Why so aggressive? Low Extra Delay Background Traffic (LEDBAT) in WebRTC

Riccardo Reale
Peerialism AB / KTH - Royal Institute of Technology, Stockholm, Sweden
riccardo@peerialism.com

Anton Blomberg
Stockholm University, Stockholm, Sweden
anton@naetet.se

Roberto Roverso
Peerialism AB, Stockholm, Sweden
roberto@peerialism.com

ABSTRACT

In this paper, we present what is, to the best of our knowledge, the first implementation of LEDBAT for the WebRTC framework. By providing support for LEDBAT, we enable the development of data-intensive peer-to-peer (P2P) applications on top of WebRTC, such as large scale file-sharing and content delivery. This work constitutes the first step in the process of studying and improving WebRTC's data transport protocol stack in the context of a real use-case, such as P2P video streaming.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Experimentation, Performance, Measurement

Keywords

LEDBAT, WebRTC, Peer-to-peer

1. INTRODUCTION

Recently, WebRTC has been steadily gaining popularity fueled by the inclusion in most of the browsers on the market. The WebRTC standard mandates the multimedia and peer-to-peer connectivity stacks for real-time services, once provided by external plugins, to be built directly into the browser. That means that features of those stacks, such as video acquisition/encoding, encryption and NAT traversal, are made available to developers through HTML5 APIs. WebRTC was mainly designed as a framework to facilitate video and audio conferencing but it does include support for building other types of peer-to-peer applications, such as CDN accellerators [7] and video streaming platforms [8].

Although WebRTC currently provides many features, it lacks support for low priority transfers. Low priority is an essential requirement for P2P data-intensive applications, e.g. BitTorrent and streaming platforms [9], which utilize this feature to avoid disrupting traffic generated by both web browsing and low delay VoIP applications. Support for low priority is provided in P2P network stacks by a delay-based congestion control protocol such as LEDBAT [1]. LEDBAT achieves low priority behaviour by completely yielding to other TCP traffic on the same network and by keeping the delay on a link fixed to a low and constant target.

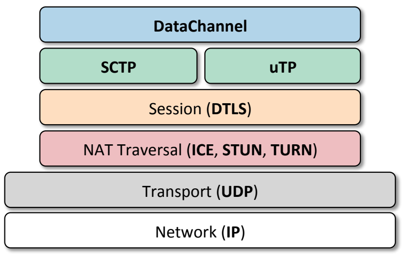

Figure 1: Modified WebRTC protocol stack

In this paper, we present what is, to the best of our knowledge, the first implementation of LEDBAT for the WebRTC framework. By providing support for LEDBAT, we enable the development of data-intensive P2P applications on top of WebRTC. In our specific case however, the interest in LEDBAT is motivated by the the needs of a commercial peer-assisted streaming application called Hive Streaming [6]. Hive utilizes low priority traffic to prefetch data from other peers ahead of the playback deadline without disrupting other traffic in the network.

Besides including LEDBAT in WebRTC, our greater goal is to study the functionality and performance of WebRTC's transport protocols in order to address possible shortcomings that would prevent wide adoption of this technology. For that, as the next step, we intend to provide a detailed analysis of the behavior of WebRTC's transport protocols in a real use-case, that is video streaming.

In order to promote adoption of our WebRTC implementation and other improvements to the WebRTC stack, we make our code available as open-source at [2].

2. LEDBAT IN WEBRTC

In WebRTC, transfer of arbitrary data over encrypted P2P channels is a feature provided by the DataChannel abstraction.

The DataChannel API make use of the Stream Control Transfer protocol (SCTP) [3] protocol in order to manage data transfers. SCTP is a message-oriented protocol that provides a variety of options on data transfer to accomodate the needs of different types of applications. When setting up a DataChannel, WebRTC applications may choose in- or out-of-order delivery and reliable or unreliable delivery. Regarding congestion control, the latest version of SCTP included in WebRTC make use of the same algorithm found in H-TCP [4]. The main characteristic of H-TCP is that it delivers the same priority as TCP in low bandwidth networks while trying to better exploit high delay but large bandwidth links compared to TCP.

The current version of WebRTC's DataChannel incorporates a user-level SCTP implementation [1] written in *C* and built on top of UDP. SCTP interfaces with a security layer, Datagram Transport Layer Security (DTLS), which adds encryption and integrity to all traffic sent through a DataChannel session. All data sent through that session is carried over connections that have been established using the ICE NAT Traversal protocol, last layer of the WebRTC protocol stack.

In order to implement LEDBAT support in WebRTC, we opted to directly integrate the *uTP* library [5] in the WebRTC protocol stack as an optional alternative to SCTP. uTP is the reference open-source implementation of LEDBAT and is provided by BitTorrent. Similarly to SCTP, uTP provides reliability, in-order delivery and is layered over UDP. Given the similarities between the protocols, we were able to easily integrate uTP with the existing WebRTC stack. That, by developing a translation layer that interpret API requests from the DataChannel layer for the uTP layer. Besides that, we modified the DataChannel API to be able to choose between SCTP and uTP as transport protocol when setting up a data channel. Finally, we were able to channel all data processed by the uTP implementation into the DTLS processing. The resulting structure of the WebRTC stack is shown in Figure 1.

3. PRELIMINARY RESULTS

We conducted a preliminary evaluation of our solution in a controlled network environment consisting of two host machines, a sender and a receiver, running Ubuntu Linux with kernel 3.11.0-12 connected through a 100 Mbps link. We used Dummynet on the sender host, configured to cap the outgoing traffic to 1 Mbps, with 100 ms base delay and 60KB of queue buffer to simulate a typical WAN gateway scenario.

In order to test our implementation, we developed a test application in *C* that generates traffic using the WebRTC Native DataChannel API to emulate the traffic of a data-intensive browser application. Our test application can be configured to initialize either a STCP DataChannel, or a LEDBAT DataChannel.

Both hosts run our test application and a separate TCP traffic generator to obtain traffic competing on the same network resources. TCP file transfer uses TCP Cubic, the default Linux congestion control. Finally, a third host acts as a server to handle the WebRTC session signalling between the sender and the receiver.

As metrics, we utilize the throughput of the DataChannel LEDBAT transfers when competing against TCP traffic, and the Round Trip Time (RTT) measured on the link. With these two metrics, we verify that LEDBAT transfers indeed yield to TCP and, at the same time, we evaluate the relative increase on the one-way delay, which directly affects latency and, therefore, web browsing responsiveness.

[1]http://sctp.fh-muenster.de/sctp-user-land-stack.html

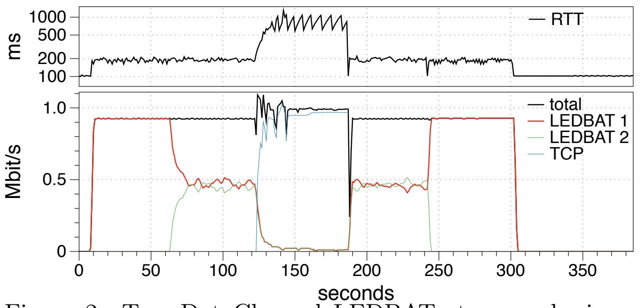

Figure 2: Two DataChannel LEDBAT streams sharing a 1Mbit/s bottleneck with a TCP stream

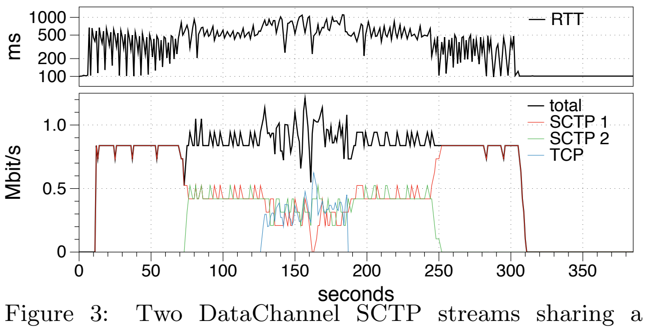

Figure 3: Two DataChannel SCTP streams sharing a 1Mbit/s bottleneck with a TCP stream

Figure 2 shows the throughput evolution of two DataChannel LEDBAT streams and a single TCP stream, starting at 60 seconds from each other. The corresponding Round Trip Time is also shown above. The two LEDBAT flows, as expected, utilize all available bandwidth, until they quickly and completely release the bottleneck in favour of the higher priority TCP flow. The increase in the RTT due to the presence of LEDBAT flows is limited to 100 millisecond, which is the default target delay configuration in uTP. Note that the number of LEDBAT streams doesn't affect the amount of extra delay introduced.

For direct comparison and validation that the normal behavior of SCTP was not affected, Figure 3 shows the same experiment performed configuring our test application to use DataChannel SCTP streams. Besides fairly sharing the link with the TCP transfer as expected, each of the SCTP streams introduce a large amount of extra-delay on the link as TCP does.

4. REFERENCES

[1] http://tools.ietf.org/html/rfc6817.

[2] https://github.com/Peerialism/webrtc-ledbat.

[3] http://www.ietf.org/rfc/rfc2960.txt.

[4] http://www.hamilton.ie/net/htcp.htm.

[5] https://github.com/bittorrent/libutp.

[6] Hive streaming. http://www.hivestreaming.com.

[7] Peercdn. https://peercdn.com/.

[8] J. Nurminen, A. Meyn, E. Jalonen, Y. Raivio, and b. y. Marrero, R. P2P media streaming with HTML5 and WebRTC (Demo).

[9] R. Roverso, S. El-Ansary, and S. Haridi. Smoothcache: Http-live streaming goes peer-to-peer. In *IFIP NETWORKING 2012*.