CDN OPTIMIZATION FOR VR STREAMING

R. van Brandenburg[1], R. Koenen[1], D. Sztykman[2]

[1]Tiledmedia, The Netherlands; [2]Akamai, USA

ABSTRACT

Streaming audiovisual VR content by brute force (i.e., streaming the entire 360° panorama) requires significant bandwidth and provides mediocre quality. Several viewport-adaptive streaming schemes have been proposed to alleviate this; "Tiled VR streaming" is one such method. A major factor determining the QoE of any viewport-adaptive streaming technology is latency, and there are many different types of latency that contribute to the overall experience. We explain these latencies, and what the authors did to reduce overall latency as perceived by the end-user. We give experimental results, showing that Tiled VR streaming using a commercial Content Distribution Network provides a great QoE, and explain how CDN and streaming protocol optimisations contribute to this QoE.

1. INTRODUCTION

Virtual Reality is gaining in popularity; new Head-Mounted Devices (HMDs) are announced almost weekly, and a significant market size is predicted for the coming decade. The study published by Citi Research [1] is one example; it predicts that [the] *"VR/AR market could grow to $2.16 trillion by 2035 as different industries and applications adopt and make us of the technology"*. Other research, e.g. by Piper Jaffray [2], points in the same direction.

This paper focuses on 360VR: an immersive audiovisual experience that is usually consumed using an HMD. The 360VR market will only take off if high quality content can be streamed. YouTube would not be nearly as successful if content needed to be downloaded first. Unfortunately, streaming the entire 360° panorama takes (and wastes!) enormous amounts of bandwidth. A user typically only sees 1/8th of the panorama in the HMD, but the brute-force method that most major streaming platforms use, streams the entire sphere: eight times as many pixels as are consumed. Various techniques for viewport-adaptive streaming enable higher efficiency, by sending the part of the video that the user sees in high quality, while relaying the rest of the sphere in much lower quality. Some of these methods encode many different viewpoints (30 or more) as independent streams, and then switch between those as the user's head turns. Kuzyakov and Pio [3] describe the basic principles in a clear way. The challenge with this approach is switching fast enough when the user's head turns. Switching out one viewport (read: bitstream) for a completely new one takes time and renders the decoder buffer obsolete, causing the bitrate to spike, at the cost of both QoE and efficiency. We employ a different method: tiled streaming. Our method relies on streaming a low-resolution base layer that covers the entire sphere as well as a selection of high resolution tiles that only cover the current viewport. We use network-optimised protocols and "Adaptive Switching Latency" client-

side logic to allow very rapid switching when head motion happens. Note that we use the words “latency” and “delay” interchangeably in this paper.

2. BASICS OF TILED STREAMING

Streaming 360VR by brute force requires significant bandwidth: streaming 4k VR (an equirectangular projection of 4k x 2k pixels) requires over 20 Mbit/s using today’s best video codecs, and streaming 8k VR (8k x 4k) requires more than 80 Mbit/s. The difference between an 8k and a 4k panorama is clearly visible, even with headsets like the Oculus Rift and the Gear VR that are still fairly limited in terms of resolution (a bit more than 1k x 1k per eye). This problem will get worse as the resolution of HMDs increases: headsets with a resolution of 2k x 2k per eye have already been announced. Also, having to decode the full 360 sphere also significantly limits the resolution that can be supported: decoding a full 8k panorama is beyond the capabilities of current smartphones. Tiled VR streaming solves both the bandwidth and the decoding problem. It allows streaming of 4k panoramas at 4 - 6 Mbit/s, and 8k panoramas at 14 - 20 Mbit/s.

For simplicity’s sake, we explain the basic principles using an equirectangular image in this paper, but we use a cubic projection practice, as tiled streaming works significantly better with cube maps. For a primer on these projections, see [4].

The first step in a Tiled Streaming system is cutting up the full VR panorama in rectangular tiles, and encoding each of these tiles such that they are independently decodable. The details of the encoding are codec-dependent, and outside the scope of this paper, but we note that the HEVC standard [5] has native support for “Motion Constrained Tile Sets”, which allows building a system that relies on standards based encoders and decoders.

The tiled, encoded and packaged content can then be placed on the CDN. Next, the trick is to only stream those tiles that are in the user’s field of view (FoV), either entirely or partially. With current headsets, the field of view is typically around 90° x 90° (horizontally and vertically), or 1/8th of the entire panorama of the equirectangular projection – see Figure 1. During playback, the player software determines which tiles are in view and retrieves those tiles from the network using http streaming techniques. No per-user edge processing is required, a significant advantage of this method.

While each tile is effectively an independently decodable video, Tiled VR streaming ensures that only a single, standardized, decoder is needed to decode these tiles. To achieve this, the tiled streaming client rewrites the video bitstream at a very low level prior to presenting it to the decoder. This means that a single hardware or software decoder can be used, typically available on the (mobile) device, with the lowest possible impact on battery life. With only a small part of the entire being decoded, 8k panoramas can be supported on devices that could only ever decode a 4k picture. When a user turns their head, it is important that the response is instant, otherwise they would get sick very easily. Tiled Streaming accomplishes this instant response by always streaming – and decoding – a low-resolution background layer. Having this background layer available implies that motion that users see is fully consistent with the motion they feel, assuming that the headset can track this motion accurately.

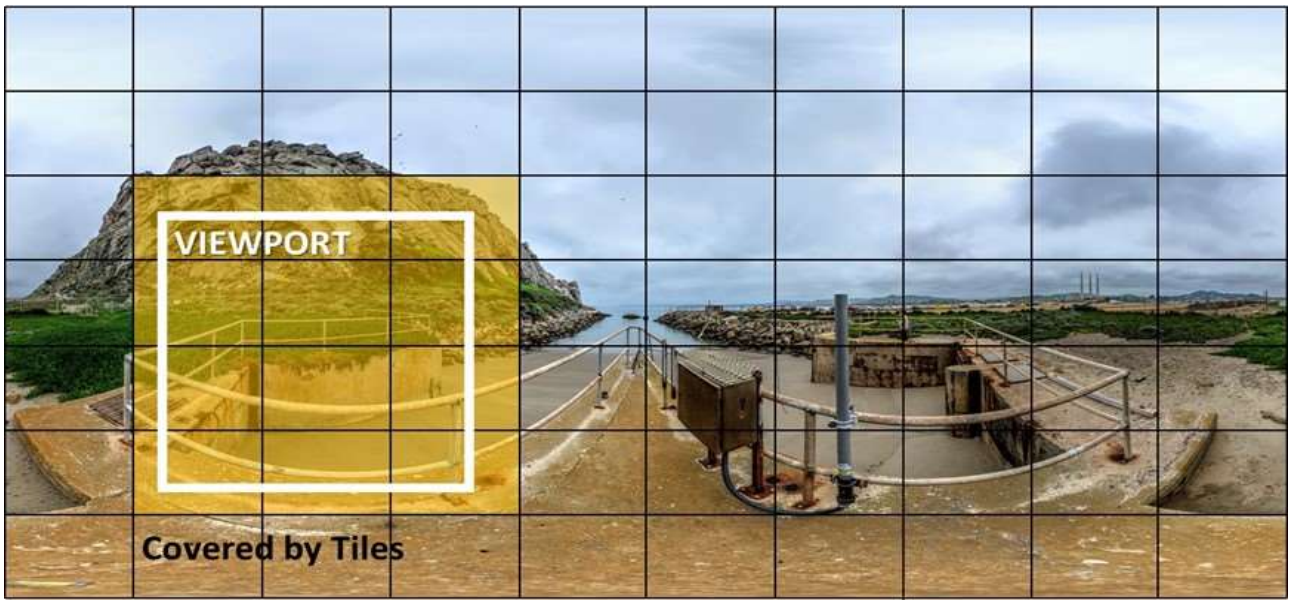

Figure 1 – Viewport and Tiles

When a VR user turns their head, the vestibulo-ocular reflex [6] makes the eyes stay focused on same point for a short while, after which they follow the head to the centre of the new viewport. The system needs to be able to switch to the high-resolution tiles quickly enough for the high-resolution tiles to be present when the eyes are re-aligned and have re-accommodated. Depending on the amount of head movement, and the size of the field-of-view, this means that tiles need to be retrieved within 20-40ms for the low-resolution nature of the background layer to be unnoticeable to the user. With slow head movement, and on HMDs with a very large field-of-view (>90 degrees), we have a bit more time because users are not able to see sharply in their peripheral vision anyway. Given adequate network conditions, our optimized system [7] can switch from the low-resolution background layer to high-resolution tiles within one or two frames, resulting in the switch being fast enough for the switch not to be noticeable or annoying.

3. LATENCIES IN VR STREAMING

As with any viewport-adaptive streaming technology, the quality of experience in Tiled VR streaming is determined by the latency in switching from the low-resolution (and low quality) background to the high-resolution imagery. We call this “motion-to-hires latency”. Note that this is distinct from motion-to-photon latency, which is extremely low because the low-resolution base layer is always present and decoded. Consider the full end-to-end chain from head motion as the trigger, to the user seeing the new field of view in high resolution in the HMD: the total motion-to-hires latency is determined by the factors shown in Figure 2. Looking at these factors in more detail, we can make the following observations.

The **sensor** latency is the time between head movement and the sensor signal becoming available to the VR system. This affects all solutions (including brute force) equally.

The **network request delay** (the client software requesting a new stream or tile from the CDN) depends on the proximity of the CDN edge to the user, and therefore on the density of the CDN: the amount of points of presence and the number of interconnects that a CDN

has. This latency affects all viewport-adaptive approaches, and can be lowered by integrating the CDN with the local Internet Service Provider or mobile operator's network.

The **origin-to-edge** latency is the sum of the delay in the two left-most boxes in Figure 2: latency caused by cache misses in the CDN, where the CDN needs to fetch the data from the origin. Given enough bandwidth in the access network, this is the largest factor in performance *variation*, from one request to the next. It can be lowered using clever ways of prewarming the CDN edge. It could be eliminated completely by pushing the entire 360 sphere to the edge, but this comes at significant cost, which we want to avoid.

The most important latency factor is transport over the **access network**, from edge to customer premises, and over the **local (in-home) network** when LAN or Wifi is used. The underlying causes are round-trip-time (typically higher on mobile networks), available bandwidth (which can vary wildly) and request size. The latter two factors together form the **transmission delay**, or the total time it takes to move the bits over the access network. Whereas the streaming stack cannot change the round-trip-time and available bandwidth, request size is an important factor, and we'll discuss this in the Section 4.

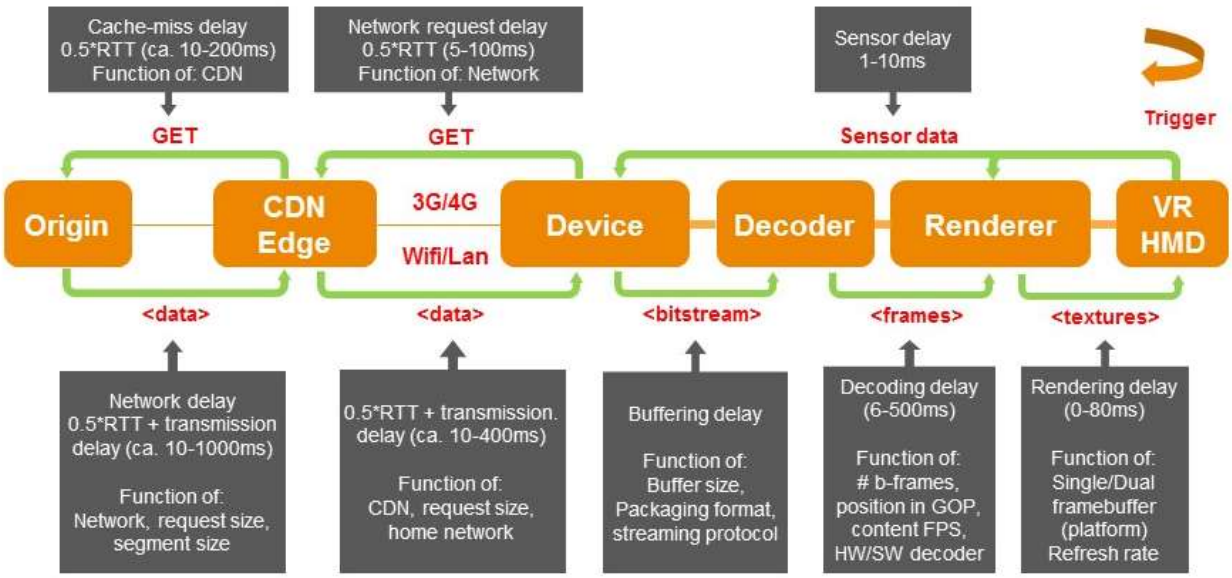

Figure 2 – Overview of latency factors in a Tiled VR Streaming system.

Buffering in client device before decoding is the latency incurred between receiving the tiles (bitstream fragments) and presenting the final bitstream to the decoder. This latency strongly depends on the streaming protocol and the packaging format. It can vary wildly depending on the implementation, and strongly impacts on user-perceived latency.

Decoding delay is determined by random access latency and the design of the decoder pipeline. Latency incurred by retrieving data back to an Independently Decodable Reference (IDR) frame and decoding all frames from that IDR frame until the required frame can be very significant. Many solutions suffer from it, including those that switch out the entire

viewport. The exact delay depends on the frequency of IDR frames in the stream. The presence of B-frames and the framerate also impact this delay significantly.

How much delay is incurred by the **rendering** in the HMD depends on the operating system's frame buffer architecture. Most video players and platforms use a dual frame buffer. Some platforms, such as Google's Daydream, have a single decoding buffer. While one frame difference seems small, at 30fps this is 30ms, which is a lot when one wants to switch tiles within 20-40ms. Again, the effect is lowered with increasing framerates. Note that this latency also affects the low-resolution base layer; it actually affects *all* VR streaming solutions. Together with sensor latency, this determines the motion-to-photon latency in a tiled VR streaming system.

The size of decoding viewport (not depicted in Figure 2) also affects motion-to-hires latency. Latency can be reduced by simply retrieving *and decoding* more tiles than are displayed; high-resolution tiles that are required for limited head motion will then already be available, eliminating all latencies up to the rendering – obviously at the expense of increasing bitrate. For 8k panoramas, this may push the decoder beyond its Profile/Level capabilities. Just fetching more tiles to the device *without decoding* also increases bitrates but pushes the decoder less, so this can be applied to higher resolution panoramas. This will eliminate all latencies up to the decoding. But both approaches increase bitrates, while the motion-to-hires latency in a well-designed system is so short that this is not required.

The next section will describe how our Tiled VR implementation, using a combination of client-side logic, optimal packaging, and CDN optimizations, reduces the total motion-to-hires latency, from sensor to decoding, to an unnoticeable less than one frame.

4. OPTIMISING THE NETWORK STACK FOR VR STREAMING

Insufficient bandwidth, either in the last mile or on the Wifi connection at home, will determine much of the latency in a real deployment. However, very significant gains can be made by making improvements to both the client and the CDN side, to make the most efficient use of the available bandwidth, and to reduce latency as much as possible. Working together, the authors have developed a number of optimisations for Tiled VR streaming in Akamai's Content Delivery Network as well as in Tiledmedia's Tiled VR client implementation. These optimisations target the two most prominent factors of latency in the end-to-end system: origin-to-edge delay, and transmission delay.

Transmission delay

As mentioned, the total transmission delay is determined by three factors: 1) round-trip-time, 2) available bandwidth (bits per second) and 3) request size (number of bits per request). The first two elements are fixed, and to reduce total latency, we need to focus on the request size: by minimizing the total number of bits that need to be sent.Traditional streaming solutions, such as HLS and MPEG DASH, use segmented content: with each segment typically between 3 and 30 seconds long, and the client sequentially retrieves these chunks for playback. This poses a problem for tiled streaming, or any FoV-adaptive streaming solution: it fixes the request size to the size of a segment. This means that with the head turning at frame X, instead of retrieving a bunch of frames starting with X+1, one must fetch the full segment of which frame X+1 is a part. Many bytes may have to be transferred before frame X+1 arrives, with a potentially huge impact on latency.

Our implementation uses a custom packaging format to provide spatial as well as temporal random access, to every frame frames of every tile. In combination with byte range requests, this avoids having to request any bytes that are not absolutely necessary, resulting in a very significant reduction of transmission delay.

Origin-to-edge delay

With, e.g., long tail content that's not frequently accessed, it's likely that a specific spatio-temporal tile is not available on the CDN edge. Given the significant latency induced by fetching data from the origin, the number of cache misses has a huge impact on the average tile switching latency. Our platform includes elements to try to reduce the number of cache misses. First, we use an intelligent cross-tile packaging algorithm, to make sure that data that is likely to be fetched closely in time is also packaged closely together. This increases the chances of the CDN caching the right bits. Second, we give the CDN hints as to which data is likely to be fetched soon. Using a combination of content-specific heuristics as well as user behaviour, the client can predict with tiles are likely to be requested shorltly. By conveying this information to the CDN, the CDN can fetch this data from the origin in anticipation, and place it on the edge, ready for retrieval by the client.

CDN & transport optimizations

Being closer to consumers means a lower round-trip time (RTT) and a higher bandwidth connection, so larger CDNs have a natural advantage, and Akamai's CDN has over 3500 points of presence. As the client is doing byte range request, the Akamai edge is using a mechanism to request byte ranges in 10MB chunks from the origin and stores these in the cache. When the next byte is requested, this range is likely to be already available. This process is illustrated in Figure 3.

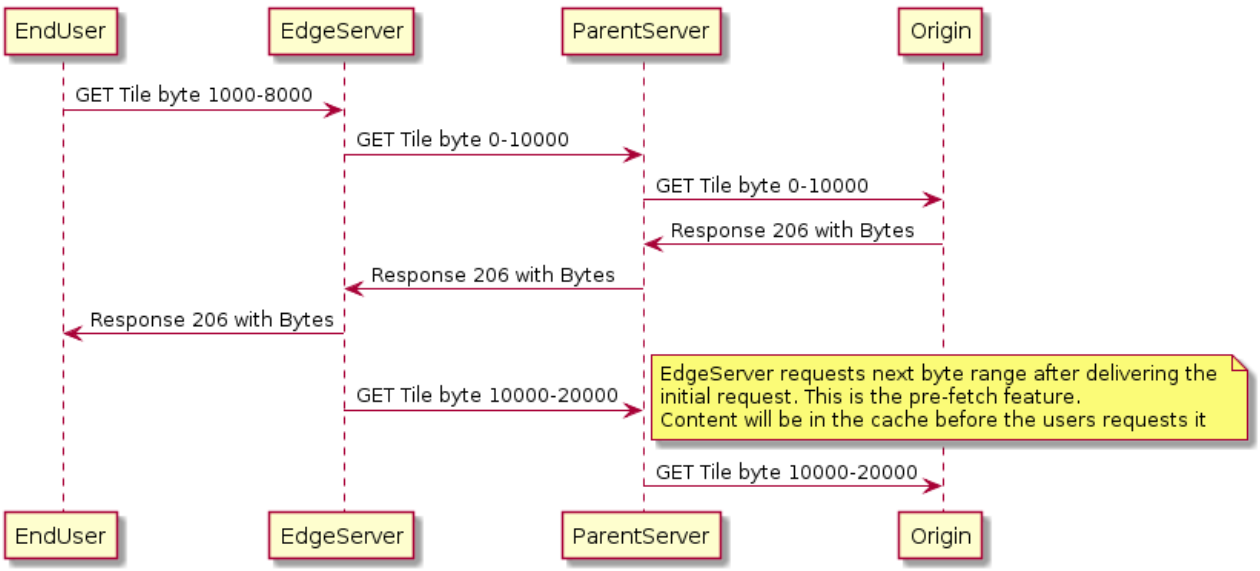

Figure 3 – Caching Process

Finally, transport-level improvements help reduce the latency. While the client is compatible with HTTP/1.1, tile switching latency is reduced when using a CDN that uses HTTP/2 and/or QUIC. QUIC (for Quick UDP Internet Connections, pronounced as "quick") is an

experimental transport layer network protocol originally designed by Jim Roskind at Google [8]. QUIC supports a set of multiplexed connections between two endpoints over UDP and was designed to provide security protection equivalent to TLS/SSL, along with reduced connection and transport latency, and bandwidth estimation in each direction to avoid congestion. QUIC's main goal is to improve perceived performance of connection-oriented web applications that are currently using TCP. With QUIC we can establish a new secure connection with zero RTT, meaning the server can start exchanging data right away, which gives a further reduction in latency. We done our measurements using HTTP/2 and are now implementing QUIC for a further latency reduction.

5. EXPERIMENTAL RESULTS

Experimental Conditions

We performed tests with simulated (i.e., recorded), but significant head motion using 30 frames per second, 8k x 4k panorama encoded using 96 high-resolution tiles. Measurements were averaged over 5 runs of 3 minutes. The content was encoded using separate HEVC encoders for each of the tiles, with tile size fixed at 512 x 512 pixels. The bitrate from the edge to the end user was between 12 and 16 Mbit/s; we note that this is a relatively easy scene to encode, but that scene complexity does not significantly affect what we measured in this experiment. The number of tile switches in each run is about 600, so the measurements cover a total of approximately 3,000 tile switches. Before each run, the cache was cleared. Note that this represents a worst-case condition, as popular content will often be available at the cache already. The content was played back using Akamai's production CDN; we did not use or require special experimental configurations as we deployed the optimizations developed in this joint effort to Akamai's production platform. The content item used was a 3-minute recording of a Blue Man Group show music item. It includes scene cuts and occasionally a slowly moving camera. The conditions are summarised in Table 1, below.

Table 1 – Summary of Experimental Conditions

Content	Blue Man Group, "Live at Luxor", recorded by Digital Immersion, usage courtesy of Harmonic, Inc.
Clip length	3 minutes
Resolution	8k x 4k Equirectangular Projection, transformed to Cubemap
Encoding	Customized HEVC encoder, with motion vectors constraints
Bitrate	12 – 16 Mbit/s from Edge to HMD, depending on head movement and scene complexity
Layers	Two, one high resolution and one low-resolution fall-back
Tile size	512 x 512
Tile switches	~600 per run
Runs	5

Experimental Results

Figure 4 depicts tile switching latency measured using Tiledmedia client, connected over Wifi and streaming from Akamai's CDN. The diagram shows the average time it takes for a tile switch to be completed (i.e. the motion-to-hires latency), in a variety of scenarios.

The grey line with the triangles shows the situation with a warm cache, meaning all tiles were on the CDN edge, with no cache misses. As shown, in this case more than 95% of the tile switches were completed within one frame, meaning they were imperceptible by the end-user. The impact of the cache hinting technology can be seen by comparing the blue and orange lines. The blue line with the diamonds shows the situation with a cold cache and without the cache hint technology. In this case, the impact of a cache miss is clearly visible, with some tiles taking more than 10 frames, or over 300 msec., to arrive, which will be noticed by the user. The orange line with the squares denotes tile switching with cache hinting enabled. While the average latency is not as good as for a fully warm cache (obviously not all head movement can be predicted), it significantly lowers the average switching latency. Some tile switches may be noticeable, but most are not.

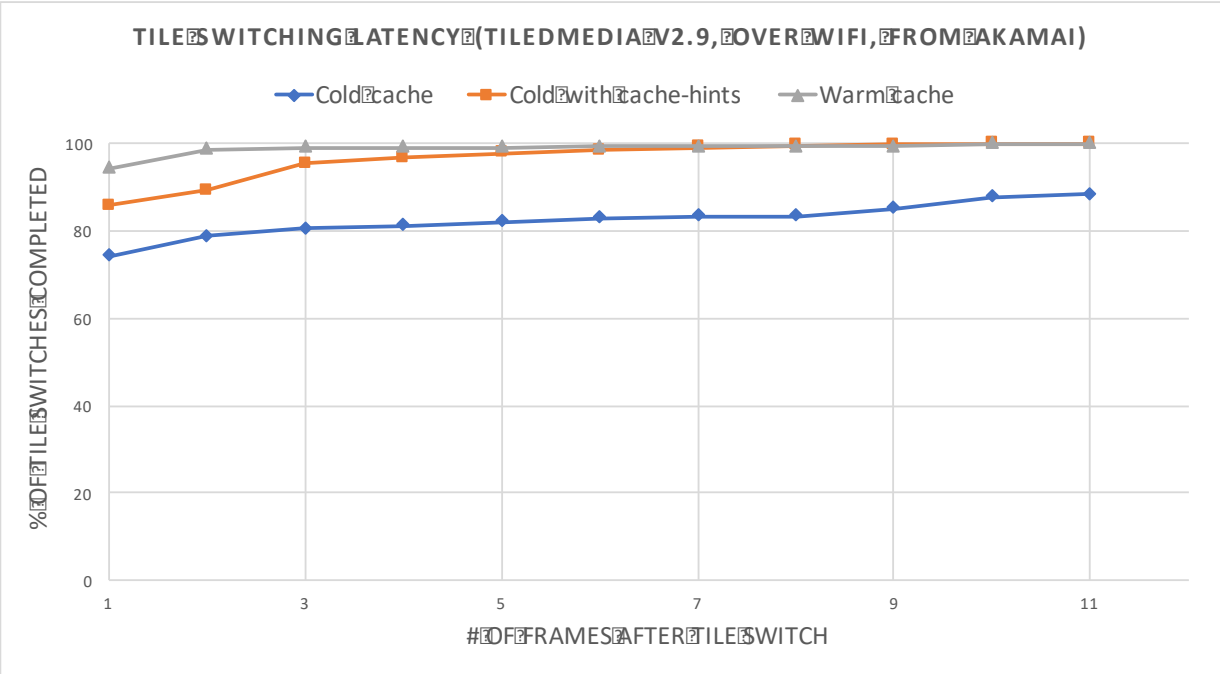

Figure 4 – Average Tile Switching Delay.

These measurements were done with pre-recorded and then simulated head motion; visual tests over a number of content different items confirm what Figure 4 indicates. With the optimized network stack, and cache hinting enabled, tile switches were seldom noticeable and never obtrusive.

5 CONCLUSIONS

For VR360 services to become successful, they need to be able to be streamed at high quality, which requires bandwidth savings of an order of magnitude over today's "brute force" delivery methods. Tiled VR streaming provides very significant bitrate savings over brute force VR360 distribution, and it is substantially more efficient than viewport-adaptive streaming methods that switch out the complete viewport, and it provides a better user experience than those approaches. A careful analysis of the various latencies in the retrieval of high resolution tiles has allowed us to optimize the protocols in our CDN network and in our tiled streaming network stack, with measurable and visible improvements, to the point that the user barely ever notices the presence of the low-resolution background layer.

We intend to continue our research to further improve the bandwidth efficiency of our VR streaming solution, and to further lower the latencies that determine the user experience.

REFERENCES

[1] Citi GPS, 2016. Virtual and Augmented Reality, Are You Sure It Isn't Real?

[2] Munster, G., Jakel, T., Clinton, D., Murphy, E., 2015 Next mega tech theme is virtual reality, Piper Jaffray Investment Research.

[3] Kuzyakov, E. and Pio, D. 2016. Next-generation video encoding techniques for 360 video and VR. tinyurl.com/z5uxuh7, last accessed 8 May 2017.

[4] Kuzyakov, E. and Pio, D., 2015. Under the hood: Building 360 video". tinyurl.com/nwrcr5b, last accessed 8 May 2017.

[5] ISO/IEC JTC1 SC29 WG11 (MPEG), 2015. ISO/IEC 23008-2:2015 Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding

[6] Wikipedia, "Vestibulo–ocular reflex", tinyurl.com/y7jtq8qc; last accessed 8 May 2017

[7] Tiledmedia, "Technology", tiledmedia.com/index.php/technology/, last accessed 8 May 2017

[8] QUIC, https://www.chromium.org/quic, last accessed 8 May 2017