# An HTTP/2-Based Adaptive Streaming Framework for 360° Virtual Reality Videos

---

## 1、Related work：

(1) 360° video streaming

- 下载低分辨率的整个视频的全部tile，防止黑屏。
- 多播传输，提出算法来确定要多播的贴图的分辨率并使所有用户的效用最大化。

但是都是基于H.264平铺的，但其实H.264不支持平铺，导致每个tile都必须独立解码。

- 用H.265对视频空间进行分割。
- 提出了一种基于可用带宽和用户视口的速率分配算法来决定每个贴图的质量。（不足：HTTP1.1且没有考虑用户的移动）
- Gaddam et al. use tiling and viewport prediction to stream interactive panoramic videos, where part of the panorama can be used to extract a virtual view.（不知道是在干啥，不足：HTTP1.1，H.264）

(2) HTTP/2-Based Adaptive Streaming

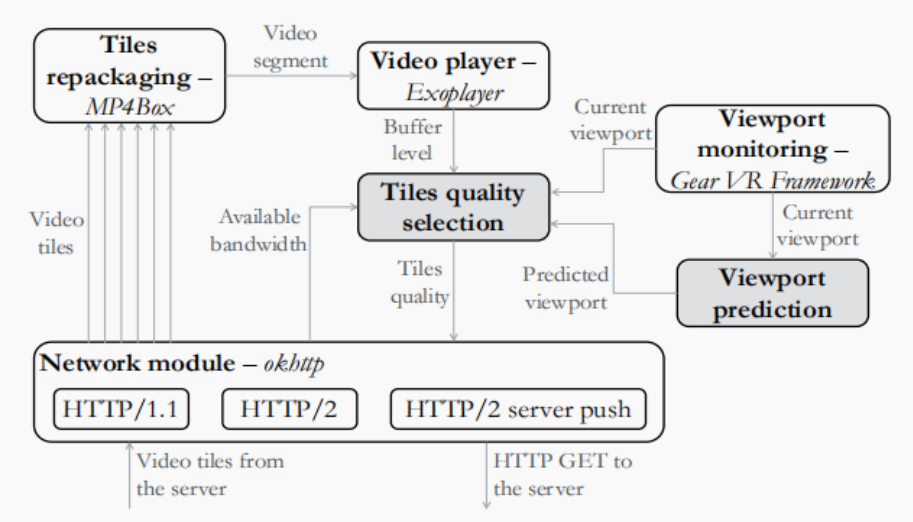
HTTP/2引入的一个新特性是服务器可以推送客户端没有直接请求的资源，以减少web和多媒体交付的延迟。

- Wei等人关注了服务器推送如何改善HAS流的交付。（减少相机到显示器的延迟，且在客户端发出一个HTTP GET请求后推送k个段）
- Xiao等人通过根据网络条件和电能效率动态改变k值，优化移动设备上的电池寿命。
- van der Hooft等人研究了H.265视频在4G网络上的服务器推送的优点。
- Cherif等人将服务器推送与WebSocket结合使用，以减少DASH流会话的启动延迟。

我们利用k-push机制且利用服务器推送功能，以减少由于将视频空间划分为单独的块而引入的网络开销。

## 2、框架提议

框架整体构成：




框架的三大亮点：

- 使用H.265标准。
- 在客户端中为每个贴图选择**最佳的视频质量**的算法。
- HTTP/2协议的服务器推送k-push功能。

## （1）H.265 Video Tiling

Tile的方法比Non-tile的方法显著减少存储空间和带宽。2 *polar* tiles and 4 *equatorial* tiles。

证明：

$$S_{\text{tiled}} = d \times (b_1 + b_0) \times \alpha(n_t)$$

解释：

$$\alpha(n_t) \text{表示平铺过程引入的编码开销}, d \text{是视频时长}, level(0 \rightarrow n_q)$$

$$S_{\text{non-tiled}} = d \times \left[ \frac{w_p \times h_p}{w \times h} \times b_1 + \left(1 - \frac{w_p \times h_p}{w \times h}\right) \times b_0 \right] \times N(w, h, w_p, h_p, \xi)$$

解释：

$$w_p \text{和} h_p \text{是视角的宽和高}, w \text{和} h \text{是整个视频的宽和高},$$
$$N(w, h, w_p, h_p, \xi) \text{是不同视角需要编码的数量}$$

得到平铺和非平铺方法在存储方面的收益，可以量化为:

$$G = \frac{S_{\text{non-tiled}}}{S_{\text{tiled}}} = \frac{b_1 - b_0}{b_1 + b_0} \times \frac{w_p \times h_p}{w \times h} \times \frac{N(w, h, w_p, h_p, \xi)}{\alpha(n_t)} + \\ + \frac{b_0}{b_0 + b_1} \times \frac{N(w, h, w_p, h_p, \xi)}{\alpha(n_t)}$$

## （2）Tiles Quality Selection

为了计算未来的视口，我们获得当前注视点在VR视频二维投影上即时k的位置p，例如，等直角投影的经纬度。

$$p(k+\Delta) = p(k) + \Delta \times \widehat{p(k)}$$
$$\widehat{p(k)} = \frac{p(k)-p(k-\delta)}{\delta}$$

解释：

$\Delta$是视频片段的持续时间，$\delta$是测量速度的时间间隔$(100ms)$，$\widehat{p(k)}$是速度

且，当计算出了未来的视口，这些图块就被逻辑地分为三种类型:视口内*viewport*、相邻的图块*adjacent*、视口外的图块以及所有剩余的图块*outside*。

算法伪代码，体现了优先级的想法：

$qt(t) = 0 \quad \forall t \in \{viewport, adjacent, outside\}$
$B_{budget} = B - n_T \times b(0)$
**for** $tiles\_category$ in $\{viewport, adjacent, outside\}$ **do**
  $qt = \max_{q \in [1;n_q]} q \quad s.t. \quad b(q) \leq \frac{B_{budget}}{n_{tiles}}$
  $B_{budget} = B_{budget} - n_{tiles} \times b(qt)$
  $qt(t) = qt \quad \forall t \in tiles\_category$
**end for**

## （3）HTTP/2 Server Push for Tiled Videos

HTTP 1.1： $n*{t}$ gets(tiles) --> need $n*{t}$ RTTs.

HTTP/2 : k-push的方法，令k = $n_{t}$，则只有一个请求从客户端发送到服务器。