PTPv2-Based Network Load Estimation and Its Application to QoE Monitoring for Over-the-Top Services

Pantelis A. Frangoudis*, Yassine Hadjadj-Aoul‡, Adlen Ksentini‡, Gerardo Rubino*,
Thierry Etame§, and Clément Brun§
*INRIA Rennes-Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes, France
‡IRISA-University of Rennes 1, Campus de Beaulieu, 35042 Rennes, France
§ip-label, 90 Boulevard National, 92250 La Garenne-Colombes, France
Email: *firstname.lastname@inria.fr, ‡firstname.lastname@irisa.fr, §{tetame,cbrun}@ip-label.com

***Abstract*—The recently-standardized Precision Time Protocol v2 (PTPv2) aims at achieving sub-microsecond-level synchronization between network clocks. In this work, we make the case for an alternative use of it: Adopting a learning approach, we observe its delay behavior during the protocol message exchange, derive models of its dependence on network load and build a real-time load estimation service. Then, as an application scenario of this service, we turn our attention to the provision of Over-the-Top (OTT) services. In such an environment, and assuming a level of cooperation between the ISP and the OTT provider, we demonstrate how our service can be used for estimating user experience for web applications. To this end, we establish quantitatively the link between network load and user experience using a state-of-the-art web QoE monitoring framework, and show how our PTPv2-based load estimation scheme can be integrated in an OTT service architecture and be utilized for load-aware, QoE-optimized content delivery decisions.**

I. Introduction

Internet traffic is currently dominated by data distributed by Content Delivery Networks (CDNs). Network awareness is critical for optimized delivery, while, on the other hand, CDN request redirection strategies can significantly disrupt ISP network operation due to the sudden traffic shifts they can cause. This calls for substantial cooperation between ISPs and content providers (CPs) for (i) more efficient delivery, from the CDN viewpoint, and (ii) more efficient use and management of network resources, from the viewpoint of the ISPs [1], [2]. This interdependence has further given rise to the emergence of *Telco-CDNs* [3], namely content delivery infrastructures deployed and controlled by ISPs.

In this work, we focus on environments where content is delivered by an over-the-top (OTT) provider, but carried out with a significant level of network-awareness, as a result of the ISP-CP cooperation. This implies the ability to perform informed delivery decisions having internal network information available by the ISP. Such decisions involve proper request redirection strategies, based on network-level information: Ideally, assuming that there are multiple data centers (DC) from which the user can retrieve the same content, the CP can resolve the user's DNS query to the appropriate DC, taking into account network load information on the user-DC paths, aiming at balancing load and improving response times, and thus user experience.

User experience is not straightforward to capture. The most accurate means would involve either having the user directly report it, or have access to her device and perform application-oriented measurements directly at the user end. However, these approaches are either intrusive or require additional software agents on user terminals. Inferring user experience based on observations of network-level quantities which are more easily measured is thus desirable. For an OTT CP, though, the availability of detailed information about the conditions on the network segments controlled by the ISP cannot always be assumed, since the network is outside the CP's control.

Our work contributes towards this end, attempting to answer to the above challenges. We have designed and implemented a network load estimation service which can be deployed by a CP in collaboration with an ISP to offer information about the conditions in a network path without requiring the ISP to expose direct real-time measurements from his network.

The distinctive characteristic of our approach is that we explore the potential of the recently standardized IEEE 1588-2008 synchronization protocol (Precision Time Protocol Version 2) [4] for an alternative purpose: We take advantage of delay measurements carried out as the protocol operates as indicators of network load [5]. We apply a measurement-based learning methodology where we perform observations, learn the delay behavior of the PTPv2 protocol under controlled load conditions, and build models of the dependence of path delay on network load.

These load indications can be translated to Quality-of-

This work was partially supported by the French national FUI project IPChronos and the CELTIC project QuEEN (Project ID: CP8-004).

Pantelis A. Frangoudis' work was carried out in part during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 246016.

Experience (QoE) estimates. To establish the link between network load and QoE, we use a state-of-the-art QoE monitoring tool [6] for web applications in a testbed we have developed.

We finally show how our tools fit in the content delivery process by demonstrating how PTPv2 protocol entities and our load estimation services could be deployed and utilized by an OTT content provider.

The remainder of this paper is structured as follows: In Section II we review the related work, before presenting our PTPv2-based network load estimation methodology and tools in Section III. We study quantitatively the relationship between the estimated network load and user experience in Section IV, present our design for OTT content delivery which utilizes our load estimation service in Section V, and conclude the paper in Section VI.

II. Related work

A. The Precision Time Protocol

We make the case for utilizing the Precision Time Protocol v2 (PTPv2), specified in the IEEE 1588-2008 standard [4] for a purpose originally unintended for. PTPv2 aims to bridge the gap between NTP and expensive timing solutions based on GPS or atomic clocks and achieve sub-microsecond-level synchronization; in this work, we apply it for estimating network load based on the protocol's delay behavior.

PTPv2 typically operates in a master-slave fashion, where slave clocks attempt to achieve phase (time) and frequency synchronization with a master clock which provides a time reference and is assumed to be connected to a high-accuracy time source (e.g., a GPS clock). A master clock periodically transmits timestamped SYNC messages, which are used by slaves to measure the master-to-slave time difference upon reception[1]. This value includes the time offset between the two clocks and the message propagation delay. Note that using SYNC messages alone, a slave can also measure its frequency offset and perform necessary corrections to its local clock frequency to keep syntonized to the master.

To calculate the message propagation delay, the slave sends a DELAY_REQUEST message to the master, to which the master replies with a DELAY_RESPONSE message. A DELAY_RESPONSE includes the timestamp when the respective DELAY_REQUEST was received by the master, which is used by the slave to calculate the slave-to-master time difference. The slave combines this measurement with the master-to-slave time difference to estimate the *mean path delay* (typically as half of the round-trip time, assuming path symmetry), and correct its time offset from the master. The PTPv2 protocol specifies further messages, mainly for management purposes, which are outside the scope of this work.

[1]Due to hardware limitations, it is not always possible to record the exact time a SYNC message was transmitted and add it to the message on the fly. In these cases, this timestamp can be conveyed in a separate message (FOLLOW_UP), transmitted after the SYNC. This mode of operation is called *two-step*.

We build on these path delay measurements for our network load estimation methodology and tools, which we describe in Section III.

B. QoE monitoring and management for OTT services

Monitoring and managing QoE is receiving significant research attention, but is also associated with significant challenges.

Depending on the level of ISP involvement, there are different approaches for QoE-driven management for OTT services. One option is to push QoE monitoring and relevant adaptation actions to the user side. In this case, a software agent on the user end monitors user experience and performs application behavior adaptation if possible. For example, in a Dynamic Adaptive Streaming over HTTP (DASH) context, if the content provider makes available different versions of a video object, the client can dynamically switch to a different bitrate (and, thus, quality) to match the network conditions and maximize user experience. This is the approach followed by Alberti et al. [7].

On the other hand, El Essaili et al. [8] adopt a network-centric approach and propose an adaptation scheme tailored to OTT video delivery over LTE: A proxy operated by the network service provider appropriately rewrites user requests towards a DASH server, using a QoE model to estimate user experience and taking into account the channel conditions, in order to select the most appropriate video representation, optimizing QoE and wireless resource utilization.

In the same network-provider-centric spirit, Seppänen et al. [9] have proposed a QoE management framework focusing on OTT multimedia services, aiming to maximize user experience and optimize resource utilization. Their framework achieves QoE-aware traffic/admission control by estimating user experience, identifying the traffic requirements of each user and application, and predicting the outcome of potential management actions based on performance models. This framework builds on a data acquisition and monitoring plane, which is where our load estimation tools would naturally reside. Their approach is ISP-driven, which gives significant control over the placement of measurement probes.

Our approach is CP-centric: In Section V, we introduce a design where user requests are redirected based on QoE estimates carried out by the CP, but with a level of cooperation with the ISP.

QoE estimation is largely application-specific. In this work, and as an example, we focused on web content and used a specific web QoE monitoring tool [6] to show the relationship between QoE and network conditions in this context. Switching to other application types, such as video streaming, would necessitate the use of the appropriate user experience estimation tool.

III. PTPv2-based network load estimation

A. Overview and assumptions

In our prior work [5], we devised a generic measurement-based methodology for estimating network load by observing

the behavior of the PTPv2 protocol. In particular, our goal is to be able to infer the conditions in the path between two PTPv2 clocks based on the measured propagation delay of PTPv2 protocol messages. Note that these delay measurements are carried out anyway as the protocol operates, via the exchange of SYNC, DELAY_REQUEST and DELAY_RESPONSE messages. In this work, we apply our methodology in the particular context of OTT content delivery architectures.

Our methodology involves generating synthetic traffic loads which correspond to the typical traffic pattern in the path between two network clocks. By generating different load levels and measuring PTPv2 protocol propagation delays, we can then build empirical models based on measurement data and derive expressions of load as a function of delay. After an expression of load as a function of delay has been derived (learning phase), delay measurements can be applied in real time to estimate current load (real-time phase). It should be noted that PTP packets should go through the same path as data, and there should be no traffic prioritization for PTP traffic.

There are two points which require careful consideration. First, there should be a means of discovering typical traffic patterns. This can be carried out by monitoring traffic while the network operates and deriving statistical models of packet size distributions. Such statistical models can be made available from the ISP. Otherwise, one has to make further assumptions about traffic characteristics. For example, in our prior work [5], we assumed the typical characteristics of Internet traffic. Using data publicly available [10], we found that the distribution of packet sizes in typical Internet traffic is bimodal, with very small and very large (close to the Ethernet MTU size) packets dominating, and we thus generated traffic workloads following this pattern. In this work, we experiment with a different traffic model (see Section III-B). The second important point is the ability to perform actual measurements for the learning phase. Again, ISP-CP cooperation is necessary. If the path cannot be reserved for experiments, then, at least, the ISP should make available load information so that they can be correlated with delay measurements and the dependence of delay on load can be established.

B. Modeling network load as a function of path delay

In this work we are mainly interested in monitoring network paths between data centers and network points close to user premises. Contrary to our prior work, we consider an environment where downstream HTTP traffic dominates, with users downloading web content. In this spirit, to model network load as a function of path delay, we generate HTTP workloads in the web server $\rightarrow$ client direction.

We have set up a testbed which is composed of two Ethernet LANs separated by a linear topology of two hosts, each with two Ethernet interfaces, acting as software *bridges* and connected via a 100 Mbps Ethernet switch, as shown in Fig. 1. This limits the capacity of a path across the two LANs to 100 Mbps. We introduce load conditions by generating traffic between these two hosts. We used dedicated

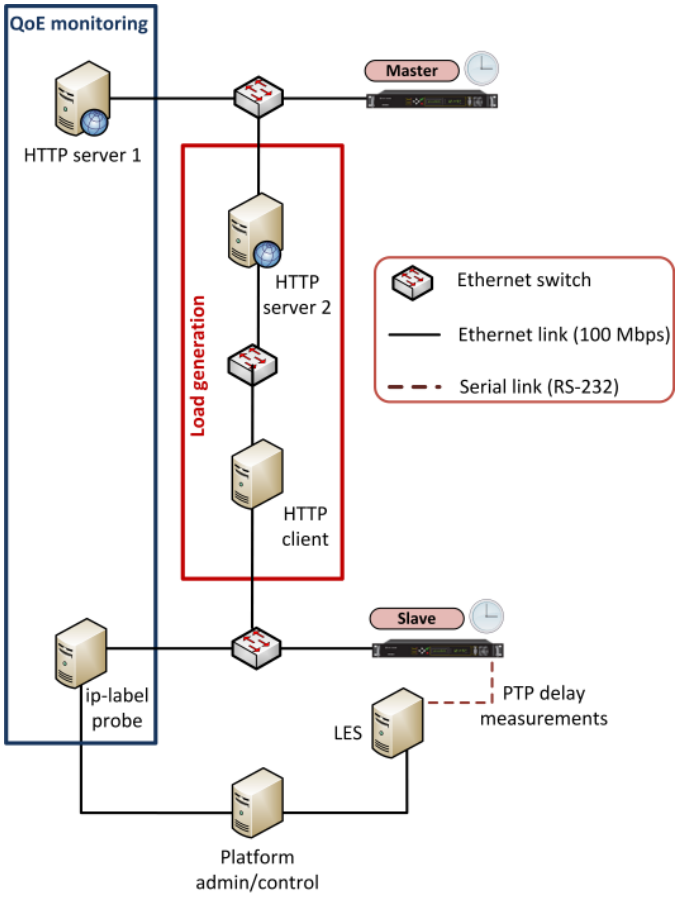

Fig. 1. Experimental testbed.

PTPv2 devices [11] where the protocol is implemented in the hardware of the Ethernet NICs and have installed one clock in each LAN. The slave clock is located in the same LAN as the host acting as an HTTP client, while the master clock is placed in the HTTP server's LAN. The traffic introduced between the load generators (*HTTP client* and *HTTP server 2*, in Fig. 1) shares a path with PTPv2 messages. To emulate varying workloads, we initiate HTTP requests from the client host, limiting the download rate appropriately so that we can achieve the desired load level. At the same time, we record delay measurements extracted from the timestamps of PTPv2 messages, which we collect from the PTPv2 slave hardware over an RS-232 connection.

Each HTTP transmission lasts for 2 hours, and we record one delay measurement upon reception of a DELAY_RESPONSE message by the slave, which we have configured to take place every $4s$. We normalize load values taking into account the capacity of the network; e.g., a value of 0.5 corresponds to a traffic workload of 50Mbps. Fig. 2 shows the evolution of the mean path delay measured from PTPv2 message timestamps for increasing traffic workloads. We observe that there is a load value (0.95) beyond which delay increases by orders of magnitude, signifying a congested path. For the rest of the cases, as Fig. 3 indicates, delay steadily

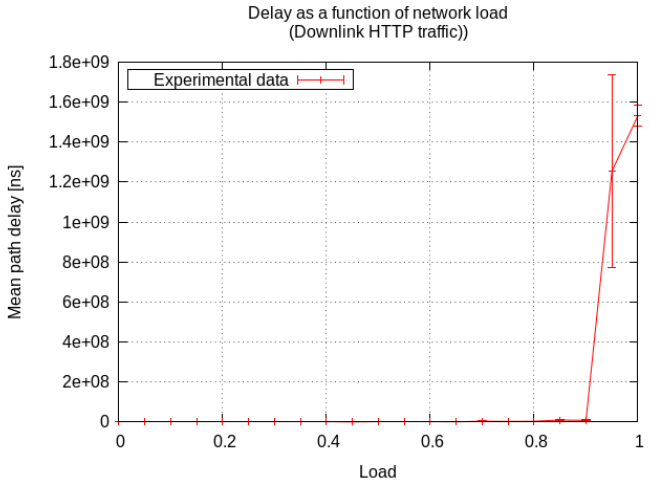

Fig. 2. Path delay as a function of network load. The generated workloads correspond to downlink HTTP traffic.

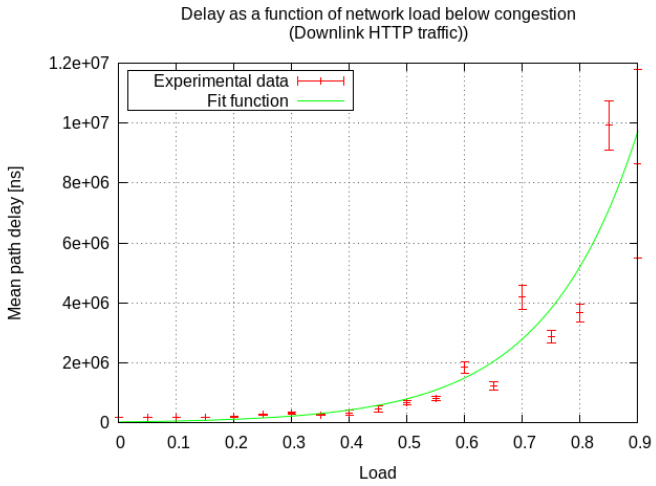

Fig. 3. Path delay as a function of network load for loads up to 0.9. An exponential function ($y(x) = 3.61 \times 10^4 e^{6.218x}$) has been fitted to the experimental data.

increases in an exponential manner.

In Fig. 3, we have fitted an exponential function of the form $f(x) = ae^{bx}$ to the measurement data for the cases where network load is below the congestion point, which describes them with reasonable accuracy. (The coefficient of determination is $R^2 = 0.89$.)

It should be noted that for very low (e.g., < 0.2) load values it is practically impossible to draw a conclusion about the precise network load based on delay.

Similar generic behavior was observed in our prior work [5], for different traffic patterns. We can thus apply the same load estimation approach, but with a different configuration. We provide a short description of our approach in the next section.

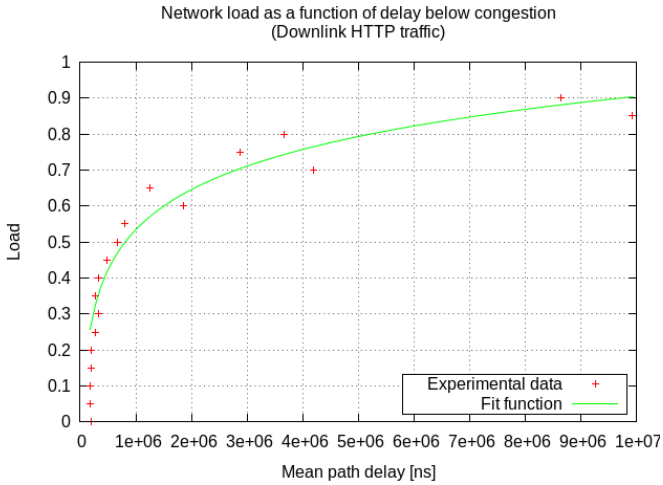

Fig. 4. Network load as a function of delay. Measurement data are plotted with the x-axis representing delay, and we have inverted the exponential expression of delay as a function of load which we had fitted to the measurement data.

C. Load estimation service

Our tool for load estimation works by receiving delay measurements in real time, and using the derived delay-load model to identify the current load level. For the specific network configuration and under the assumed underlying traffic patterns, a low-load threshold and a congestion point can be identified. For the rest of the cases, the empirical function of load with respect to delay is applied, which is the inverse of the fit function we selected in Section III-B. In this example, it is straightforward to invert this exponential expression analytically (Fig. 4).

For robustness against transient load variations, we maintain an exponentially weighted moving average of the current load estimate according to the following equation:

$$\overline{L}_i = w\overline{L}_{i-1} + (1-w)l, \tag{1}$$

where $\overline{L}_i$ is the running load estimate after considering the ith delay sample, l is the load value to which the current delay measurement maps and $w \in [0, 1)$ is a smoothing factor.

The actual value of the smoothing factor is determined after a training phase: We carry out measurements for random *known* load patterns and select the value of w which minimizes the load estimation Root Mean Square Error (RMSE). We found that a value of $w = 0.87$ is optimal in our experimental settings.

We have implemented our load estimation service (LES) with ease of integration in mind. We expose a JSON API accessible over HTTP, making it straightforward to build load-aware applications that take advantage of it. For example, in our prior work [5], we designed a load-aware video bitrate/quality adaptation scheme for Dynamic Adaptive Streaming over HTTP (DASH), which we implemented into the popular open-source VLC media player. In the next section, we integrate it with a web QoE monitoring architecture and

study the correlation of network load with user experience for web applications. Our aim is to further show how our LES can be utilized for QoE-optimized content delivery in an OTT architecture.

IV. Network load as a QoE estimate

We demonstrate quantitatively the correlation between user experience and network load. To this end, we deploy our LES and a state-of-the-art web QoE monitoring tool (*ip-label Datametrie* [6]) to our testbed (Fig. 1), and extend Datametrie so that it periodically receives load estimates from the LES, in order to associate them with QoE metrics.

The QoE monitoring tool we have used fully emulates user behavior by executing predefined web usage scenarios. It integrates a web browser component where it emulates user activities, as specified in scenario configuration files, and can track various user-experience-related metrics. The QoE metric we selected was the *full page download time*, which includes the delay components of (i) DNS resolution for all objects included in a web page, and (ii) the actual TCP transfer of these objects. We present experiments where the estimated load is correlated with the full page dowload time, indicating that the former can actually be used as a QoE estimate to drive content delivery decisions.

Our approach works as follows: While introducing load as described in Section III-B, we initiate a web usage scenario where an emulated user (*ip-label probe*), located at the PTPv2 slave's LAN, accesses a web page stored in a web server (*HTTP server 1*) in the master clock's LAN. User traffic, introduced load, and PTPv2 messages share a path between the two LANs. The LES receives delay measurements directly from the slave device, and the ip-label probe periodically queries the LES, while recording full page download times.

Fig. 5 shows the relationship between the actual load injected in the monitored path and user experience when accessing two types of content: (i) A "small" web page, which is a single HTML file with links to 27 binary objects (total web page size is approximately 2.5 MB), and (ii) a single "large" file (30 MB). There is a clear correlation between the two metrics, which motivates us to apply load estimates, as reported from our tool, as indicators of user experience. This approach has some distinct advantages: It does not require access to the user terminal for measurements and has generic applicability, provided that there is an established link between user experience for the specific application (e.g., web, video, VoIP) and network load. It should be noted, though, that this approach is dependent on the accuracy of the load estimation process. Without careful LES configuration, its output may show significant estimation error.

V. QoE-aware OTT content delivery architecture

In this section, we discuss how our load estimation service could fit in the content delivery process. Our puprose is to be able to monitor in real time parts of the paths between data centers and users.

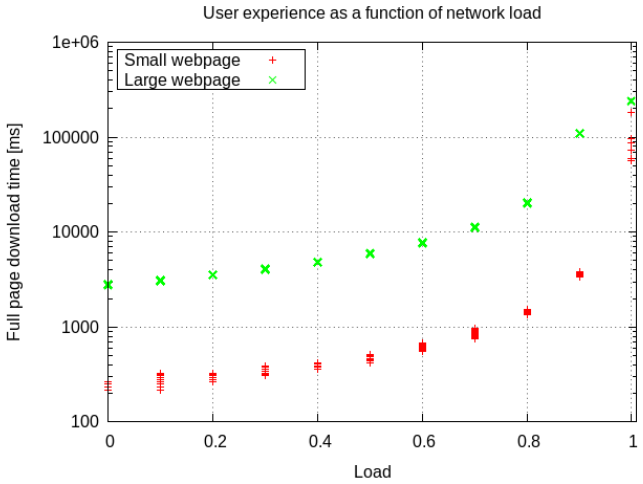

Fig. 5. Full page download time (our QoE metric) for two different web pages as a function of the actual network load.

A. Clock placement and configuration

Ideally, PTPv2 slave clocks should be relatively close to user premises, in order to maximize the length of the monitored path. A reasonable choice would be to host instances of the load estimation service at customer aggregation points inside ISP Points of Presence (PoPs), through which user traffic traverses.

A PTPv2 master clock is collocated with each data center. In each slave clock location, multiple instances of the load estimation service should be present, each one monitoring a single master-slave path. This can be achieved by operating multiple virtual PTPv2 slaves on the same host. This design is compatible with the PTPv2 *Telecom Profile* recommendation [12], where transmission of messages is carried out in unicast mode over UDP and each slave clock instance is configured to synchronize to a single master. We have tested this functionality in a LAN testbed using the *PTPd* [13] open-source implementation of the protocol. Note that the availability of mature software implementations of the IEEE 1588-2008 standard spares the need for dedicated hardware for master and slave clocks.

B. Request redirection

Each time a user requests a specific content item, the following procedure takes place:

1) The user utilizes DNS to resolve the location of the content to an IP address. The request is sent to a resolver, which is an entity managed by the content provider and is responsible for redirecting user requests to the most appropriate data center.
2) The resolver identifies the LES location which is responsible for this user and queries it for load information on the paths between the user and all candidate data centers. The load estimates are mapped to QoE estimates (e.g., full page download times) and the data center which is expected to provide the best user experience is selected.

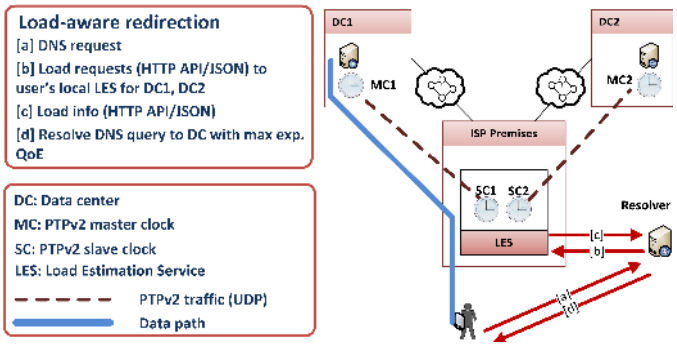

Fig. 6. Using the PTPv2-based load estimation service for load/QoE-aware request redirection strategies in an OTT content delivery environment.

3) The resolver resolves the query of the user to the selected data center.
4) The user downloads the content.

Fig. 6 presents our proposed architecture.

C. Limitations

Our architecture has some limitations. By its nature, it is not possible to directly measure user experience, since no direct information from user terminals is assumed. Also, it is not possible to monitor the full path between the user and the data center, since this is not realistic in our approach. However, we monitor critical paths between data centers and locations relatively close to user terminals. How close these locations are to end users is a matter of design and slave clock placement, which further depends on the level of ISP-CP cooperation.

VI. Conclusion

In this work, we focused on estimating user experience based on network-level indications, as a tool for QoE-optimized content delivery by OTT service providers. Given the challenges posed to OTT providers due to the inherent reduced network awareness, but also given the trend towards ISP-content provider cooperation, we presented a design where network load estimation tools can be deployed inside the ISP network to provide real-time information about the conditions in the content delivery paths. In particular, we applied our IEEE 1588-based network load estimation methodology, using state-of-the-art QoE measurement tools and testbed experiments to establish the relationship between network load and QoE, and showed how load-aware content request redirection strategies can be implemented by the OTT provider in such an environment. We selected web content delivery as our target application, but our methodology, tools and architecture can be extended to other types of content, such as video delivered over HTTP. Our future work will focus on this aspect in particular, by attempting to correlate video QoE with the estimated network conditions and further studying how our mechanisms should be adapted to support QoE-driven delivery of heterogeneous OTT services.

References

[1] B. Frank, I. Poese, G. Smaragdakis, A. Feldmann, B. Maggs, S. Uhlig, V. Aggarwal, and F. Schneider, "Collaboration Opportunities for Content Delivery and Network Infrastructures," *ACM SIGCOMM ebook on Recent Advances in Networking*, vol. 1, August 2013.

[2] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber, "Pushing CDN-ISP Collaboration to the Limit," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 34–44, Jul. 2013.

[3] Z. Li and G. Simon, "In a Telco-CDN, Pushing Content Makes Sense," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 300–311, September 2013.

[4] IEEE 1588 WG, *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE 1588-2008*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA, July 2008.

[5] P. Frangoudis, A. Ksentini, Y. Hadjadj-Aoul, and G. Boime, "PTPv2-based network load estimation," in *Proc. 2013 International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, Sep. 2013, pp. 101–106.

[6] Datametrie Global Experience. ip-label. [Online]. Available: http://www.ip-label.co.uk/products-quality-of-experience-end-user/datametrie-global-experience/

[7] C. Alberti, D. Renzi, C. Timmerer, C. Mueller, S. Lederer, S. Battista, and M. Mattavelli, "Automated QoE evaluation of Dynamic Adaptive Streaming over HTTP," in *Proc. Fifth International Workshop on Quality of Multimedia Experience (QoMEX '13)*, July 2013, pp. 58–63.

[8] A. El Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, and E. Steinbach, "Quality-of-Experience Driven Adaptive HTTP Media Delivery," in *Proc. IEEE International Conference on Communications (ICC)*, June 2013, pp. 2480–2485.

[9] J. Seppänen, M. Varela, and A. Sgora, "An autonomous QoE-driven network management framework," *Journal of Visual Communication and Image Representation*, vol. 25, no. 3, pp. 565–577, 2014.

[10] The CAIDA Anonymized Internet Traces 2012 Dataset. CAIDA. [Online]. Available: http://www.caida.org/data/passive/passive_2012_dataset.xml

[11] SecureSync Synchronization System. Spectracom. [Online]. Available: http://www.spectracomcorp.com/ProductsServices/TimingSynchronization/GPSTimeFrequencyReferences/SecureSyncSynchronizationSystem/tabid/1304/Default.aspx

[12] International Telecommunication Union, *Recommendation ITU-T G.8265.1/Y.1365.1: Precision time protocol telecom profile for frequency synchronization*, Oct. 2010. [Online]. Available: http://www.itu.int/rec/T-REC-Y.1365.1/en

[13] PTPd–Precision Time Protocol daemon. [Online]. Available: http://ptpd.sourceforge.net/