INTERNATIONAL JOURNAL OF SATELLITE COMMUNICATIONS AND NETWORKING
Int. J. Satell. Commun. Network. 2015; **00**:1–14
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/sat

Using SPDY to Improve Web 2.0 over Satellite Links

A. Cardaci[1], L. Caviglione[2*], E. Ferro[1], A. Gotta[1]

[1] *Institute of Information Science and Technologies (ISTI), National Research Council (CNR), 56124, Pisa, Italy*
[2] *Institute of Intelligent System for Automation (ISSIA), National Research Council (CNR), Via de Marini 6, 16149, Genoa, Italy*

SUMMARY

During the last decade, the Web has grown in terms of complexity, while the evolution of the HTTP has not experienced the same trend. Even if HTTP 1.1 adds improvements like persistent connections and request pipelining, they are not decisive, especially in modern mixed wireless/wired networks, often including satellites. The latter play a key role for accessing the Internet everywhere and they are one of the preferred methods to provide connectivity in rural areas or for disaster relief operations. However, they suffer of high latency and packet losses, which degrade the browsing experience. Consequently, the investigation of protocols mitigating the limitations of HTTP, also in challenging scenarios, is crucial both for the Industry and the Academia. In this perspective, SPDY, which is a protocol optimized for the access to Web 2.0 contents over fixed and mobile devices, could be suitable also for satellite links. Therefore, this paper evaluates its performance when used both in real and emulated satellite scenarios. Results indicate the effectiveness of SPDY if compared to HTTP, but at the price of a more fragile behavior when in the presence of errors. Besides, SPDY can also reduce the transport overhead experienced by middleboxes typically deployed by service providers using satellite links.

Received . . .

KEY WORDS: SPDY, Web 2.0, HTTP/HTTP 1.1, Satellite Networking, Performance Enhancing Proxy.

1. INTRODUCTION

In less than a decade, the World Wide Web completely changed its shape. Nowadays, it is highly dynamic, also merged with services encouraging social interaction and the sharing of multimedia materials. Besides, it is also used to remotely access full-featured applications, as it happens in the Software-as-a-Service paradigm [1]. Such an evolution, commonly defined with the Web 2.0 hyponym, is still built on the original architecture of pages composed by two types of objects: the main object containing the HTML and a variable number of linked inline objects. This imposed the need of supporting the more interactive and content-rich nature of Web 2.0, which culminated into an explosion of inline objects, both in size and number [2]. As a consequence, the original version of HTTP fails to handle Web contents characterized by an increased complexity, hence its update is mandatory.

To this aim, HTTP 1.1 introduces improvements like persistent connections and requests' pipelining, but they are not decisive, still resulting in poor performances in terms of Page Loading Time (PLT) for many modern Web destinations [3]. Even if many workarounds in the management

*Correspondence to: Institute of Intelligent System for Automation (ISSIA), Via de Marini 6, Genoa, Italy, I-16149, email: luca.caviglione@ge.issia.cnr.it

of objects and engineering of pages have been proposed, the root of the problem still lies within the HTTP architecture.

Another important aspect influencing the performance of HTTP and Web 2.0 concerns the intrinsic mobile nature of the Internet. In fact, modern network architectures are highly heterogeneous and they mix wired trunks with different types of wireless accesses, such as IEEE 802.11, Long Term Evolution (LTE) and satellites. The latter are still the preferred tool to provide connectivity for Public Protection and Disaster Relief (PPDR) purposes, to bring the Internet in rural areas, or to reduce the digital divide in developing Countries [4], [5] and [6]. Unfortunately, they are often characterized by high delays, (e.g., $\sim$260 ms one-way, for the case of geostationary satellites) or severe packet losses. As a consequence, sophisticated Web 2.0 services, such as websites embedding plug-ins for social interactions or mash-up contents, might be not easily accessed from satellite channels, at least with a proper Quality of Experience (QoE) [7]. Additionally, improvements introduced by HTTP 1.1, such as the more flexible pipelining architecture, could decrease the performances since they inflate the overall Round Trip Time (RTT) [8].

Thus, new protocols are needed to enhance the behavior of Web 2.0 contents when accessed over satellite channels. For instance, a relevant amount of work has been done on the TCP, see, e.g., [9] and references therein. For the case of Web 2.0, one of the most interesting approaches has been proposed by Google and it is named SPDY [10]. Put briefly, it resembles an evolution of the HTTP and it has been engineered to counteract issues and performance degradations typical of links used by mobile devices, such as cellular radio and IEEE 802.11. Besides, SPDY also offers some techniques to better handle Web 2.0 contents, eventually relieving website developers from implementing ad-hoc solutions. Its preliminary assessment over wired/wireless links shows improvements in the range of $27 - 60\%$ [11] [12], while more recent investigations over cellular radio underline a reduced performance, mainly due to the complex cross-layer nature of the carrier [13]. However, in simpler settings, some of its features (e.g., the native support for header compression) still lead to relevant improvements [14].

In this perspective, SPDY would be also able to improve the access to Web 2.0 via satellite channels. In fact, our past works considering mixed WLAN/satellite accesses through emulated settings indicate non-negligible performance gains [16] [17] and [18]. Additionally, SPDY is a very mature technology, thus offering a stable benchmark of the fast evolving and still uncertain HTTP 2.0 - Web 2.0 panorama.

Therefore, this paper investigates SPDY to access Web 2.0 contents via a production quality Internet Service Provider (ISP) offering connectivity through a GEO satellite. To take into account wider sets of use cases, we also emulate some behaviors, such as longer delays and severe packet losses. Nevertheless, as highlighted in [13], realistic deployments could have some pitfalls not revealed by simulated/emulated experimental campaigns. Thus, trials in ISP environments are mandatory to have a correct assessment of the protocol.

At the best of the authors' knowledge, [19] is the only prior work considering the performance of SPDY when used on a real scenario. However, it mainly aims at providing a general analysis of Web-related protocols used over satellites, as well as the impact of bandwidth on demand schemes and specific issues of the HTTP. Instead, in this paper we solely concentrate on SPDY when used on a real satellite ISP, and its main contributions are: *i)* to showcase the SPDY protocol as a tool to mitigate the performance degradations in place of network-centric mechanisms, such as middleboxes; *ii*) to evaluate the impact of its reduced transport complexity over Performance Enhancing Proxies (PEPs); *iii)* to discuss the development of a set of reusable tools to conduct measurements campaigns; *iv)* to provide a performance evaluation of SPDY over a real satellite ISP.

The remainder of the paper is structured as follows: Section 2 introduces the most popular Web 2.0 optimizations, as well as the SPDY protocol. Section 3 showcases the testbed, while Section 4 discusses the methodology used to perform tests. Section 5 presents the experimental results and, lastly, Section 6 concludes the paper.

DOI: 10.1002/sat

2. WEB 2.0 OPTIMIZATIONS AND SPDY

As said, HTTP 1.1 has been introduced to optimize the performance of Web 2.0, but it still requires additional design efforts on contents to achieve satisfying results. On the contrary, SPDY has been engineered to bring greater benefits without altering the architecture of the Web, thus becoming adopted in many large sites (e.g., Twitter and Facebook).

In this section, we quickly review the most popular performance enhancements for Web 2.0 together with their limitations. Then, we showcase SPDY as a possible solution to group optimizations within a single entity. In the following, except when doubts arise, we refer to HTTP 1.1 as HTTP.

2.1. Pipelining, Resource Inlining and CSS Image Sprites

One of the most limiting aspects of HTTP, especially when used over high latency channels, was its inability to handle more than one request per RTT. Such a bottleneck has been removed in HTTP 1.1 by enabling requests to be pipelined (this practice is often defined as "pipelining"). Unfortunately, since the flow of requests/responses must be ordered, Head of Line (HOL) blocking issues could arise. In more details, HOL happens when an object blocks the whole queue, potentially postponing the rendering of the page. Nevertheless, pipelining also has the following additional fragilities: *i)* `POST` methods cannot be easily parallelized, since proper locks are needed to avoid hazards caused by a `GET` needing the response of a prior `POST` [20], and *ii)* a client, being optional, is unable to know a-priori whether a server offers such a feature, thus requiring additional RTTs for the discovery phase.

When both endpoints support pipelining, the reference HTTP implementation imposes a maximum of two connections per domain [20], which could be a too tight requirement for complex contents. Hence, about the totality of browsers violate the protocol specification and use six concurrent connections (increased to eight for the case of Internet Explorer). To have a more aggressive retrieval of data (e.g., images, scripts, and style sheets) and to bypass limitations of the standard protocol specification, Web developers also distribute inline objects over multiple servers.

To increase the throughput in terms of resources per connection, it is common to embed an object within another one, e.g., injecting client-side scripts directly in the HTML. Such a mechanism is defined as resource inlining, and it is very effective if objects are smaller than the size of the HTTP header. As a drawback, this badly impacts over web caches, as a change in the embedded object invalidates also the parent.

A similar approach is the Cascading Style Sheets (CSS) image sprites, where several images are merged into a single larger one. Then, rules defined within the CSS are used to split the content back to the original form. This method leads to a poor maintainability of sources, therefore it is seldom utilized in high quality websites.

2.2. Compression and tuning of TCP

To better take advantage of the available bandwidth, HTTP supports compression of objects and headers through the use of content type negotiation, i.e., a client declares its ability to receive compressed data via the `Accept-Encoding:gzip` header (even if references [21] and [22] show that 66% of sites only actually send compressed responses).

As regards possible cross-layer optimizations, the efficiency of HTTP can be further expanded by properly tuning some parameters of the transport. A typical tweak is adjusting the Initial Congestion Window (ICW) of the TCP as to avoid performance degradations over short-lived connections. Especially, to mitigate the impact of large RTTs, setting the `ICW` $\sim$15 KB nowadays is a quite common practice [23].

2.3. SPDY

Compared to HTTP, SPDY uses a more rational design. In more details, enhancements are enclosed in the protocol, rather than scattered over multiple entities. Also, they do not require content

DOI: 10.1002/sat

developers to tweak the architecture of the page or make adjustments to inline objects. To prevent complexity, as well as to use the available bandwidth in a more effective manner, SPDY relies upon a single-connection architecture, which is based on the following considerations: *i)* HTTP-related transport flows are almost bursty and short-lived, thus causing major overheads in terms of additional RTTs to retrieve a page, and *ii)* avoiding the continuous creation of new connections ensures the Congestion Window (CW) of the TCP to grow faster, also maintaining its "state".

An important design choice of SPDY is that it must guarantee the backward compatibility with the HTTP semantic. Consequently, it does not aim at replacing the HTTP; rather, it provides a multiplexing and priority service to be funneled over a single TCP connection, called `session`. Within a session, a sequence number, defined as `streamid`, identifies each flow carrying encapsulated protocol messages. Finally, to solve the HOL blocking issues and to avoid the need of techniques like CSS inlining, resources are multiplexed. In essence, the four major features of SPDY are:

- *prioritized requests*: each stream has a priority assigned, allowing the `User-Agent` acting in the browser to retrieve objects according to relevance criteria, e.g., to make a page readable as quick as possible, even if incomplete;
- *compressed headers*: SPDY only enforces header compression, while offering the payload processing as optional, e.g., to prevent the duplication of such a procedure;
- *secure sockets*: to provide authentication and encryption, SPDY solely uses Transport Layer Security (TLS). Also, to save additional RTTs, as well as achieving independence from the application layer, SPDY endpoints must be compliant with Next Protocol Negotiation (NPN);
- *server pushed streams*: since the server knows in advance objects needed to complete a page, SPDY can send them "proactively". In this manner, it can populate the browser's cache and mitigate latencies due to additional requests.

We underline that similarly to HTTP, also the behavior of SPDY highly depends on the underlying transport layer. Therefore, to improve its performance, reference [24] suggests an ICW at least ten times the Maximum Segment Size (MSS), which is $\sim$15 KB. Luckily, ICW $=$ 10 is the default choice for the most popular Operating Systems (e.g., since kernel 2.6.38 Linux adopts such a value). Nevertheless, the single-connection architecture of SPDY would benefit from an ICW constantly growing without suddenly shrinking. Then, to avoid connection resets triggering the slow start phase [25], it is a common practice to set the `tcp_slow_start_after_idle` kernel/stack parameter to 0.

To further improve the bandwidth usage, SPDY offers a `settings` session-wide message to negotiate parameters between endpoints. For instance, a typical conversation enables the client to communicate the size of the ICW to the remote server.

3. DEVELOPMENT OF THE TESTBED

To have an effective comparison between HTTP and SPDY, we want to retrieve real Web destinations via a production quality satellite ISP. To this aim, we had to face the following challenges: *i)* the number of SPDY-enabled services is still modest and mainly dealing with online social networks sites; *ii)* the extreme degree of personalization/mutability of Web contents introduces high variability, which can reduce the accuracy of the overall measurements; *iii)* to have a sufficient statistical relevance, we need to properly automate repeated data collection and its organization; *iv)* to precisely understand whether SPDY could be a replacement for HTTP over high delay links, we need to alter some structural parameters of the ISP.

To solve requirements *i)* - *iv)* we developed an instrumented client and an SPDY proxy using open source components. Such tools can be also used to switch to an emulated platform if needed, for instance to add delays and errors. Figure 1 depicts the overall testbed deployed over the satellite ISP.

In more details, the client has been built on top of Google Chrome, which natively supports both HTTP and SPDY, and offers a comprehensive set of debugging and scripting features. To store traces

DOI: 10.1002/sat

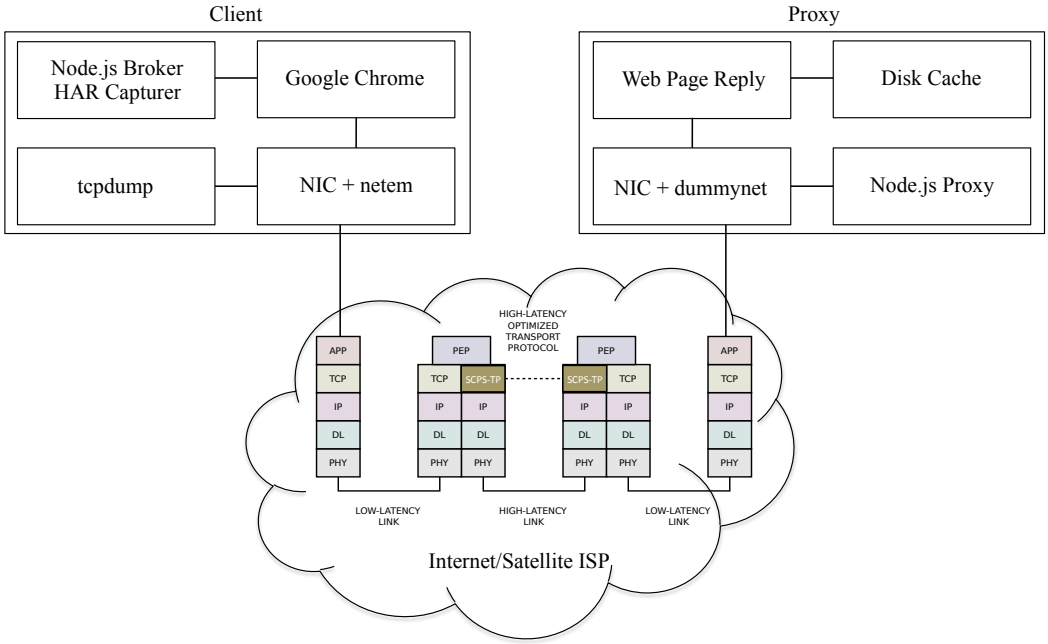

Figure 1. The testbed/protocol architecture used in our trials.

of pages, we used the HTTP Archive (HAR) file format, i.e., a Javascript Object Notation (JSON) document collecting a variety of statistics, such as the page size and timestamps for each inline object. Repeated trials have been automatized via proper shell scripts and a `Node.js` module. To collect data for packet-level analysis, the related traffic has been captured with the tcpdump network sniffer, and stored in different trace files.

As regards the proxy, it has been engineered to cope with the high degree of personalization and mutability of Web 2.0 contents. In fact, due to user interactions such as comments/posts displayed in a real-time manner and changing advertisements, the same page can consistently vary from two adjacent fetches, even performed very close in time. Therefore, retrieving Web pages directly from the server would lead to incoherent measurements. To remove any possible source of non-determinism, all the contents used for the performance assessment are cached via Web Page Replay (WPR) [26]. In essence, it is a tool for making a snapshot of a page, and then it acts both as a Web and a DNS server in the replaying phase. We underline that using a proxy also enables to deliver via SPDY sites hosted on servers that do not natively support this protocol.

To have a thorough understanding of the interaction of SPDY with protocol accelerators deployed within the ISP network, we also need to modify the RTT, to introduce an arbitrary packet loss, and to bypass PEPs. Even if we had a dedicated access, we were not able to alter the set-up without impacting over the normal functioning of the ISP. Therefore, for the round of tests without PEPs, we used netem and dummynet properly attached and configured on the Network Interface Controller (NIC) of the client and proxy, respectively.

The satellite ISP exploits the iDirect platform. Access terminals are iDirect Evolution X3 Satellite Routers offering HTTP and TCP acceleration combined in a unique device. TCP is enhanced via standard connection splitting mechanism, while HTTP relies upon the “Split HTTP” proxy architecture, as depicted in Figure 2.

In more details, the proxy is composed of two entities, namely the Remote Proxy (RP) and the Local Proxy (LP), and they are placed at the borders of the satellite channel, i.e., to isolate the high latency link. We point out that RP and LP should not be confused with the SPDY proxy, which is placed “outside” the ISP and only used to make our tests coherent and comprehensive. While the RP is a classical HTTP proxy, the LP has an enriched set of functionalities, e.g., from the processing of Javascript code and the management of dynamic contents, to a simpler one, like the retrieval of

DOI: 10.1002/sat

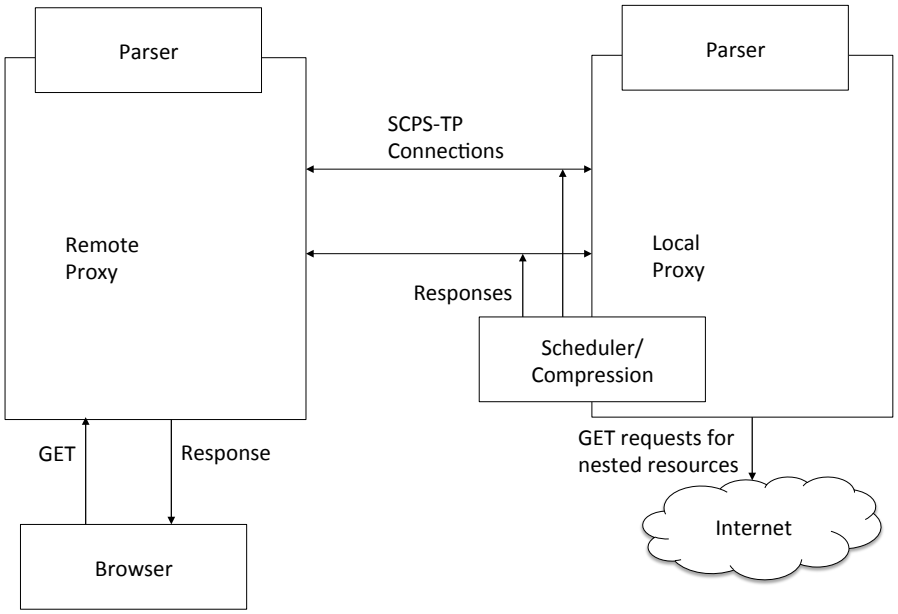

Figure 2. System components of the HTTP proxy based on HTTP/TCP splitting present in our testbed.

nested inline objects (but limited to the single-page dependency). In essence, to mitigate the impact of the satellite link, the RP and LP split the HTTP conversations and transfer the Web contents by using proper compression mechanisms and per-object scheduling disciplines. To recap, for each page request the following steps are performed:

1. the browser requests a page by issuing a `GET`;
2. the RP traps the request, extracts the related URL, which is forwarded to the LP;
3. the LP retrieves the main object, as well as the linked inline objects. If any, additional information like cookies is also acquired and locally stored;
4. all data are compressed and delivered towards the original requestor via an ad-hoc protocol. A proper scheduling could be specified and applied to the resulting stream;
5. as soon as the RP receives the main object, it determines the amount of resources "on the fly", and starts to push back data to the browser.

To move data between RP and LP (step 4), the Space Communications Protocol Specification-Transport Protocol (SCPS-TP) [27] [28] is used, which enables to stream multiple TCP connections via a single SCPS-TP one. To achieve further improvements, resources can be delivered in parallel through multiple SCPS-TP channels, or multiplexed into a single SCPS-TP flow. We point out that the latter resembles the approach of SPDY.

From the viewpoint of the HTTP, the architecture of Figure 2 leads to the following improvements: *i)* the three-way handshake of the TCP is performed over the ground portion of the network (i.e., the one with lower RTT values); *ii)* the parser within LP prevents a `GET` to be transmitted over the satellite link; *iii)* the proxying entities allow persistent connections at the transport layer, thus reducing the number of slow-start phases; *iv)* compression reduces the amount of traffic.

The provided satellite channel is a Ku link using the Time Division Multiple Access (TDMA) and having rates of 128 kbps for the upstream and 1 Mbps for the downstream, respectively. The satellite is the Express AM44@11°W bent-pipe GEO, operated by RSCC (Russian Satellite Communication Company). The average RTT is 620 ms, also including the traversal of middleboxes and the wired trunks of the network.

Lastly, to have a fair evaluation, all the best practices in terms of TCP tweaking presented in Section 2.3 have been applied during the entire round of tests.

DOI: 10.1002/sat

Table I. Number of TCP connections per site when using standard HTTP. Rank is performed according to Kbytes per page *on the wire*. Reported values are average over the entire dataset.

Rank	Site name	N of Connections	Kbytes/Page	Requests	N. Domains
1	Wikipedia	17	111.13	18.97	3.00
2	Reddit	41	371.42	51.44	14.72
3	Flickr	14	499.28	17.34	4.24
4	Slashdot	50	712.62	48.76	10.84
5	BBC	88	950.73	85.01	12.00
6	Microsoft	58	1176.25	52.43	9.56
7	Huffington Post	173	1481.80	110.11	32.89

4. MEASUREMENT METHODOLOGY

In order to have an effective assessment of the performances of SPDY, we firstly had to choose an effective synecdoche of the actual Web 2.0 panorama. To this aim, we decided to take as a reference the ranking compiled by Google back in 2010 [22]. In fact, it contains sites having interesting features potentially impacting both over HTTP and SPDY. Especially: many destinations have an average page size of 320 Kbytes; on average, only the $2/3$ of inline objects are actually compressed; at least 80% of pages composing such sites have 10 or more inline objects retrieved from a single host. However, as a consequence of the fast evolution of the Internet, when comparing such works with more recent measurements (see, e.g., references [2], [29] and [30]), we noticed that Web 2.0 could be not effectively described with such values anymore. The list available in [22] still contains the most popular and accessed Web destinations, but needs further investigations to properly select the most representative ones.

Therefore, after a preliminary set of measurements (see [16] and [17] for a detailed discussion), showcased in Table I, we decided to select 7 sites to have a properly mixed number of destinations. Especially, they represent an accurate snapshot of sites designed to use a high-volume of inline objects, also with many interdependences, as it happens in the most complex Web 2.0 services. Moreover, their features are general enough to allow modeling a wider variety of cases, thus making our investigation more effective and general. Summing up, we selected the following set of websites:

- *Reddit* and *Slashdot*: they represent services aggregating people to foster discussion. They are based on a limited amount of large images, but also using a variety of small graphical elements (often defined as thumbnails);
- *Huffington Post* and *BBC*: they mainly provide news, also embedding videos and plugins to share and discuss over online social networks. Consequently, both sites exploit a relevant number of inline objects scattered across different domains, also containing additional software components;
- *Flickr* and *Microsoft*: they have been selected as typical examples of pages crafted for showcasing or advertising contents/products. Particularly, they include very large graphic elements, and plug-ins to implement carousels or multimedia playback;
- *Wikipedia*: it relies on a simple/text-based layout, also reducing to the minimum the number of external dependencies.

In more details, Table I reports the number of transport connections required to retrieve all the objects that compose the selected sites when using standard HTTP. This can be adopted as a metric to quantify the stress in terms of transport layer complexity a PEP or a middlebox has to handle. Such values range from 17 (for the case of Wikipedia) to 173 (for the case of Huffington Post) and allow to consider the content-richness nature of Web 2.0 applications, as well as the need of retrieving a heterogenous set of objects likes Javascript code or additional plug-ins.

DOI: 10.1002/sat

Instead, when using SPDY, the number of transport connections always reduces to 1, which is a direct consequence of the protocol architecture. In fact, all the data is sent via a single flow, and this can be exploited to reduce the overheads experienced by PEPs, as well as to shift all the complexity of an ISP to the border, in this case into a software layer running into end nodes.

The set considered is also characterized by wide variations in terms of page sizes (denoted as Kbytes/Page in Table I). In fact, in the normal daily usage of the Web, users also access simple destinations. Thus, we included sites like Wikipedia and Reddit as to avoid the pitfall of having a biased performance evaluation, i.e., as it can happen for protocols primarily optimized to handle complex sites.

We also considered the "implementation" of a given site by taking into account the number of requests and domains accessed. In this extent, the former indicates sites having a quite relevant overhead in terms of HTTP conversations. The latter offers an idea on how many domains are used to store objects to increase the degree of parallelism of the retrieval phases, as well as the "mashed" nature of Web 2.0 sites.

As regards the trials, they have been performed in three different satellite configurations: the one of Figure 1 with the real ISP (denoted in the following as "PEP" to emphasize the presence of a middblebox), and by using `netem+dummynet` to emulate round trip delays of 520 ms and 720 ms. In this case, bandwidths have been set to 1 Mbit/s and 256 kbit/s, in the forward and return link, respectively (see, e.g., [31] for a discussion on tuning emulated satellite testbeds starting from real measurements). In addition, the SPDY proxy depicted in Figure 1 has been used in all configurations to add different packet losses.

For each configuration, we retrieved all the sites of Table I by using both HTTP and SPDY. Each test has been repeated 20 times, resulting into statistics and timings of about $38,000$ objects. To handle and process such data, we used an SQL database and ad-hoc scripts.

5. EXPERIMENTAL RESULTS

To have a basic characterization of the traffic, we preliminary investigated the overall dataset. As expected, we found a very high amount of TCP conversations mainly due to the content-richness nature of Web 2.0 applications using a composite set of objects, plug-ins or multi providers mash-ups. The only exception is Wikipedia, since it mainly exploits a text-based layout, and does not embed additional services, such as widgets à la Google Maps. From a "complexity" viewpoint, SPDY surely mitigates the number of transport connections traversing the middlebox. Thus, the number of sockets to be handled is smaller, reflecting into less state information to be stored within a PEP or in less overheads in the stack of mobile or limited-capabilities devices.

To better comprehend results, we recall that, when the PEP is deployed, two kinds of acceleration are used. One acts by splitting TCP connections, thus enhancing both SPDY and HTTP. Another one involves the processing of HTTP traffic. However, even if SPDY preserves the HTTP semantic, it encrypts all the data, eventually becominge unrecognized by the PEP. Thus, such an improvement is only applied to HTTP.

5.1. PLT on lossless and lossy links

In this Section we compare the PLT of HTTP and SPDY by showcasing the 95th percentile of the repeated trials averaged over the set of considered sites.

Figure 3 depicts the PLT values collected when using an error-free satellite link. For the case of RTT=520 ms, SPDY grants smaller average times compared to HTTP. But, when RTT=720 ms, such an enhancement vanishes, with HTTP performing the same. Instead, when RTT=620, the HTTP performs slightly better than SPDY, mainly owing to the presence of the PEP deployed by the ISP.

Figure 4 portraits the PLT values when the satellite link introduces a packet loss of 1%. We point out that, even if we performed tests with different losses, only results with 1% are shown, as they are the most relevant, also representing the worst case. In fact, when in the presence of losses $> 1\%$, the browser usually fails to complete about the totality of transfers, thus aborting the rendering of the

DOI: 10.1002/sat

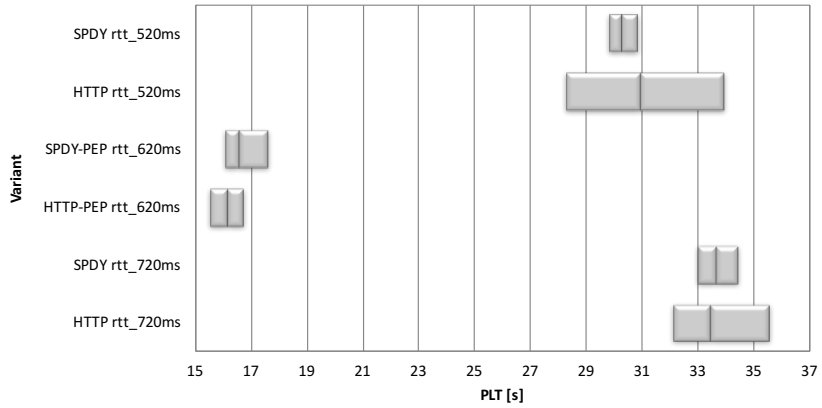

Figure 3. Average PLT on a lossless link.

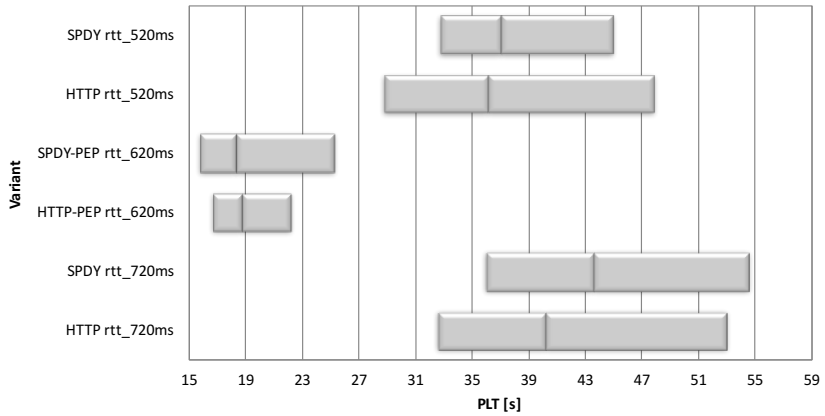

Figure 4. Average PLT on a 1% lossy link.

hypertext and returning an error. When packet losses are present, the PLT experiences an increment of 11% with SPDY and ∼16% with HTTP. On the emulated platform, values are almost doubled, since the lack of PEP increases the PLT to 18% for HTTP, and to 31% for SDPY. Then, when large RTTs and losses are not mitigated by PEP, SPDY seems to be less robust due to its single TCP connection design, eventually causing an increase in the experienced PLTs. In addition, the variation range of PLT is very limited with PEPs and is kept particularly small with SPDY on PEP.

However, as it will be shown later, even if the PLT is an effective and widely used parameter to assess the performances of Web, it does not take into account QoE metrics. For instance, it fails to capture the effective readability of a web page or its level of completeness during the download process.

5.2. Throughput Analysis

A relevant aspect useful to characterize the usage of network resources is given by the analysis of the throughput. Similarly to the PLT, such a behavior does not efficiently capture how "promptly" a page is delivered to users. Instead, it gives some hints on how HTTP and SPDY react against latency and losses introduced by the satellite channel, especially in terms of utilization of the transmission resources available on the link.

In all the trials, both protocols experience higher throughputs when retrieving the Huffington Post, since its content-rich nature enables the TCP to have a longer temporal horizon to increase the transmission window and to exploit the available resources (i.e., to "fill the bandwidth pipe").

DOI: 10.1002/sat

In more details, the parallel connection flavor of HTTP enables to partially compensate for high latencies even when the PEP is not deployed. For the case of SPDY, its single-connection blueprint leads to a less-aggressive behavior in terms of throughput if compared to HTTP. As regards the PEP, its connection splitting nature enables to saturate all the available bandwidth both when used through HTTP and SPDY. When in the presence of errors and high latencies, SPDY performs worse than HTTP in all scenarios, mainly due to its more fragile nature rooted within the exploitation of a single connection. In other words, a burst of lost packets will not distribute over multiple connections, but it concentrates on the single flow causing the congestion control of the TCP to react. Table II presents the average values computed over the entire dataset.

Table II. Throughput for Wikipedia and Huffington Post.

	RTT= 520		RTT= 620		RTT = 720	
ploss=0%	HTTP	SPDY	HTTP	SPDY	HTTP	SPDY
Wikipedia	149.40	135.40	141.17	132.20	117.00	111.80
Huffington Post	217.75	210.40	219.28	216.59	221.20	213.40
ploss=1%	HTTP	SPDY	HTTP	SPDY	HTTP	SPDY
Wikipedia	89.40	123.80	93.67	90.65	97.00	89.40
Huffington Post	193.00	193.25	194.10	137.73	204.00	172.12

5.3. Packet Size Analysis

To better understand the impact of the Web 2.0 paradigm over the satellite link, we want to quantify the improvement of SPDY in terms of the usage of transmission resources. Thus, we conducted an investigation on the average size of the Protocol Data Units (PDUs) generated by each protocol. Specifically, in all the considered scenarios, SPDY exhibits a reduced number of tiny PDUs (i.e., < 80 bytes) if compared against HTTP, despite the presence of the PEP. In more details, $\sim 60\%$ of PDUs generated by SPDY has a size in the range 1280 - 1500 bytes, while for HTTP this reduces to $\sim 28\%$. Such a behavior is of particular importance in the case of satellite, since fewer packets reflect into less time spent in accessing the channel. Therefore, the access to Web 2.0 contents through high-delay links should not experience a loss of performance due to additional rounds of contention at the MAC layer.

5.4. Per-object analysis

As said, the PLT only offers information on the timeframe between the request of a page and its completion (i.e., the last inline object linked against the hypertext is received) and does not allow to quantify the QoE perceived by users. Therefore, we processed the HAR collected for each trial and we investigated timing statistics with a per-object granularity. To this aim, we developed an ad-hoc software, which has been released under the open source license [32]. Figure 5 depicts an example of HAR waterfall diagram containing all the timing information for the objects composing the homepage of Wikipedia.

Specifically, three time statistics have been extracted:

- block: it is the time spent by the browser to gain access to a free socket. This strictly depends on the number of parallel connections supported. For the case of HTTP, it is equal to 6, while for SPDY it is always equal to 1 since it relies upon a unique multiplexed transport flow;
- wait: it is the time the client awaits before receiving a response from the server. In other words, it is the time that the server uses to deliver the beginning of a response header;
- receive: it is the time needed to completely receive an object.

Figures 6, 7 and 8 show the Cumulative Distribution Functions (CDF) of the aforementioned time values.

DOI: 10.1002/sat

Figure 5. HTTP archive (HAR) capturing for Wikipedia: green is the connect time, purple is the wait time, and gray is the receive time.

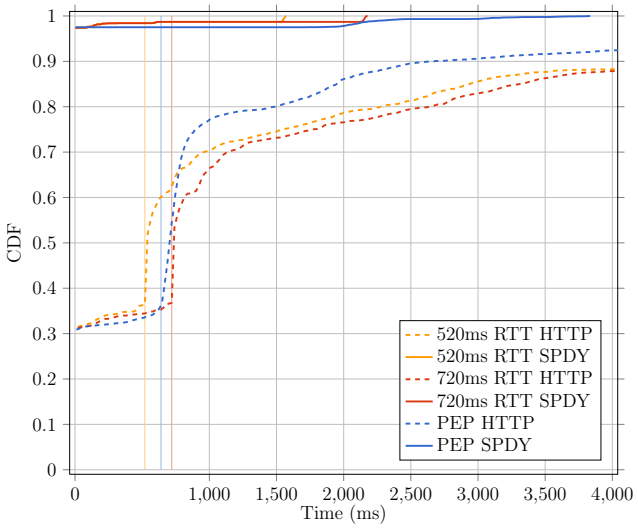

Figure 6. CDF of block timings of the objects composing webpages (different RTT values are marked with vertical lines).

Specifically, the CDFs for the block time depicted in Figure 6 clearly highlights the impact of high RTTs when HTTP is used. For what concerns SPDY, for more than 95% of the trials, the delays due to block conditions are mostly near 0. In more details, its single socket architecture enables to constantly feed the channel, thus avoiding overheads in accessing/creating different transport layer connections.

As regards the wait time showcased in Figure 7, since it represents the timeframe between the transmission of the request and the reception of the response header, it is at least always equal to one RTT plus the service time needed by servers and PEP for processing. In our trials, SPDY does not totally outperform HTTP. The main reason is due to the multiplexing policy used in the adopted

DOI: 10.1002/sat

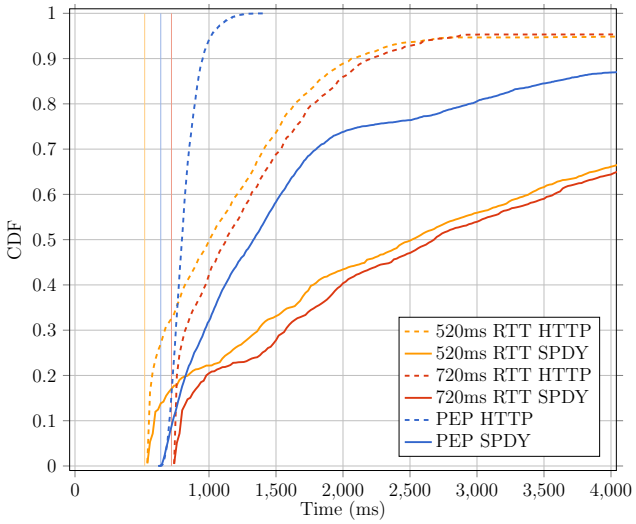

Figure 7. CDF of wait timings of the objects composing webpages (different RTT values are marked with vertical lines).

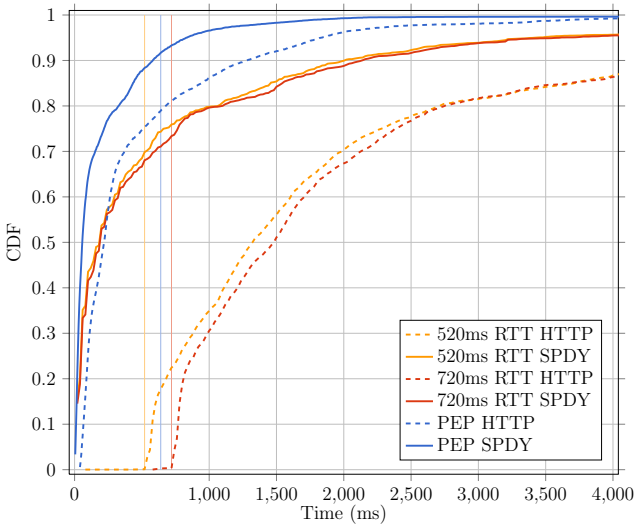

Figure 8. CDF of receive timings of the objects composing webpages (different RTT values are marked with vertical lines).

implementation. In fact, according to [10], the TCP provides a single stream of data on which SPDY multiplexes multiple logical streams, thus clients and servers must intelligently interleave data messages for concurrent sessions. Unfortunately, our set-up does not exploit any predictive or optimized frameworks, hence causing the serialization of inline objects to limit the performance.

The receive time presented in Figure 8 closely relates to how much the application layer takes advantage of the available bandwidth. In all scenarios, we found that both protocols produce bursty traffic, which interferes with flow control algorithms of the TCP. Also in this case, the policy used by SPDY to interleave data coming from different streams can impact on the receive time. With respect to Figure 8, we can notice that the PEP effectively mitigates the impact of high RTTs, when using HTTP. Similarly, for the case of SPDY, the larger congestion window, as well its enhanced transport behavior, allow to have similar benefits.

DOI: 10.1002/sat

5.5. *Discussion of Results*

As discussed, to effectively assess the performance of HTTP and SPDY, the presented metrics should be considered as a whole. In other words, evaluating them separately does not allow to have any idea on the user-experience, which is a critical aspect for the usability of Web 2.0 contents via satellite links. The first consideration concerns the PLT. Our investigation revealed that the lower values achieved by HTTP are partially voided by the timing statistics characterizing the reception of inline objects composing the page. In fact, SPDY reduces waiting times, thus making the reception of a complete page more "responsive". Such an improvement quickly vanishes when a PEP is deployed, hence making the behaviors of the two protocols very similar. However, when using SPDY, the HTTP acceleration deployed by the ISP is not applied, since the related traffic is encrypted. Therefore, SPDY can be used in place of such a module, thus making middleboxes less complex and expensive.

A similar consideration can be done for the throughput. Numerical results indicate that the main bottleneck is still the TCP, which can be mitigated by the PEP and the aggressive parallel connection flavor of HTTP. In this perspective, SPDY cannot saturate the available bandwidth as the HTTP (even if parameters ruling congestion control algorithms have been finely tuned). Yet, users perceive similar browsing QoE for RTT=520 ms and RTT=620 ms and, for the case of RTT=720 ms, SPDY outperforms HTTP. This is mainly due to the reduced overheads in terms of HTTP headers and TCP connection set-up/tear-down procedures. Therefore, a greater throughput of data over the link does not imply a greater throughput in terms of inline objects.

Lastly, in all scenarios, the errors severely influence the behaviors of SPDY, which can be partially mitigated by the PEP or by using ad-hoc countermeasures deployed in the lower layers of the stack [33].

6. CONCLUSIONS AND FUTURE WORK

In this paper we investigated SPDY as an alternative to HTTP for accessing content-rich Web 2.0 destinations via satellite links. To this aim, we developed an ad-hoc client, a proxy and a set of tools to perform trials both in real and emulated satellite environments. Results revealed that in some cases SPDY can be used in place of the HTTP acceleration part of the PEP, thus reducing both the complexity and costs for the ISP. However, when in the presence of errors, its single-stream nature introduces fragilities, thus proper countermeasures should be used in the lower layers of the protocol stack (e.g., forward error control or coding scheme).

Future work aims at improving the scheduling policy of SPDY, especially to find an optimal mapping between a priority class and the HTML object(s), in order to minimize latencies in the rendering of a Web page or, at least, for the most significant part of it.

ACKNOWLEDGEMENT

This work has been partially funded by the European Space Agency (ESA) within the framework of the Satellite Network of Experts (SatNex-III), CoO3, Task3, ESA Contract no. 23089/10/NL/CLP. Thanks are due to Link Telecomunicazioni S.r.l. (http://www.linksrl.tv/) for providing the satellite experimental platform, and in particular to Ing. Ilaria Agostini, Ing. Giuseppe Spanò and Mr. Roberto Parodi, for the technical support and the preparation of the satellite test bed.

REFERENCES

1. I. Hsu, "Multilayer Context Cloud Framework for Mobile Web 2.0: a Proposed Infrastructure", Int. Journal of Communication Systems, Vol. 26, No. 5, pp. 610 - 625, 2013.
2. L. Caviglione, "Extending HTTP Models to Web 2.0 Applications: The Case of Social Networks", in Proc. of the 4th IEEE Int. Conference on Utility and Cloud Computing (UCC), pp. 361 – 365, Dec. 2011.
3. S. Souders, "High-performance Web Sites", Communications of the ACM, vol. 51, no. 12, pp. 36 – 41, 2008.
4. L. Caviglione,"Introducing Emergent Technologies in Tactical and Disaster Recovery Networks", Int. Journal of Communication Systems, Vol. 19, No. 9, pp. 1045 - 1062, 2006.

5. G. Bartoli, R. Fantacci, F. Gei, D. Marabissi, L. Micciullo, "A novel Emergency Management Platform for Smart Public Safety", Int. Journal of Communication Systems, Vol. 28, No. 5, pp 928 - 943, 2015.
6. G. Fairhurst, L. Caviglione, B. Collini-Nocker, "FIRST: Future Internet - a role for Satellite Technology", in Proc. of the IEEE Int. Workshop on Satellite and Space Communications, (IWSSC 2008), Siena, Italy, pp. 160 – 164, Oct. 2008.
7. L. Caviglione, "Can satellites face trends? The case of Web 2.0", in Proc. of the 2009 Int. Workshop on Satellite and Space Communications (IWSSC 2009), Siena, Italy, pp. 446 – 450, Sept. 2009.
8. R. Chakravorty, A. Clark, I. Pratt, "Optimizing Web Delivery over Wireless Links: Design, Implementation, and Experiences", IEEE Journal on Selected Areas in Communications, vol. 23, no. 2, pp. 402 – 416.
9. E. Rendon-Morales, J. Mata Diaz, J. Alins, J. L. Munoz, O. Esparza, "Performance Evaluation of Selected Transmission Control Protocol Variants over a Digital Video Broadcasting - Second Generation Broadband Satellite Multimedia System with QoS", Int. Journal of Communication Systems, Vol. 26, No. 12, pp. 1579 - 1598, 2013.
10. M. Belshe, R. Peon, "SPDY Protocol", draft-mbelshe-httpbis-spdy-00, Network Working Group, IETF, 2012.
11. M. Welsh, B. Greenstein, M. Piatek, "SPDY Performance on Mobile Networks", available online https://developers.google.com/speed/articles/spdy-for-mobile, 2012. [Last Accessed Feb. 2014].
12. X. S. Wang, A. Balasubramanian, A. Krishnamurthy, D. Wetherall, "Demystifying Page Load Performance with WProf", In Proc. of the 10th USENIX conference on Networked Systems Design and Implementation (NSDI13), N. Feamster, J. Mogul (Eds.), USENIX Association, Berkeley, CA, USA, 2013, pp. 473 – 486.
13. J. Erman, V. Gopalakrishnan, R. Jana, K. Ramakrishnan, "Towards a SPDY'ier Mobile Web", in Proc. of the 9th ACM Conference on Emerging Networking Experiments and Technologies, ACM, pp. 303 – 314, Dec. 2013.
14. H. Kim, G. Yi, H. Lim, J. Lee, B. Bae, S. Lee, "Performance Analysis of SPDY Protocol in Wired and Mobile Networks", In Ubiquitous Information Technologies and Applications, pp. 199 – 206, Springer Berlin Heidelberg, Jan. 2014.
15. P. Davern, N. Nashid, A. Zahran, C. J. Sreenan, "HTTP Acceleration over High Latency Links", in Proc. of the 4th IFIP Int. Conference on New Technologies Mobility and Security (NTMS), pp. 1 – 5, Feb., 2011.
16. A. Cardaci, L. Caviglione, A. Gotta, N. Tonellotto, "Performance Evaluation of SPDY over high Latency Satellite Channels", in R. Dhaou et al. (Eds.), PSATS 2013, LNICST 123, pp. 123 – 134, 2013, Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, 2013.
17. A. Cardaci, N. Celandroni, E. Ferro, A. Gotta, F. Davoli, L. Caviglione, "SPDY - a new Paradigm in Web Technologies: Performance Evaluation on a Satellite Link", in Proc. of the 19th Ka and Broadband Communications Navigation and Earth Observation Conference, Florence, Italy, FGM Events LLC 239, Oct. 2013.
18. L. Caviglione, A. Gotta, "Characterizing SPDY over High Latency Satellite Channels", EAI Endorsed Transactions on Mobile Communications and Applications, Vol. 14, No. 5, pp. 1 - 10, Dec. 2014.
19. L. Caviglione, N. Celandroni, M. Collina, H. Cruickshank, G. Fairhurst, E. Ferro, A. Gotta, M. Luglio, C. Roseti, A. A. Salam, R. Secchi, Z. Sun, A. V. Coralli, "A deep Analysis on Future Web Technologies and Protocols over Broadband GEO Satellite Networks", Int. Journal of Satellite Communications and Networking, Vol. 33, No. 5, pp. 451-472, Sept./Oct. 2015.
20. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", IETF, Network Working Group, RFC 2616, 1999.
21. F. Schneider, S. Agarwal, T. Alpcan, A. Feldmann, "The new Web: Characterizing AJAX Traffic", in Passive and Active Network Measurement, Springer Berlin, Heidelberg, pp. 31 - 40, 2008.
22. S. Ramachandran, "Web Metrics: Size and number of Resources", online: https://developers.google.com/speed/articles/web-metrics. [Last Accessed: April 2015].
23. O. Spatscheck, J. S. Hansen, J. H. Hartman, L. L. Peterson, "Optimizing TCP Forwarder Performance", IEEE/ACM Transactions on Networking, vol. 8, no. 2, pp. 146 – 157, Apr. 2000.
24. N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, N. Sutin, "An Argument for Increasing TCP's Initial Congestion Window", ACM SIGCOMM Computer Communications Review, vol. 40, pp. 27– 33, 2010.
25. M. Handley, J. Padhye, S. Floyd, "TCP Congestion Window Validation", IETF, Network Working Group RFC 2861, 2010.
26. Web Page Reply, available online: https://github.com/chromium/web-page-replay.
27. Consultative Committee for Space Data Systems (CCSDS), "Space Communications Protocol Specification-Transport Protocol (SCPS-TP)", Recommendation for Space Data Systems Standards, CCSDS 714.0-B-1, no. 1, Blue Book, 1999.
28. R. C. Durst, G. J. Miller, E. J. Travis, "TCP Extensions for Space Communications", ACM/Kluwer Wireless Networks Journal (WINET), vol. 3, no. 5, pp. 389–403, 1997.
29. W. Li, A. W. Moore, M. Canini, "Classifying HTTP Traffic in the new age", in Proc. of ACM SIGCOMM, pp. 17 – 22, 2008.
30. S. Ihm, V. S. Pai, "Towards Understanding Modern Web Traffic", in Proc. of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, ACM, p. 295 – 312, 2011.
31. A. Gotta, F. Potorti, R. Secchi, "An analysis of TCP startup over an experimental DVB-RCS platform", in Proc. of the 2006 Int. Workshop on Satellite and Space Communications, pp. 176 – 180, 2006.
32. Automatic HAR capturer, available online: https://github.com/cyrus-and/chrome-har-capturer
33. C. I. Kuo, C. K. Shieh, W. S. Hwang, C. H. Ke, "Performance Modeling of FEC-based Unequal Error Protection for H. 264/AVC Video Streaming over Burst-Loss channels", Int. Journal of Communication Systems, pp. 1099 - 1131, 2014.

DOI: 10.1002/sat