Virtual Reality Streaming over 5G

Project 5G Hub Vaasa

Author: Dennis Bengs

Funded by:
European Regional Development Fund
Regional Council of Ostrobothnia

Participating organisations:
University of Vaasa
Vaasa University of Applied Sciences (VAMK)
Novia University of Applied Sciences

Project webpage:
https://www.5ghubvaasa.fi/

Contents

Virtual Reality Streaming over 5G

Introduction

This paper discusses the development of an edge-based Virtual Reality (VR) streaming use case for the Technobothnia 5G laboratory, developed as part of the 5G Hub Vaasa project. The project was funded by the European Regional Development Fund (ERDF) and the Regional Council of Ostrobothnia.

The paper It will outline the fundamentals of VR streaming, explore available technologies for this purpose, describe the selected technologies for the use case, and address the challenges faced during development.

What is VR Streaming?

Virtual Reality (VR) and Augmented Reality (AR), together known as Extended Reality (XR), have gained attention for their applications in both the business and entertainment sector. Extended Reality (XR) encompasses both Virtual Reality (VR) and Augmented Reality (AR) technologies. In Virtual-Reality, users are completely immersed in a virtual environment. Augmented Reality on the other hand overlays virtual elements onto the real-world using devices like smartphones, AR glasses (such as Microsoft Hololens) or VR headsets with video passthrough (such as Meta Quest 3).

VR is an inherently graphically intensive technology, requiring a low latency between action and feedback and a high pixel density to feel immersive and preventing motion sickness. VR headsets have typically required powerful computers or used simplified graphics to allow lighter hardware, such as smartphones and Android-based VR headsets, to display the virtual scene. VR streaming is a way to get around this restriction by rendering high fidelity scenes on nearby edge servers, which are then streamed as video and audio to the local VR headset.

While current telecommunication technologies are somewhat effective in supporting VR streaming, there are still limitations in terms of latency and overall reliability. These challenges become particularly apparent in use cases that require real time transmission of large amounts of data.

The introduction of 5G networks brings advancements in Ultra Reliable Low Latency Communication (URLLC) and Enhanced broadband (eMBB) capabilities that could be beneficial in Virtual Reality streaming. This document will delve into the approaches of designing a VR streaming solution running on a 5G network.

In this context streaming refers to the real-time transmission of Extended Reality content from a server to the user's XR device. Unlike video streaming, which involves a single 2D video stream, VR streaming is more complex. It involves delivering 1 to 2 high resolution video streams (and sometimes even a depth stream/other depth data for local 3D reconstruction), spatial audio and requires support for low-latency real-time user interactions. The goal is to create an immersive experience that feels seamless, where latency is minimal, and interactions feel natural and "local".

Common networking issues with Virtual Reality streaming are latency: the delay between sending and receiving data, and bandwidth: the maximum rate of data transfer. When the network fails to meet these demands, users can experience several issues such as motion sickness.

The URLLC (Ultra-Reliable Low Latency Communications) pillar of 5G networking provide lower latencies while ensuring reliable transmission of data. This is beneficial in VR streaming where quick response time and reliability are important.

VR streaming in business

The low-latency, high-reliability, high-bandwidth characteristics of 5G have many applications that meet various business needs across multiple industrial sectors. What follows are a list of use cases of Virtual Reality streaming within a few select industries:

- **Manufacturing**: Virtual walkthroughs of factory layouts for better planning and remote operation/inspection/troubleshooting. A highly detailed immersive floorplan can be streamed to lightweight XR glasses from a local server.
- **Retail**: Small retailers can showcase their products to an audience through storefronts without needing physical stores. Low latency streaming enables real time customer interactions and in-store customer service.
- **Healthcare**: Medical services can be improved through consultations or training in a VR environment.
- **Entertainment**: High quality VR streaming enhances gaming and virtual live events. Low-latency high-definition streaming is particularly important in gaming.
- **Education**: Virtual classrooms and training environments can provide learning experiences.
- **Real Estate**: Virtual property tours simplify the buying process in real estate.
- **Inspections**: Industries such as energy or agriculture can utilize VR streaming to enable real time monitoring from a remote location.
- **Robotics**: Robots could perform hazardous tasks remotely using low latency VR streaming.

Companies operating in these sectors could potentially enhance their efficiency, as well as explore new ways to engage with their customers and maybe even create entirely new services based around Virtual Reality.

Available Technologies for VR Streaming

In this section we will explore the current options that exist for VR streaming. This information was gathered from many sources, including Wikipedia, the linked websites and ChatGPT. This is not a comprehensive list of VR streaming solutions since it's a relatively new and growing field.

Game Engines

Game engines are software frameworks created to assist in the development and production of video games, simulations, and other interactive 3D environments. When it comes to VR streaming, game engines can function as either the server rendering the scene or as the local client software displaying the rendered stream to the user.

- **Unity**:
  - https://unity.com/
  - **Description**: Has native support for various VR hardware and offers a range of networking and streaming solutions.
  - **Cost and Open-Sourceness**: Free for small projects, with paid licenses for larger commercial projects. Closed source.
  - **Platform**: Supports VR headsets, smartphones, and computers.
  - **Community**: Large community with active forums.
  - **Notes**: Easy. Popular. well-documented.

- **Unreal Engine**
  - https://www.unrealengine.com/
  - **Description**: Unreal Engine is known for its high-quality graphics. Like Unity, Unreal Engine has native supports for most consumer VR headsets, with solutions for various streaming technologies.
  - **Cost and Open-Sourceness**: Free with royalty fees for commercial products. Source code is available but Unreal Engine is not open-source.
  - **Platform**: Supports VR headsets, smartphones, and computers.
  - **Community**: Large community with forums.
  - **Notes**: Not as easy as Unity but not overly difficult.

Streaming Protocols

Streaming protocols operate at a lower level compared to software frameworks. A protocol refers to the set of rules that govern how content, such, as video and audio is transmitted and received over a network. There are many types of protocols specialized for streaming different types of content such as video or audio.

- **RTMP (Real-Time Messaging Protocol)**
  - https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol
  - **Description**: Commonly used for streaming video and audio.
  - **Cost and Open-Sourceness**: Has open-source implementations.
  - **Platform**: Works across VR headsets, smartphones, computers, and web platforms.
  - **Notes**: May require some networking knowledge.

- **WebRTC**
  - https://webrtc.org/
  - **Description**: Designed for real-time communications and low-latency streaming.
  - **Cost and Open-Sourceness**: Open-source standard.
  - **Platform**: Supported on web platforms, smartphones, and some VR headsets support hardware encoding/decoding.
  - **Notes**: Difficult, demands understanding of real-time network communications.

- **MPEG-DASH**
  - https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP
  - **Description**: Used in high-quality adaptive streaming.
  - **Cost and Open-Sourceness**: Has open-source implementations.
  - **Platform**: Supported on most platforms including web browsers, smartphones, and VR headsets.
  - **Notes**: Requires understanding of adaptive bitrate streaming.

Libraries, SDKs and APIs

Libraries and Software Development Kits (SDKs) provide pre-made functions, routines and tools that simplify the process of building applications. APIs are a set of functions provided by an underlying software, driver or operating system.

- **NVIDIA CloudXR**
  - https://developer.nvidia.com/cloudxr-sdk
  - **Description**: A GPU-accelerated XR streaming platform for NVIDIA hardware.
  - **Cost and Open-Sourceness**: Closed-source.
  - **Platform**: For VR headsets and computers using NVIDIA hardware.
  - **Notes**: Requires familiarity with the NVIDIA ecosystem.

- **OpenVR**
  - https://en.wikipedia.org/wiki/OpenVR
  - **Description**: APIs that enable streaming to a wide array of VR devices.
  - **Cost and Open-Sourceness**: Not open-source but has open-source wrappers.
  - **Platform**: Supports many VR headsets and computers.
  - **Notes**: May require manual integration.

- **WebVR/WebXR**
  - https://en.wikipedia.org/wiki/WebXR
  - **Description**: Web-based frameworks that allow the creation of browser-accessible XR experiences.
  - **Cost and Open-Sourceness**: Open standard.
  - **Platform**: Web platforms, some smartphones, and specific VR headsets.
  - **Notes**: Web development skills needed.

- **Oculus SDK**
  - https://developer.oculus.com/documentation/
  - **Description**: Software development kit for Oculus devices.
  - **Cost and Open-Sourceness**: Free but closed-source.
  - **Platform**: Specific to Oculus VR headsets.
  - **Notes**: Limited to the Oculus ecosystem.

Server Software

Server software handles the transfer of video and audio data from the source to end users.

- **Wowza Streaming Engine**
  - https://www.wowza.com/streaming-engine
  - **Description**: Offers support for 360-degree VR streaming over protocols like RTMP and WebRTC.
  - **Cost and Open-Sourceness**: Closed-source commercial software.
  - **Platform**: Compatible with VR headsets, smartphones, computers, and web platforms.
  - **Notes**: Setup is well-documented.

- **Red5 Pro**
  - https://www.red5pro.com/
  - **Description**: Provides low latency streaming over multiple protocols like WebRTC and RTMP.
  - **Cost and Open-Sourceness**: Closed-source commercial software.
  - **Platform**: Compatibility with VR headsets, smartphones, and computers.
  - **Notes**: Requires an understanding of various streaming protocols.

- **Ant Media Server**
  - https://antmedia.io/
  - **Description**: WebRTC-based low-latency streaming.
  - **Cost and Open-Sourceness**: Closed-source commercial software.
  - **Platform**: Supports web platforms, smartphones, and specific VR headsets.
  - **Notes**: Documentation available for setup and configuration.

Cloud Services

Cloud services offer complete solutions for video streaming without the need for hosting your own servers.

- **AWS IVS (Interactive Video Service)**
  - https://aws.amazon.com/ivs/
  - **Description**: Live streaming solution which uses the AWS ecosystem.
  - **Cost and Open-Sourceness**: Commercial; based on usage. Closed-source.
  - **Platform**: Supports smartphones, computers, and web platforms.
  - **Notes**: Familiarity with AWS required.

- **Azure Media Services**
  - https://azure.microsoft.com/en-us/products/media-services
  - **Description**: Microsoft's cloud-based platform offering VR streaming.
  - **Cost and Open-Sourceness**: Closed-source Commercial software.
  - **Platform**: Supports VR headsets, smartphones, and computers.
  - **Notes**: Understanding of Azure services required.

- **Google Cloud's Anvato**
  - https://cloud.google.com/blog/products/gcp/welcome-anvato-to-the-google-cloud-platform-team/
  - **Description**: Offers encoding and streaming capabilities.
  - **Cost and Open-Sourceness**: Closed-source commercial.
  - **Platform**: Supports VR headsets, smartphones, and computers.
  - **Notes**: Some experience with Google Cloud services required.

Specialized VR Streaming Solutions

There are platforms designed specifically to deliver high quality VR streaming experiences.

- **Tiledmedia**
  - https://www.tiledmedia.com/
  - **Description**: Focuses on delivering high-quality VR streaming experiences.
  - **Cost and Open-Sourceness**: Closed-source commercial.
  - **Platform**: VR headsets.
  - **Notes**: Specialized for VR streaming.

- **NextVR**
  - https://xinreality.com/wiki/NextVR
  - **Description**: High-quality live event streaming in VR.
  - **Cost and Open-Sourceness**: Closed-source commercial.
  - **Platform**: VR headsets and some smartphones.
  - **Notes**: Targeted towards live streaming setups.

Technology selection for VR streaming use case

Our use case aimed to develop a VR streaming solution that could function on a private 5G network without relying on internet connectivity. To meet this constraint, we decided against cloud-based options. Instead, we focused on finding a technology stack that provided low latency video streaming and high compatibility across different platforms. After considering many different frameworks and protocols, we decided to use Unity Engine as our server-side rendering engine, WebRTC as the streaming protocol and WebGL/WebXR and JavaScript for the client software.

- **Unity Engine**: Unity is a popular game engine with a large community and a large repository of plugins and assets available through GitHub and the Unity asset store. Unity provides plugins which aids in the development of WebRTC streaming applications.

- **WebRTC**: WebRTC is an open-standard streaming protocol which is widely used in online meetings and other low-latency tasks. WebRTC also generally does not need a separate plugin, with support being built-in in most web browsers. This includes the web browsers in most smartphones and in VR headsets such as the Meta Quest.

  WebRTC is also one of the few protocols which have near real-time latency (HLS vs. WebRTC: What to Know Before Choosing a Protocol, n.d.).

- **WebGL/WebXR**: WebGL and WebXR are web-based technology stacks which allows for serving 3D and Virtual Reality content in web browsers using JavaScript. This allows Virtual Reality content to be served to most consumer devices without relying on native software or app stores to provide the required software.

Finally, a Node-based webserver was used to facilitate communication between our server and the client devices. The webserver serves the website with the WebXR client app, as well as acts as a signalling server for WebRTC.

Development process

Unity Server

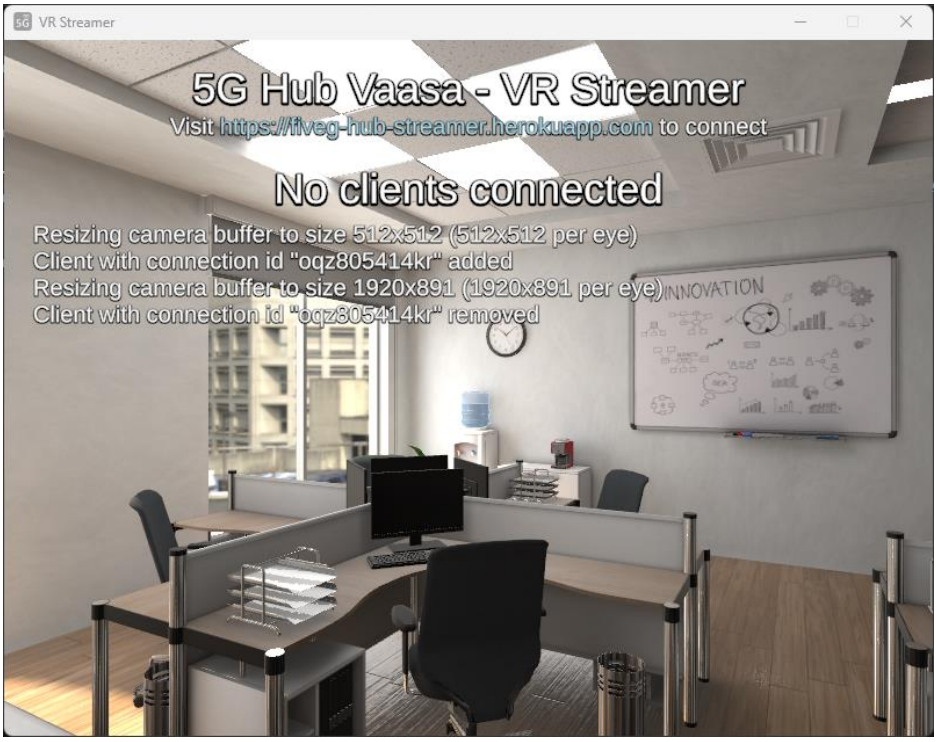
The server developed in the Unity Game Engine does the actual rendering of the virtual scene. Unity is a popular engine for game development, and for various industrial uses.

Unity has a flexible component system which makes it inherently modular. Components can be re-used, shared, and sold. They can provide pre-built functionality such as VR-support, actors, graphical effects, and support for WebRTC streaming.

Another benefit to using Unity is the Unity asset store, which contains many assets ready to be used in a project. These assets may be fully setup 3D scenes, 3D models of humans, tools, virtual-reality menus and more.

The Unity Render Streaming addon was used to provide WebRTC support. The addon provides an API for peer-to-peer streaming of Unity cameras over WebRTC (About Unity Render Streaming, n.d.).

We chose a high-fidelity and graphically demanding office scene to demonstrate the advantages of using a high-capacity server for rendering, as opposed to relying on less powerful, self-contained VR headsets.



Streaming server made in the Unity game engine

WebRTC connections

WebRTC, while fast, is not easy to work with. For a WebRTC connection to be established between two or more peers, there are several steps which must be done. The general process of setting up a peer-to-peer WebRTC connection goes like this:

1. The peers connect to a signalling server, which can be any form of communication which allows the exchange of data between the peers (even a non-digital handshake is possible).

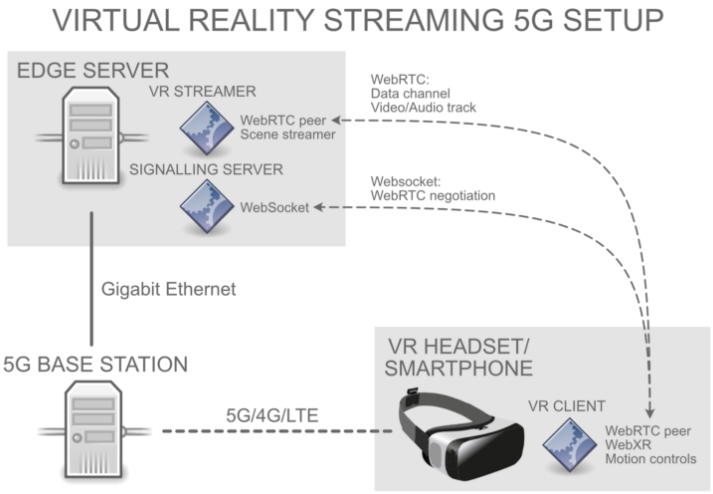
2. The peers exchange information back and forth using the signalling server. This information contains an offer with data on the media capabilities, data channels and streams. This data is sent to the other peers.
3. The other peers receive this offer and return an answer.
4. This process may happen many times for different reasons such if a new peer joined or if there are any changes to the existing streams or data channels. This is also one of the places where a developer can easily get stuck, as the process is not deterministic and requires mechanisms to handle edge cases. This could for example happen when two peers both try to send an offer at the same time, since there is no canonical server in this relationship.

For peers to communicate across the Internet and NAT servers, there are special servers called STUN/TURN servers. These are used for NAT traversal.

In addition, the peers all gather ICE (Interactive Connectivity Establishment) candidates, which are potential network paths between the peers. The peers exchange these candidates across the same signalling server as the offers and answers.

The peers can each have any number of audio, video, or data channels. These channels are used to stream data to other peers.

Many modern devices support hardware encoding for the codecs available in WebRTC, such as H.265. This is a necessity for low-latency streaming, as the server can use its graphics cards to encode the captured scene and stream it via WebRTC, and the VR headset or smartphone can use hardware decoding to quickly decode the stream.



WebRTC communication in the 5G network

Web client

For the client-side software (the software running on the smartphone or XR headset) we choose to develop a cross-platform web application based on WebGL and WebXR. These technologies enable us to create VR experiences that can be accessed on most platforms through a web browser. This includes platforms such as VR headsets, smartphones and computers running most operating systems.

The purpose of the client is to use the WebXR API provided by modern browsers to read the movement and orientation of the XR headset or smartphone. This data is sent to the Unity server via a direct peer-to-peer connection across the 5G network via a WebRTC data channel. The rendering

server aligns a virtual camera to the real-world movement of the user device, and a video of the virtual world is streamed back to the user device.

The WebGL/WebXR allows for the addition of other 3D elements to the virtual scene on the client-side. Two virtual grids were added. One on the server side and another on the client-side. The more closely these two grids align, the lower the latency between the client and server.

The web client running on an iPhone

Webserver

For our use case we decided to run a Node-based web server alongside the Unity server. Node.js is a JavaScript-based runtime environment that is used in web development as well as in many other fields. The Node Package Manager is one of the largest ecosystems in the world, providing a large array of tools for most tasks. For example, we used a module called Express which simplifies the creation of web servers.

The express web server serves the web client by hosting the static HTML, JavaScript, and CSS files.

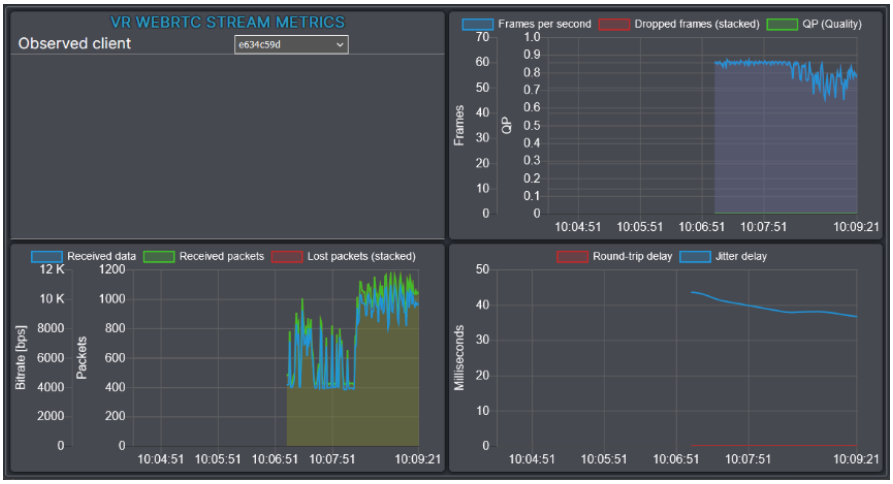
The web server also hosts the signalling server in the form of a WebSocket server, which is a TCP-socket API for web applications. This is the signalling server over which the streaming server and clients exchange WebRTC offers and answers.

Metrics dashboard

The server has the capability to collect and display WebRTC statistics. Each client which establishes a WebRTC connection to the streaming server will simultaneously gather statistics from the WebRTC Statistics API (Identifiers for WebRTC's Statistics API, n.d.). These statistics are summarized and uploaded to the webserver via a REST API.

These WebRTC metrics can then be read through a separate dashboard section of the web client. The graphs in this dashboard were created using an open-source JavaScript library called Chart.js.

The integrated metrics dashboard offers insights into performance related statistics such as frame drops, jitter buffer size, and video quality of each of the active WebRTC streams.



WebRTC metrics dashboard

Latency problem

There was a significant amount of latency when the user changes camera orientation, meaning there was a noticeable delay between the user moving the device and the picture changing. While the latency would be acceptable for uses such as web meetings, in VR streaming the latency is crucial as it makes the user's head align with the picture they see. A significant amount of delay between user action and feedback can cause motion sickness as what the user sees doesn't align with how they move.

This could have been solved using one of several methods:

- A dynamic resolution system could reduce the video stream resolution.
- Another streaming protocol or codec may have been more suited for VR streaming.
- The scene could be rendered on a static depth-mesh in front of the player, which remains stationary between frames. This would ease motion sickness as the user's head would always match the virtual head position.
- WebRTC could be configured to reduce the size of the Jitter buffer, which is the buffer which stores frames before they are displayed to the user. These settings were not yet available in the current web APIs (2023).
- The movement of the user's head could be predicted in advance to allow the streaming server to compensate for movements.

It remains to be seen what the VR streaming use case feels like with a fully capable 5G VR headset. These 5G headsets have only recently in 2023 become available to consumers.

5G-compatible VR headsets

The best test for trying out the VR-streaming solution would have been using a Virtual Reality headset with built-in 5G modem, such as those based on the Snapdragon processors. These VR headsets were not commercially available during most of the 5G Hub Vaasa project, and only really become commercially available at the end of the project.

References

About Unity Render Streaming. (n.d.). Retrieved from https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/index.html

HLS vs. WebRTC: What to Know Before Choosing a Protocol. (n.d.). Retrieved from https://www.wowza.com/blog/hls-vs-webrtc

Identifiers for WebRTC's Statistics API. (n.d.). Retrieved from https://www.w3.org/TR/webrtc-stats/