Measured HTTP Performance and Fun Factors

Joachim Charzinski

Siemens AG Munich, Germany, Tel. +49.89.722.46803, Fax +49.89.722.26877, e-mail: j.charzinski@ieee.org

Abstract

Recent work has emphasized the importance of pure delay components as well as rate components in the user perceived performance of elastic Internet applications, namely Web browsing. “Fun factors” have been previously introduced to describe the obtained performance with respect to the maximum possible performance on a scale of zero (no fun) to one (maximum fun).

In this paper, several options for defining such fun factors using delay and rate components are presented. In order to better understand the influence of delays and transmission rates, two extensive traffic traces have been evaluated (a) to yield information about the way browsers use persistent and parallel HTTP/TCP connections to access Web servers and (b) to serve as an example for quantifying quality of service with fun factors.

Keywords: Internet Traffic, Traffic Measurement, HTTP Performance, Parallel Connections, QoS for Elastic Applications, Fun Factor

ITC Topics: IP services and QoS / Traffic modeling and measurements / Performance issues related to traffic

1 Introduction

Accessing Web pages is the main cause for traffic and the main interest of users in today’s Internet. Consequently, the quality of service associated with Web browsing is taken as representative for the service possible in the Internet. While in the late 1990s the focus of research was on characterising Web traffic and user behaviour, several recent studies have aimed at quantifying the delay performance of the Internet and HTTP transfers over the Internet [1, 2, 3, 4, 5]. The persistent connection mechanism introduced in HTTP version 1.1 to reduce the latency which is due to connection set-up times has been applied in browsers sending HTTP version 1.0 GET requests by using the “keep-alive” option, leading to a significant fraction of connections serving more than one request even in times where HTTP 1.1 was only rarely used [6].

Active measurement studies have been conducted to assess the delay performance of HTTP in the Internet, having recognized latency as a main performance problem in Web access. Krishnamurthy and Wills [7] analyzed the influence of HTTP protocol options on performance by actively testing download times from selected sites. Similarly, Huitema and Weerahandi [8] investigated the performance of DNS servers by actively requesting name resolution and comparing the performance of different servers and the evolution of loss rates and response times over a period of six months. Barford and Crovella [1, 2] attributed latencies to network and server influences through active HTTP experiments while simultaneously observing packet delays and loss rates in the Internet between selected sites. Habib and Abrams [4] used repeated active measurements on 290 different URLs to separate four delay contributions: DNS lookup time, connection set-up time, the delay between a GET request and reception of the first byte and the actual download time. Cohen and Kaplan [3] used a simplified Web browser re-fetching URLs previously logged in their research lab’s proxy server and compared the performance of several strategies to reduce the latency of DNS lookups, connection set-ups and GET request delays.

From the user’s perspective, the latency between requesting a new link and getting the contents displayed on the computer monitor is the most relevant performance measure [9]. Due to the extreme variation in file sizes retrieved during a download [10], a rate based performance measure is more suitable than a delay based measure for the actual loading phase of a download.

In an extension to the rate based performance characterisation approaches in[11], connection based performance measurements in [12] and single element based performance evaluations in [5], this paper is dedicated to analyzing the performance received when multiple Web elements are loaded in reaction to a single user request in multple connections, and to defining and evaluating a simple performance measure – a “fun factor” – that is able to capture both delay and rate based performance characteristics. The behaviour related to persistent and parallel connections is highly dependent on browser (and server) software characteristics. Consequently, the passive measurements used in this paper are less of a controlled experiment but better represent the real-world performance in contrast to the above mentioned active measurements.

After introducing several options for extending the “fun factor” from [11] to (a) capture delay as well as rate performance and (b) allow more realistic target values in Sec. 2, we discuss measurement and trace evaluation details in Sec. 3 needed to exctract the information from traces used to determine the proposed fun factors in the presence of parallel persistent connections. Distributions of characteristic values are given and example fun factor results are presented for two traces in Sec. 4.

2 Fun Factors

The notion of “fun factors” has been introduced in [11] to describe user perceived quality of service for elastic applications by a number between 0 (no quality) and 1 (full quality). The term was deliberately chosen to indicate that elastic applications can very well cope with a wide variety of available bit rates in the network but the joy of using such an application is determined by the time it takes to complete a given task. As some of the characteristic distributions in the Web show heavy tails [10], it is not advisable to use downloading times directly but rather to relate them to the size of the item to be loaded, leading to a rate based measure that can be normalized e.g. to the access line rate or to another target download speed. In this paper, we discuss further definition options of fun factors in order to take the high portion of serial packet transmissions in HTTP downloads [5] into account.

2.1 Serial Transmissions in HTTP

Even though the current versions 1.0 and 1.1 of HTTP allow keeping connections open for multiple downloads, a significant amount of delay is still due to serial processing, i.e. client and server exchange single packets and wait for the other side before they proceed.

Fig. 1 gives an idealized sketch of the steps necessary to load a single item after a user instructed the browser software e.g. by clicking on a hyperlink. After receiving the click, the browser tries to get the hostname resolved to open a TCP connection to. If the IP address for this name is not already known, a query is sent to a DNS server (instants 1 and 2). As soon as the IP address to contact is known, a TCP connection is opened (instants 3–5) and an HTTP GET request is transmitted in the established connection (instant 6). The instant when the first and last data packets are received in response to this request are labeled 7 and 8. If this item was the only one to be loaded, it is displayed by the Web browser for the user to view.

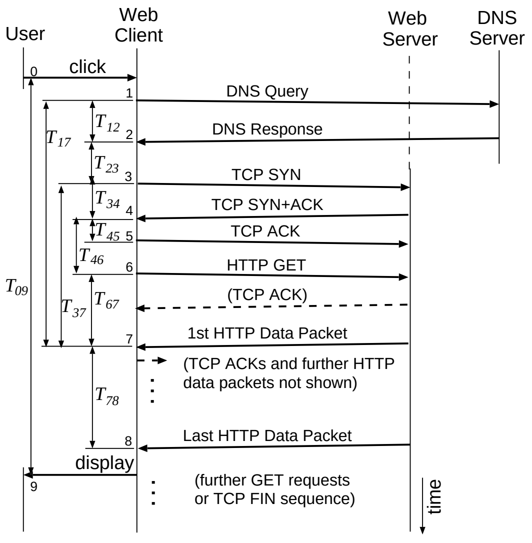

Fig. 1: DNS, TCP and HTTP message sequence for a single HTTP download.

In reality, this simple process is augmented by two methods: Connections are kept open (HTTP version 1.0 “keep-alive" or version 2.0 "persistent" connections) so that for following requests to the same Web server, the connection establishment phase can be saved. In addition, current browsers use parallel connections to increase the client's fair share on congested links as well as effectively trying to approach a *Processor Sharing* service discipline, which prevents exceptionally large elements from completely blocking downloads of smaller ones.

2.2 Definition Options for Fun Factors

Let the random variables T_D, B_L and T_L denote the initial delay before loading an item, the item's size and its observed loading time, respectively. Using $R_L = B_L/T_L$ for the observed rate during T_L and r_t for a target bit rate, the fun factor defined in [11] is given by

$$\Phi_R = \min\{1, (R_L/r_t)\} \tag{1}$$

where in [11] it was suggested to set r_t to the access line rate if this was expected to significantly limit the loading speed. Including the minimum function allows to set target values lower than physical limitations, yielding a fun factor of 1 if the target value is exceeded.

There are several ways to combine delay and rate measures into a single "fun factor". Using a delay target d_t for the pure waiting phases of the loading process, similar to the rate target r_t, the following definition options can be applied:

$$\Phi_1 = \frac{1}{2}\left(\min\left\{1, \frac{R_L}{r_t}\right\} + \min\left\{1, \frac{d_t}{T_D}\right\}\right) \tag{2}$$

$$\Phi_2 = \min\left\{\min\left\{1, \frac{R_L}{r_t}\right\}, \min\left\{1, \frac{d_t}{T_D}\right\}\right\} \tag{3}$$

$$\Phi_3 = \min\left\{1, \sqrt{\frac{R_L \cdot d_t}{r_t \cdot T_D}}\right\} \tag{4}$$

$$\Phi_4 = \min\left\{1, \frac{d_t + B_L/r_t}{T_D + T_L}\right\} \tag{5}$$

The behaviour of the different definitions is sketched in contour plots in Fig. 2. If the rate or the delay component has very bad performance and the other is very good, averaging the two values can still (wrongly) lead to acceptable numeric results for Φ_1. On the other hand, the minimum function in Φ_2 completely prevents any compensation of long delays by faster downloads or of slow downloads by shorter delays. A simple approach for such a compensation is used in Φ_3. A similar idea is followed in the definition of Φ_4, which transforms the rate target into a part of the delay target, allowing to take the relative relevance of the delay and rate dominated phases into account. The effect of this relative relevance is demonstrated for an assumed transfer size of $B_L = d_t r_t$ in Fig. 2 d) and $B_L = 10 d_t r_t$ in Fig. 2 e). The longer transfer reduces the relevance of the initial delay, which is reflected in an decreased dependence of Φ_4 on T_D in Fig. 2 e).

Note that the definition of these fun factors is on a per-download or per-activity basis, i.e. mean values or distributions of fun factors will consider one sample per activity. On the other hand, mathematical analyses often yield

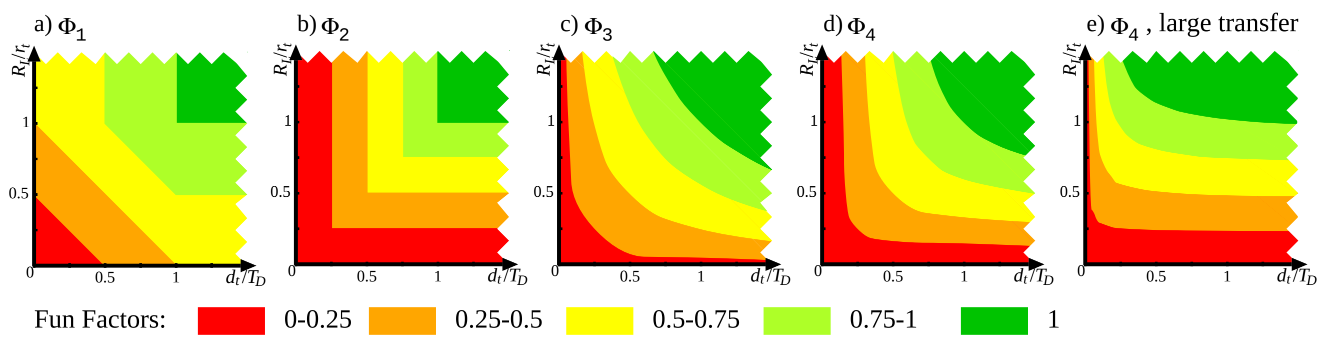

Fig. 2: Contour plots of fun factors versus d_t/T_D and R_L/r_T. a) Φ_1 from (2), b) Φ_2 from (3), c) Φ_3 from (4), d) Φ_4 from (5) for $B_L = d_t r_t$, d) Φ_4 from (5) for $B_L = 10 d_t r_t$.

time based statistics, i.e. the samples are either weighted with the duration of the corresponding activity, or results describe time averages of an underlying system state. Consequently, the results obtained from different evaluations differ if there is a correlation between the measure and the duration of the corresponding activity.

3 Measurements

3.1 Trace Collection

The results presented in Sec. 4 have been obtained from two extensive packet traces. Traffic characteristics observed in these traces have been published before [6, 12, 5] with a focus on other measures than presented here. Both traces were collected using a modified version of Tcpdump [13] on a FreeBSD computer connected to a local Ethernet segment between client computers and their connection to the Internet. The clients observed in Trace A had fixed ADSL access lines configured to 2.5Mbit/s downstream and 384kbit/s upstream, connecting them to their University's campus network through a 100 Mbit/s link. The campus network in turn offered a 34 Mbit/s connection to the Internet via the German research network. The clients observed in Trace B used Modems or one or two ISDN channels with optional data compression to dial into a local ISP network, which offered a local server and was connected to the Internet through a 128 kbit/s link. For more details on the measurement setup see [6].

3.2 Trace Evaluation

The measurement point between on the access network between clients and core network allowed us to record the full traffic from and to users' computers. However, when considering delays, the additional delay on the actual access line (ADSL, modem or ISDN) between measurement point and client computer must be taken into account. This delay was found to be very small (at most a few ms) for ADSL access and rather large on some modem lines (more than 100 ms) in a previous study [5]. For the whole study, a client view is assumed, so that a distinction between network and server delays is not necessary.

The packet based traces were evaluated by analyzing the time stamps, sizes and TCP flags of the recorded packets as well as information if the packet contained an HTTP GET request. In the analysis, a separate HTTP/TCP state machine was run for each HTTP/TCP connection, delivering connection start, GET request, download start, download end and connection end events to a per-client calendar. After the end of a client's activity, this calendar was played out to trace the numbers of parallel connections, connections waiting for a download start after a GET request was transmitted, and connections actively receiving downloaded data. The separation between reading the trace and collecting numbers of parallel connections was necessary as some events like the end of a download could only be infered from the trace after reception of the following packet.

In contrast to [14] and [15], our separation between main and inline objects is based not only on the delay between *GET requests* but rather on the separation between downloads: The number of parallel connections waiting for an answer to a GET request ($N_{GETwait}$) and the number of parallel connections currently receiving data ($N_{Loading}$) are observed. When $N_{active} = N_{GETwait} + N_{Loading} = 0$ during a time interval of more than one second, the user is supposed to view the contents whereas periods of $N_{active} > 0$ are assumed to be due to a single user click. Those periods are in the following denoted as "activities" or "one click" flows. In this way, requests issued serially in one persistent or keep-alive connection as well as requests handled concurrently in parallel connections can be brought into relation.

For the evaluation of fun factors in Sec. 4.3, activities are separated into pure waiting time ($N_{GETwait} > 0$ and $N_{Loading} = 0$) and loading periods ($N_{Loading} > 0$), the cumulative duration of which is used for T_D and T_L, respectively. The total amount of data loaded in the sum of all loading sub-phases of an activity is taken to be B_L.

The distribution of the number of parallel connections in different states were recorded on a per-time basis whereas distributions of the number of requests per flow or the downstream rate achieved per flow were recorded on a per-sample basis. Confidence intervals have been added to some graphs, computing the Student-t confidence for the respective value from batches of 10^6 total evaluated packets each.

4 Results

In [6], all statistical characteristics were evaluated on a per-flow basis with a flow being defined as port-to-port (P2P), host-to-host (H2H) or a complete HTTP client session (CL). In [5], single requests within persistent or keep-alive connections were isolated. In this paper, the concept of a series of requests following a user action (click) is added to the above. Tab. 1 summarizes the number of P2P, H2H, CL flows, GET requests and activity phases following a user click isolated from traces A and B. In order to allow a comparison between these units of transfer, in contrast to [6], only those flows have been counted that contained at least one GET request. The number of clicks per client session observed in Trace A is twice as large as in Trace B, an effect that is most likely due to the difference in tariffs – Trace A was collected in a flat-rate environment whereas the users observed in Trace B had to pay a time-based fee for the telephone line based access to their ISP.

Trace	P2P Flows	H2H Flows	CL Flows	GET Requests	Activity Phases
A	456000	42200	2200	862000	132000
B	740000	88000	6200	1.3 Mio.	138000

Tab. 1: Number of HTTP P2P (port to port), H2H (host to host) and CL (client session) flows as well as GET requests and download activity phases (user clicks) isolated in Traces A and B.

4.1 Single Requests and Persistent Connections

The main result from [5] is repeated in Fig. 3. Based on the numbering of instants introduced in Fig. 1, the mean delay observed at the measurement point during the different phases of a connection is visualized for Traces A and B. Although the DNS lookup and connection establishment phases constitute an important source of delay [12, 5, 8], these are not considered further in this paper as it seems impossible to reliably relate them to user perceived delay in a scenario of parallel connections and multiple downloads per connection when some Web browsers open connections in advance well before using them [5].

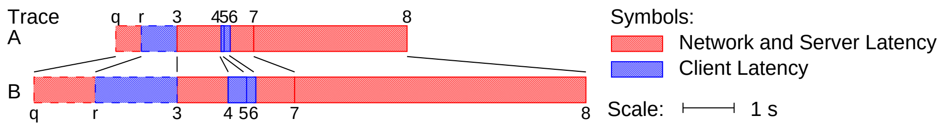

Fig. 3: Mean latencies during a single HTTP item download in a TCP connection. Instant numbers refer to Fig. 1. In contrast to [5], instants 1 and 2 refer to a single DNS query here. The DNS query-response sequence is dashed as it is only needed before every 9th TCP connection setup.

In a single connection, the mean values sketched in Fig. 3 indicate the time to get started before the first item is loaded. Today, most HTTP browsers nowadays support and use "keep-alive" (HTTP version 1.0) or "persistent" (HTTP version 1.1) HTTP/TCP connections in order to save on connection establishment overhead for Web pages that contain multiple elements to be fetched from the same Web server, e.g. icons, images, frames, style sheets, Java applets. In that case, the connection is not closed by the server after the transfer of one element, allowing the client to issue further HTTP GET requests. The number of GET requests observed in the same connection is discussed along with Fig. 7 in Sec. 4.2.

In Fig. 4, the distribution of the time between the end of one transfer and the next GET request in the same HTTP/TCP connection is given for Traces A and B. Using the numbering of instants from Fig. 1, this time interval is called T_{86}. In addition, the viewing time defined as the time between two user click triggered download activity phases (see Sec. 3.2), which in the notation of Fig. 1 is called T_{90}, is plotted. Comparing the distributions, we find that the tail distribution of T_{86} is due to viewing times, i.e. browsers using the same TCP connection for loading objects in successive activity phases. This tail of the complementary distribution function $C(x)$ has an exponent of $\alpha \approx 1.5$ in the power law $C(x) \sim x^{-\alpha}$. On the other hand, there is a high probability of reusing an open TCP connection within less than one second, which is due to consecutive item loads during one activity period.

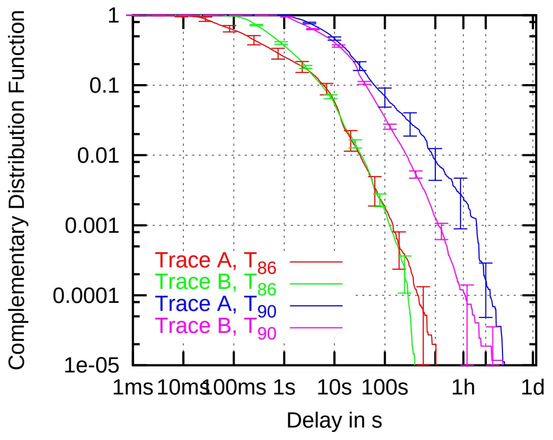

Fig. 4: Distributions of delay T_{86} between end of transfer and next GET in the same connection and of the viewing time T_{90} between the end of an activity phase and the start of the next. 95 % confidendce invervals have been added at every 50th data point.

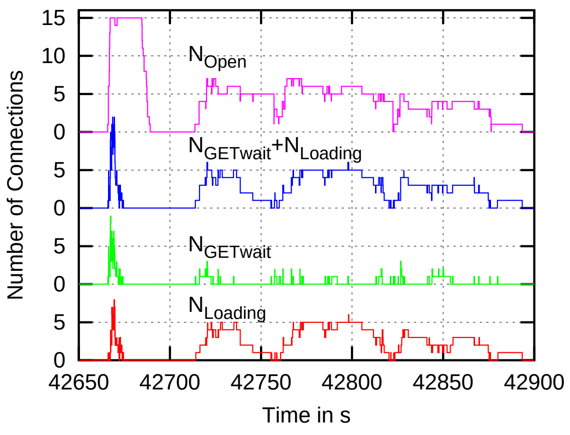

Fig. 5: Excerpt trace for one client showing the evolution of the number of parallel connections, connections waiting for data after sending a GET request, connections receiving data and the sum of GET waiting and receiving connections over 250 s (roughly 4 min).

4.2 Parallel Connections

In order to further save on sequential waiting times, browsers often open multiple parallel connections to the same server or to different Web servers. Fig. 5 traces the number of concurrently active HTTP/TCP connections from the same client computer as a function of time. In this excerpt, a maximum of 15 parallel connections was observed, whereas in the whole traces, this number is as large as 35 (Trace A) or even 82 (Trace B). In Trace B, during 31 % of a client's HTTP session, there was no connection active at all, during 42 % of the time there were one or two connections and in 5 % of the time there were more than 6 connections open at the same time.

The distributions of the number of simultaneously active HTTP/TCP connections in Fig. 6 have been conditioned on the time when there is at least one connection in the corresponding state. This probability was found to be 3.3(11) % for GET wait, 15(48) % for loading and 17(51) % for GET wait or loading connections in Trace A (Trace B), indicating that the ratio of loading to idle phases (or waiting to viewing, from the users' point of view) was much higher in the low speed access scenario of Trace B than with the high-speed access of Trace A. When there is a large number of parallel open connections, these are either idle or receiving data. Whether the high number of parallel connections was due to users deviating from the default configuration values of their browsers or using multiple browser windows in parallel could not be determined as both traces were recorded in the "real world" where users were free to configure and use their software the way they wished.

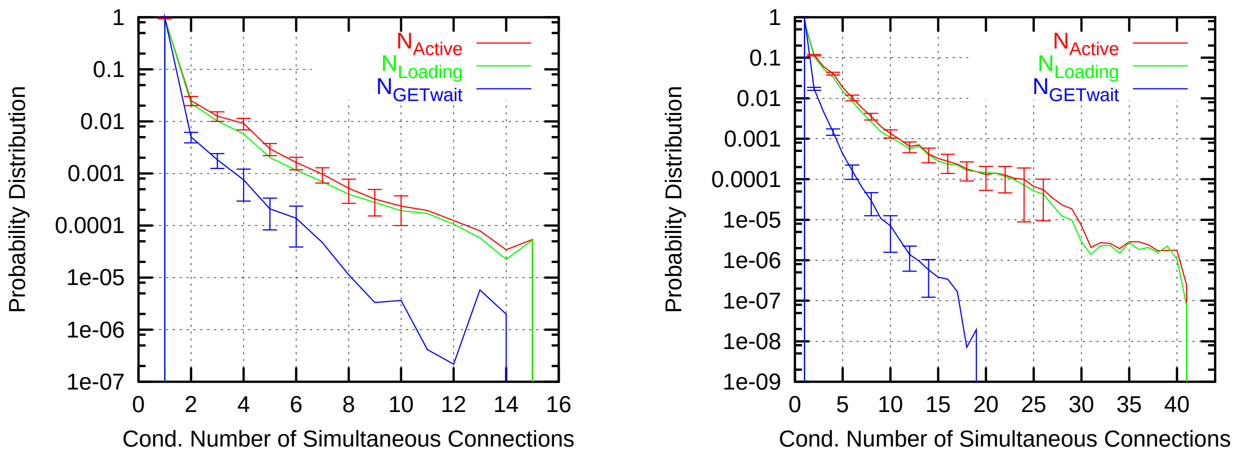

Fig. 6: Distributions of $N_{Active} = N_{GETwait} + N_{Loading}$, $N_{GETwait}$ and $N_{Loading}$, each conditioned on the the respective random variabel being > 0. 95 % confidence indicated for every second value (Trace A: each value) as long as they are less than the value itself. Left: Trace A, right: Trace B.

Fig. 7 shows how the single elements are used on different levels of aggregation. While 1.9 (1.8) GET requests are transmitted in an average HTTP/TCP connection, the average activity period triggered by a click requests 6.8 (6.9) items. The average number of requests in a host-to-host flow was 20.5 (14.9) and in a complete client HTTP session there were 395 (209) items requested in Trace A (Trace B). This is an indication that the general behaviour of Web browsers and the structure of Web pages did not change significantly between the two measurements whereas the number of items requested from the same server or during a complete client session was restricted by the lower session durations in Trace B due to the different tariffing structure (see above).

The distributions of download rates observed during a single item load, contiguous activity sub-periods of $N_{Loading} > 0$, the $N_{Loading} > 0$ parts of activity periods and average rates during complete activity periods are depicted in

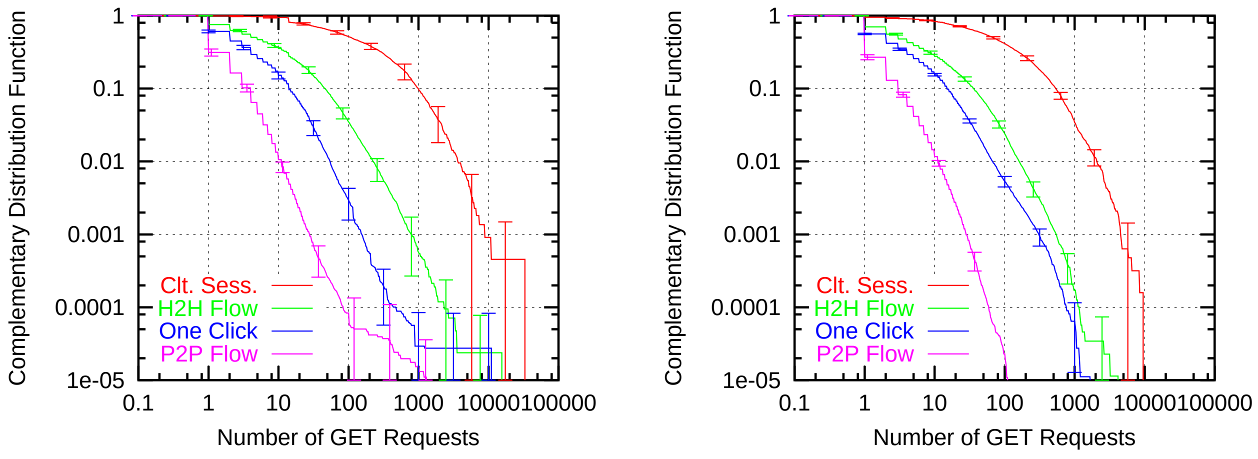

Fig. 7: Number of GET requests in an activity period and on different flow levels. 95 % confidence intervals have been added for every 50th data point. Left: Trace A, right: Trace B.

Fig. 8. As a significant number of requests leads to the transmission of only one data packet, the corresponding link rate of 10 Mbit/s observed at the point of measurement is recorded in those cases. There is also a small fraction of rates measured above 10 Mbit/s, which is due to improper time stamps recorded by the monitoring PCs. Apart from this maximum rate limited by the Ethernet feeder link, the distribution of bit rates received in both traces were found to agree very well with log-normal distributions.

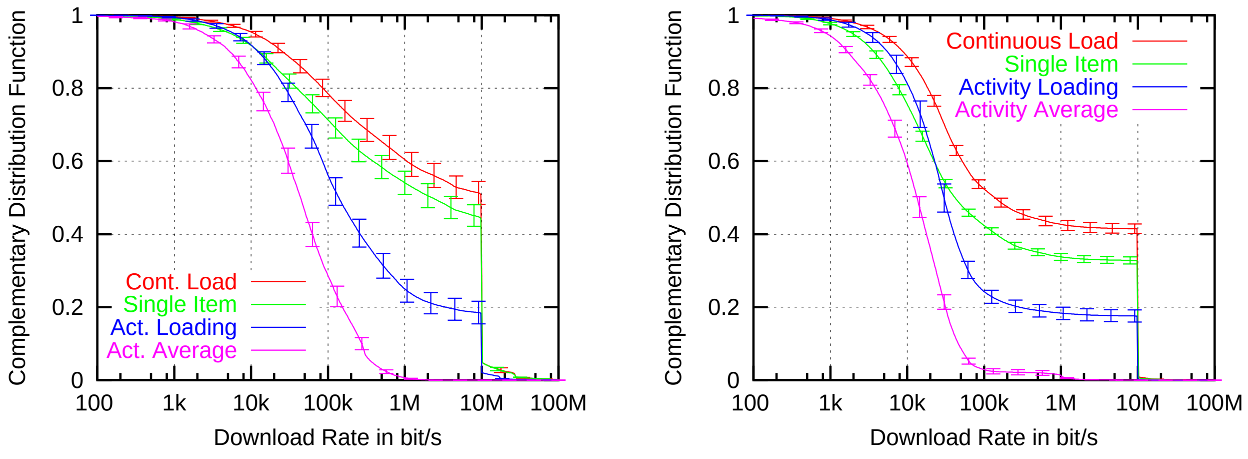

Fig. 8: Download speed during activity phases. 95 % confidence intervals have been added at every 30th data point. Left: Trace A, right: Trace B.

The rate obtained by a client computer is essentially limited by three factors: (a) the access line rate, (b) the rate available in the world-wide Internet and (c) the rate delivered by the server. The difference in access and trunk line rates is responsible for the difference between the rate distributions observed in Traces A and B. The rate available from the Internet can be increased by running parallel connections and thus increasing one's own "fair share" of the total rate available on a congested link. It is not possible to judge from the present traces what the individual connection rates would have been if no other connections had been competing with them on the access line. However, the fact that the difference between the per-GET and continuous-load rate distributions, which include all parallel loading connections, is comparable in both traces, leads us to the conclusion that there is actually a gain from using multiple connections in parallel. A rate of at least 100 kbit/s was reached by 71$\pm$3% (42$\pm$1%) of all single item loads, in 79$\pm$3% (52$\pm$1%) of all continuous loading phases, 56$\pm$3% (24$\pm$2)% of all accumulated loading parts of activities and 29$\pm$3% (29$\pm$1%) of all activity averages in Trace A (Trace B). Independent of this possible gain in downloading speed, the usage of parallel connections for loading a Web page serves the purpose of reducing the chance of a large item's blocking all other element downloads during its transfer by approximating a Processor Sharing service discipline, which is quite insensitive to heavy-tailed service times [16].

Fig. 9 gives the complementary distribution of durations on different levels. As there is a negative correlation between the number of requests served per HTTP/TCP connection and their average size, and connections often last for several activity periods, the distribution of "One Click" loading times converges towards the distributions of single item loading times for larger values whereas loading times below 100ms only occur in single item loads. Starting from the distribution of single item loading times, the distributions of P2P, H2H flows and complete client HTTP sessions are approximately similar, shifted by one order of magnitude each on the time scale.

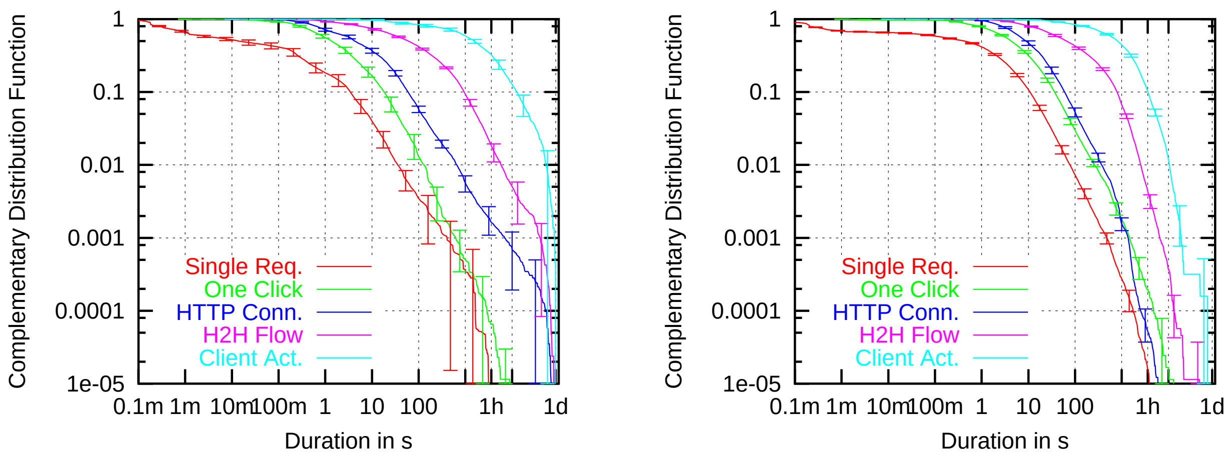

Fig. 9: Duration distributions of request downloads and different flows. Error bars have been added at every 50th data point only to maintain readability. Left: Trace A, right: Trace B.

These heavy tails are due to the correspondings heavy tails in the item size distributions (see [10, 6]) and in the number of items transferred on each level. The size characteristics naturally have a big influence on the number of packets received for each item, which is roughly summarized in Tab. 2. 50–60% of all items can be transfered in one or two IP packets. In those transfers, TCP – when starting with a window size of one packet – operates on a purely serialized (“handshake”) basis. Only 10–15 % of all items consist of more than 10 packets, allowing the TCP flow control to reach a steady state.

Packets	1	2	3	4	5–10	11–100	101–1000	1001–10000	>10000
Trace A	34±2.2	24.7±2	10.8±0.9	6.8±1.5	14.1±1.9	8.2±1.8	0.37±0.2	0.1±0.1	0.003±0.003
Trace B	32.1±0.9	18.2±0.6	10.4±0.3	6.5 ± 0.2	18.2±1	14.1±0.7	0.51±0.1	0.03±0.01	0.001±0.008

Tab. 2: Distribution of the number of downstream packets received in response to a single GET request. Values are given in percent together with 95 % confidence intervals.

4.3 Fun Factors

Example fun factor results are derived for the two traces using the target values given in Tab. 3. The cumulative observed pure GET waiting delay in an activity is used for T_D and the average loading speed during loading phases of an activity is used for R_L. The delay targets have been deliberately chosen to be rather low compared to observed user *tolerance* values of around 8 s for the total loading time of a page [9] – after all, they define a threshold for “as much fun as possible” rather than a pain threshold, and there must be some time left for the actual loading of elements as well. Following the idea of [11], the rate targets have been set to values specific to the respective access system. The rate related portion of fun is assumed to be 100 % when in a modem or ISDN scenario the received rate is more than 50 kbit/s or when in an ADSL scenario the received rate is more than 500 kbit/s, taking into account the different user expectations associated with different access line speeds.

Trace	d_t	P$\{T_D < d_t\}$	r_t	P$\{R_L > r_t\}$
A	0.5s	62±2.5%	500kbit/s	32±3%
B	0.5s	61±1.8%	50kbit/s	35±3%

Tab. 3: Target values d_t and r_t for T_D and R_L for the evaluation and probabilities to meet the uncorrelated target values.

	Φ_R	Φ_1	Φ_2	Φ_3	Φ_4
Trace A	0.47	0.62	0.40	0.64	0.59
Trace B	0.61	0.61	0.43	0.63	0.57

Tab. 4: Mean fun factors from evaluation of Traces A and B according to (1)–(5) using d_t and r_t from Tab. 3. For all values, 95 % confidence intervals are between 0.02 and 0.04.

The resulting fun factor distributions have been plotted in Fig. 10 and their mean values summarized in Tab. 4. Φ_2 is a lower bound for the other definitions. Even under this very strict evaluation, 32±3% (17±2%) of the activity periods reached $Phi_2 = 1$ in Trace A (Trace B). The same percentage is found for Φ_1, which is a direct consequence of both definitions (see also Fig. 2). On the other hand, Φ_1 allows for a high degree of linear compensation

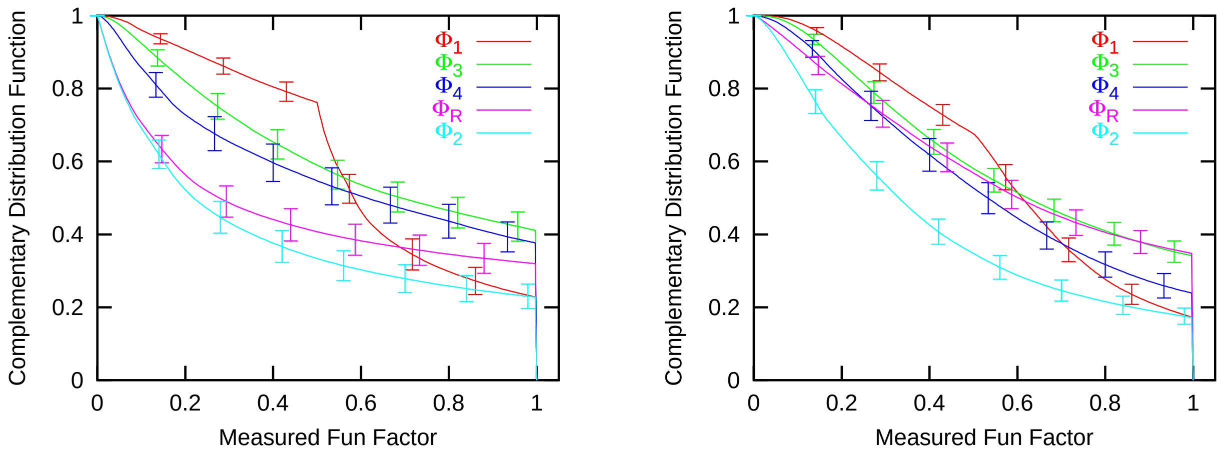

Fig. 10: Complementary distribution functions of measured Fun Factors according to (1)–(5) using target delay and rate values from Tab. 3. 95 % confidence intervals have been added at every 20th data point. Left: Trace A, right: Trace B.

between delay and rate problems, leading to lower probabilities for small values of Φ_1. The way of compensation between rate and delay in Φ_3 and Φ_4 is responsible for the higher values reached. This compensation is slightly overdone in the simple definition of Φ_3, as an exceptionally low delay value can compensate for exceptionally low rate values and vice versa, cases that would nevertheless be perceived as undesirable by the user.

The definition of Φ_4 makes this fun factor a suitable measure for perceived fun as it gives rate and delay the right relative relevance depending on the size of the downloads. The results also show that considering rate and delay in the context of Φ_4 yields more positive results with Trace A than considering the rate alone (Φ_R), as in some cases a relatively low rate may be of low importance if the downloaded items are so small that the transfer delay is still short and if at the same the waiting time until transfer is short enough. A fun factor of $\Phi_4 = 1$ was reached by 38±4% of all activity phases in Trace A in contrast to 32±4% for $\Phi_R = 1$. On the other hand, the activities captured in Trace B had a more prevalent delay problem, leading to a lower portion of activities with $\Phi_4 = 1$ (24±3%) than with $\Phi_R = 1$ (35±3%).

4.4 Further Issues

The evaluations presented here considered only the delay portion due to browsers waiting for the answer to a GET request in already open TCP connections, as it is impossible to deduce the real waiting time from packet traces due to intelligent browsers opening TCP connections before they are actually needed or used [5].

If fun factors are to be used to characterize what performance is obtained from network and servers in relation to what could be expected, effects like the TCP slow start limiting the rate of short multi-packet transfers need to be taken into account, leading to definitions including the number of packets in the rate to be expected. (See the discussion around Tab. 2.) The definition of Φ_4 however deals with this startup issue in a relatively robust way as both the very high observed loading speed of one-packet transfers and the lower startup speed of transfers consisting of a few packets are mostly considered in the form of a delay contribution, which is comparable to the amount of delay experienced in the waiting phase before a download.

Another open issue is how to consider parallel applications in performance measures. In this case, the users (or the operating systems or configurations) are responsible for a degradation of the performance perceived with a foreground application, by consuming network bandwidth, computing power or simply input/output performance of the user’s computer.

5 Conclusions

Fun factors have been presented as a means of describing both delay and rate performance of elastic applications on a scale of $[0, 1]$ easily understandable for users. Together with new statistical evaluations of Web traffic traces on a per-click level including persistent and parallel connections, example evaluations of observed fun factor values have been presented for two extensive client-side traffic traces.

The complex processing required to deduce such characteristics from passive traffic traces and the impossibility to relate some of the delay parts to user perception due to intelligent browser behaviour makes it difficult to apply this kind of measure in a network operator scenario. However, applications like Web browsers could easily include a measure like the fun factor Φ_4 defined in (5) to give users objective and simple to understand feedback regarding network and server performance.

Acknowledgment

The author would like to thank his co-workers at Siemens and his project partners at the Universities of Stuttgart and Würzburg for the fruitful co-operation within the Internet traffic characterization project. Special thanks go to the people at Deutsche Telekom, University of Münster and Bürgernetz Fünfseenland who allowed us to record the traffic traces used in this paper.

References

[1] P. Barford and M.E. Crovella. A performance evaluation of hyper text transfer protocols. In *Proc. ACM Sigmetrics*, pages 188–197, Atlanta, GA, USA, 1999.

[2] P. Barford and M.E. Crovella. Measuring web performance in the wide area. *ACM Perf. Eval. Review*, 1999.

[3] E. Cohen and H. Kaplan. Prefetching the means for document transfer: A new approach for reducing web latency. In *Proc. IEEE Infocom*, Tel Aviv, Israel, 2000.

[4] A. Habib and M. Abrams. Analysis of sources of latency in downloading web pages. In *Proc. Webnet*, San Antonio, USA, 2000.

[5] J. Charzinski. Web performance in practice – why we are waiting. *AEÜ International Journal of Electronics and Communications*, 55, 2001.

[6] J. Charzinski. HTTP/TCP connection and flow characteristics. *Performance Evaluation*, 42:149–162, 2000.

[7] B. Krishnamurthy and C.E. Wills. Analyzing factors that influence end-to-end Web performance. *Comp. Netw.*, 33:17–32, 2000.

[8] C. Huitema and S. Weerahandi. Internet measurements: the rising tide and the DNS snag. In *Proc. ITC Specialist Seminar on IP Traffic*, page Paper 3, Monterey, CA, USA, 2000.

[9] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating user-perceived quality into Web server design. *Comp. Netw.*, 33:1–16, 2000.

[10] M.E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Trans. Networking*, 5:835–846, 1997.

[11] J. Charzinski. Fun factor dimensioning for elastic traffic. In *Proc. ITC Specialist Seminar on IP Traffic*, page Paper 11, Monterey, CA, USA, 2000.

[12] J. Charzinski. Fun factor characterization of user perceived quality of service for elastic Internet traffic. In *Proc. KiVS*, Hamburg, Germany, 2001.

[13] S. McCanne, C. Leres, and V. Jacobson. tcpdump. LBNL Network Research Group, ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[14] B. Mah. An empirical model of HTTP network traffic. In *Proc. IEEE Infocom*, Kobe, Japan, 1997.

[15] H.-K. Choi and J.O. Limb. A behavioral model of Web traffic. In *Proc. ICNP*, Toronto, Canada, 1999.

[16] D.P. Heyman, T.V. Lakshman, and A.L. Neidhardt. A new method for analysing feedback-based protocols with applications to engineering Web traffic over the Internet. *ACM Perf. Eval. Review*, 25:24–38, 1997.