or more FINs from the client
	$F_cF_sR_s$	One or more FINs from the client followed by one or more FINs from the server followed by one or more RSTs from the server.

7. The user-interruption criteria

In [20], the authors proposed a user-interruption criterion. This criterion is based on a heuristic to determine those connections, which are interrupted by the users before a transfer is completed. To test this criterion, we applied it on our collected traces with the help of Tstat [22]. Tstat is a TCP statistics and analysis tool that implements the user interruption criterion. We will first explain this criterion and then discuss the outcomes that we got after the application of this criterion on our traces.

According to the criterion mentioned in [20], a connection is terminated by the client if the server sent data but didn't send a FIN or an RST segment, and the client sent an RST segment. These connections were named as *eligible* connections, and are expressed by:

$$Eligible \coloneqq \neg(FIN_s \vee RST_s) \wedge DATA_s \wedge RST_c \quad (1)$$

Where FIN_s is a FIN flag from the server and RST_s is the RST flag from the server. $DATA_s$ is the data transfer from the server and RST_c is the RST flag from the client. "¬" represents "NOT", "∨" represents "OR" and "∧" represents "AND" as logical signs. Hence, the equation says: *"A connection is eligible, if NO FIN OR RST from the server is seen, AND DATA from the server AND the RST flag from the client is observed."*

Although *Eligible* connections represent those terminated by the client, they do not tell whether the connection is terminated by the user or not. The *Eligible* connections could be terminated by the client-side software when the data is already transferred, and the server is idle, waiting for the time-out. In order to identify the connections interrupted by the users during a data transfer, the authors also consider the connection termination time. The interruption criterion is thus expressed by:

$$Interrupted \coloneqq Eligible \wedge \left\{\left(\frac{t_{gap}}{\alpha \cdot \mu_{RTT} + \beta \cdot \sigma_{RTT}}\right)\right\} \leq 1 \quad (2)$$

Where t_{gap} is the time elapsed between the last data segment from the server and the actual flow end. μ_{RTT} and σ_{RTT} are the mean and the standard deviation of the RTT per connection, respectively. Hence, if a connection is *Eligible* and t_{gap} is less than one RTT time, then the connection is said to be interrupted by the user.

We executed Tstat on our interrupted and uninterrupted transfers. We found out that the user interruption criterion was not working accurately for those video transfers, in which Internet Explorer was used as client-side Web browser.

While this interruption criterion determines the connections interrupted within $t_{gap,}$ it does not identify the user action performed in the Web browser that resulted in the interruption of ongoing transfer. The identification of the user action in the Web browser is also important to know, as there could be different motivation behind each user action. For example, users usually press the stop or reload button when they are annoyed. However, they might also follow a link before the completion of a page when they have already seen enough information on the page, which may not be the result of anger or dissatisfaction.

To address the aforementioned shortcomings, we propose a set of criteria. The set of criteria are specified by a finite state machine diagram in Figure 9. There are two types of transitions shown in the diagram in Figure 9. One is the TCP flow transition triggered mainly by the TCP control flags. It is shown with the solid black arrows in the diagram. Another type of transition is the HTTP request transition, which is initiated each time by the arrival of a new HTTP request from the client side. HTTP request transitions are represented with dotted arrows in the diagram. Additionally, there is one more transition shown in the diagram, which takes both TCP and HTTP into the idle state. This transition is shown by dashed arrow. The machine comes out of this state when the user requests for the next web page from the same Web server. We will first describe the TCP flow states in detail, followed by the description of the HTTP transitions.

State 0. Start: The state machine starts with this state as soon as a new request from the user is received. It refers to the user action of opening a new URL, clicking a link on a page or starting a completely new Web session.

State 1. Handshake: When the user requests for a Web page, the handshake process for the connection establishment starts. The Client-side TCP initiates this handshake with a SYN flag, shown by the transition number 1a in the Figure 9.

State 2. Connection established/data transfer: The server responds to the client with a SYN/ACK shown by the transition 2a, which is further acknowledged by the client, and the connection is established. The first HTTP transition is then made with the request for the base file H_B of the page, represented by the dotted arrow 2b. After the connection establishment and H_B request from the client, the data transfer starts. TCP and HTTP both stay in this state until a termination flag (FIN or RST) is seen from any side (client or server). During or after the data transfer, there could be any of the following three more events, which result in transition out of state 2:

State 3. Eligible: If a RST flag is seen from the client before any FIN or RST from the server as represented by 3a, then the connection becomes *eligible*, as defined by Equation 1. This state could be reached before or after the completion of the data transfer (state 2). Sometimes, client-side browsers send a RST flag after the completion of the data transfer. Hence, merely seeing a RST flag from the client does not confirm that a data transfer is interrupted**.** To identify those client-side RST flags indicating interruption of the data transfer, HTTP transactions (request and response) and t_{gap} (see Equation 2) should be taken into account.

State 4. Uninterrupted: If a FIN or a RST flag is received from the server (transition 4a), then the connection is called uninterrupted and hence, TCP goes into state number 4, which is named as ***uninterrupted*** in the diagram. HTTP also moves to the ***uninterrupted*** state, if the FIN or RST flag from the server is seen after the HTTP response code. Subsequently, the client web browser may request the embedded file of the page H_E, automatically. It takes both TCP and HTTP to the data transfer state, as shown by the transition 2d.

State 5. Interrupted: If the RST flag from the client is seen when the data transfer from the server is still going on (shown by transition 5a), then the connection is called interrupted. However, there are two types of such interruptions: One made by the client-side web browser and another one by the user. It is not possible to differentiate between both of them only by observing the TCP flow. Let's recall the video page download case, which was performed on the IE web browser. We observed that the client terminates the data transfer each time after it requests for the video player. In this case, although the TCP connection termination met the interruption criteria, the transfer is not interrupted by the user. In order to identify the TCP connections interrupted by the user and not by the browser, we need to take into account the HTTP request and response messages. For instance, if a TCP connection interruption is followed by the arrival of the last HTTP request, then it shows that the TCP connection is interrupted automatically by the web browser (see Internet Explorer case with the video transfer). Conversely, if the TCP interruption is followed by either of the transitions 2e, 2f and 6b then the connection was interrupted by the user and not by the browser. When a transfer is reloaded by user, request for the last base file H_B is repeated. The transition 2e shows that the Web transfer is reloaded by the user. Similarly, when a user follows a new link or a bookmark (link-follow), before the completion of the previous transfer, clients sends the HTTP message containing the request for a completely new file. The transition 2f shows this link-follow behavior. Finally, if the user interrupts a transfer by pressing the stop button or by killing the web browser, then it does not immediately trigger any new HTTP request message

but followed by the user think time (the time user takes before launching the new request for a page). It is illustrated by the transition 6b in the state diagram.

State 6. Idle/User think time: Let's recall the ON-OFF model of the Web transfer described previously in Figure 1, where each web transfer is followed by *inactive OFF time*. This state could be reached by either of the two possibilities. First, if a transfer is completed without any interruption by the user, the user takes some time for reading the page or thinking about the next link before launching the request for the new page. Second, when the user interrupts the previous transfer by pressing the stop button or killing the web browser, then it takes a period of silence time before the user requests for a new page.

Summing up the above discussion, we observed from the active and passive tests that the web browsers often do not follow the given TCP standards. We classified several types of TCP connection terminations between the client and the server. The authors in [20] proposed a criterion, which is quite helpful in detecting those TCP connections that are interrupted by the client before the end of data transfer from the server. However, there could be two causes of such client-side interruptions: the client web browser or the user. To identify the interruptions done by the user and not by the web browser, we presented a set of criteria. According to the criteria, the HTTP request and response messages need to be taken into account beside the TCP flags in order to identify the transfers interrupted by the users.

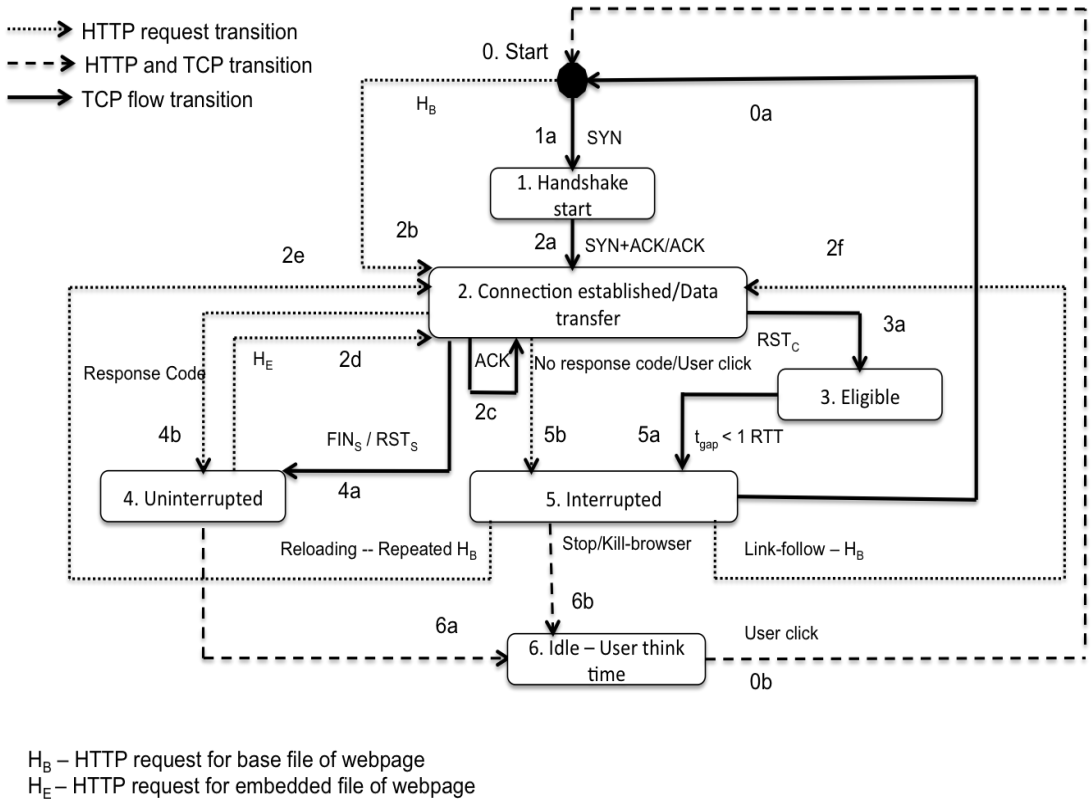

Figure 6. State diagram of a Web transfer

8. Conclusions

In this paper, we proposed a set of criteria to monitor the user actions in the Web browser. The monitoring of these actions can provide indications about user reaction to the network performance. These are based on the TCP interruptions and HTTP requests. In order to study whether TCP RST flags could be used to monitor the user behavior on the Web, we conducted several experiments with different web browsers. We found out that some web browsers send TCP RST flags without any interruption by the user. Therefore, TCP RST flags alone could not be used to monitor the user actions in the web browser. However, TCP RST flags along with the knowledge of HTTP request and response messages can allow us to passively monitor the user-perceived performance.

Additionally, we also showed some of the abnormal behaviors by the web browsers. We believe that there is a need of a better mechanism for communication between web browsers and the web servers, in order to improve the performance of TCP connections and raise the user experience.

Acknowledgment

This work has been sponsored by the Network of Excellence Euro-NF (FP7 ICT N° 216366) and was started within Euro-NF's Specific Joint Research Project QoEWeb. We would also like to thank our master students Adeel Ashfaq and Umer Bilal for their work on the active tests in the lab environment.

References

[1] Labovitz C., Iekel-Johnson S., Mcpherson D., Oberheide J., Jahanian F., "Internet Inter-domain Traffic". ACM SIGCOMM 2010. New Delhi, India, August 2010. http://dx.doi.org/10.1145/1851182.1851194

[2] PhD comics. "http://www.phdcomics.com/comics/archive.php?comicid=1456". Last seen: Feb 3, 2012.

[3] Tao S., Apostolopoulos J., Guerin R., "Real-time Monitoring of Video Quality in IP Networks". IEEE/ACM Transactions on Networking. Volume 16, Issue 5, October 2008, pp. 1052-1065. http://dx.doi.org/10.1109/TNET.2007.910617

[4] De Moor K., Ketyko I., Joseph W., Deryckere T., De Marez L., Martens L., Verleye G., "Proposed Framework for Evaluating Quality of Experience in a Mobile, Testbed-Oriented Living Lab Settings". Mobile Networks and Applications. Volume 15, Number 3, 2010, pp. 378-391. http://dx.doi.org/10.1007/s11036-010-0223-0

[5] Fiedler M., Tutschku K., Chevul S., Isaksson L., Binzenhöfer A., "Throughput Utility Function: Assessing Network Impact on Mobile Services". Lecture Notes in Computer Science. Vol. 3883, 2006, pp. 242.254. http://dx.doi.org/10.1007/11750673_19

[6] Shaikh J., Fiedler M., Collange D., "Quality of Experience from User and Network Perspectives". Annals of Telecommunication. Volume 65, Number 1-2. Pp. 47-57. 2010.

http://dx.doi.org/10.1007/s12243-009-0142-x
[7] Shaikh J., Fiedler M., Arlos P., Minhas T., Collange D., "Classification of TCP Termination Behaviors for Mobile Web". In the proceedings of 1st workshop on Smart Communication Protocols and Algorithms (SCPA 2011), collocated with IEEE Globecom, Houston, USA, December 2011. http://dx.doi.org/10.1007/10.1109/GLOCOMW.2011.6162351
[8] Collange D., Hajji M., Shaikh J., Fiedler M., Arlos P., "User Impatience and Network Performance". 8th Euro-NF Conference on Next Generation Internet (NGI 2012), Karlskrona, Sweden, June 2012.
[9] Shaikh J., Fiedler M., Minhas T., Arlos P., Collange D., "Passive Methods for the Assessment of User-perceived Quality of Delivery", 7th Swedish National Computer Networking Conference (SNCNW 2011), Linköping, Sweden, June 2011.
[10] Postel J., "Transmission Control Protocol", RFC 793. September 1981, http://tools.ietf.org/html/rfc793
[11] Perez P., Macias J., Ruiz J., Garcia N., "Effect of Packet Loss on Video Quality of Experience". Bell Labs Technical Journal. Vol. 16, Issue 1. Pp. 91-104. June 2001. http://dx.doi.org/10.1002/bltj.20488
[12] Jiang W., Schulzrinne H., "Modeling of Packet Loss and Delay and Their Effect on Real-time Multimedia Service Quality". In Proceedings of NOSSDAV, 2000.
[13] Chen K., Tu C., Xiao W., "OneClick: A Framework for Measuring Network Quality of Experience". In Proceedings of INFOCOM 2009. Rio De Janeiro, Brazil. April 2009. http://dx.doi.org/10.1109/INFCOM.2009.5061978
[14] Diana Joumblatt, Renata Teixeira Jaideep Chandrashekar, Nina Taft "Performance of Networked Applications:The Challenges in Capturing the User's Perception". In the first workshop on Measurements Up the Stack 2011. Toronto, Canada. August 2011. http://dx.doi.org/10.1145/2018602.2018612
[15] Miller S., Mondal A., Potharaju R., Dinda P., Kuzmanovic A., "Understanding End-user Perception of Network Problems". In the first workshop on Measurements Up the Stack 2011. Toronto, Canada. August 2011. http://dx.doi.org/10.1145/2018602.2018613
[16] Shaikh J., Fiedler M., Collange D., Arlos P., "Modeling and Analysis of Web Usage and Experience Based on Link-Level Measurements". 24th International Teletraffic Congress (ITC-24), Krakov, Poland, September 2011.
[17] Arlitt M., Williamson C., " An Analysis of TCP Reset Behaviour on the Internet". ACM Sigcomm Computer Communication Review. Vol. 35, Issue 1. January 2005. http://dx.doi.org/ 10.1145/1052812.1052823
[18] John W., Tafvelin S., Olovsson T., "Trends and Difference in Connection-Behavior within Classes of Internet backbone Traffic". Proceedings of the 9th International Conference on Passive and Active Measurement. 2008. http://dx.doi.org/10.1007/978-3-540-79232-1_20
[19] Fukuda K., "An Analysis of Longitudinal TCP Passive Measurements". Lecture Notes in Computer Science. Vol. 6613/2011. Pp.29-36. 2011. http://dx.doi.org/10.1007/978-3-642-20305-3_3
[20] Rossi D., Mellia M., Casetti C., "User Patience and the Web: A Hands-on Investigation". In the Proceedings of IEEE Globecom 2003.

http://dx.doi.org/10.1109/GLOCOM.2003.1259011
[21] http://www.webkit.org, Last visited: Feb 04, 2012.
[22] Tstat TCP Statistic and Analysis Tool, http://tstat.tlc.polito.it/index.shtml, Last visited: Feb 06, 2012.